Ατομική Διπλωματική Εργασία

Δυναμικός έλεγχος λογισμικού με τη χρήση της Java Modeling Language (JML)

Παναγιώτης Χριστοφή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αύγουστος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δυναμικός έλεγχος λογισμικού με τη χρήση της
Java Modeling Language (JML)

Παναγιώτης Χριστοφή

Επιβλέπων Καθηγητής
Ανδρέας Ανδρέου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Αύγουστος 2010

Ευχαριστίες

Ολοκληρώνοντας αυτό το δύσκολο, αλλά παράλληλα και εποικοδομητικό έργο που μου ανατέθηκε θα ήθελα και εγώ με τη σειρά μου να δώσω τις ευχαριστίες μου στα άτομα που με στήριξαν και με βοήθησαν καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας.

Καταρχάς θα ήθελα να πω ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή μου Δρ. Ανδρέα Ανδρέου για την υποστήριξη και καθοδήγηση που μου πρόσφερε κατά τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας καθώς επίσης και για την επίτευξη μιας άψογης συνεργασίας.

Στη συνέχεια θα ήθελα να ευχαριστήσω τον Δρ. Αναστάση Σοφοκλέους για τη βοήθεια που μου παρείχε αλλά και για τις πολύτιμες συμβουλές που μου έδινε για να φέρω εις πέρας αυτή την εργασία.

Τελειώνοντας δε θα μπορούσα να μην αναφέρω τους γονείς μου, που πάντοτε με στήριζαν με το δικό τους τρόπο και μου συμπαραστάθηκαν όχι μόνο για τη διπλωματική μου εργασία, αλλά καθ' όλη τη πορεία μου στο πανεπιστήμιο Κύπρου στο τμήμα της πληροφορικής. Γι αυτό το λόγο τους ευχαριστώ, και θα ήταν σίγουρο ότι χωρίς αυτούς δεν θα είχα καταφέρει όσα κατάφερα μέχρι στιγμής στην παρούσα πορεία μου.

Περίληψη

Μεγάλο ενδιαφέρον προσελκύει στον καιρό μας, η δημιουργία λογισμικών case tools τα οποία βοηθούν τον χρήστη στη διεκπεραίωση διεργασιών πιο γρήγορη αλλά και πιο εύκολα. Για αυτό το λόγο και έχοντας υπόψη την μεγάλη ανάγκη που υπάρχει σε case tools τα οποία μπορούν να εντοπίζουν λάθη δυναμικά σε επίπεδο πηγαίου κώδικα σε πολύπλοκα αλλά και μεγάλα προγράμματα, αναλάβαμε την διεκπεραίωση αυτής της διπλωματικής εργασίας.

Ο σκοπός αυτής της διπλωματικής εργασίας (ΑΔΕ) είναι η δημιουργία ενός λογισμικού το οποίο θα έχει την δυνατότητα να ασκεί δυναμικό έλεγχο στις προδιαγραφές, σε σχέση με τον πηγαίο κώδικα κάποιου λογισμικού χρησιμοποιώντας δηλωτικές προτάσεις της συμπεριφοριστικής γλώσσας προδιαγραφών διαπροσωπίας Java Modelling Language (JML).

Η συγκεκριμένη εργασία αποτελεί την συνέχεια και επέκταση της ΑΔΕ του Κωνσταντίνου Ιωάννου [8] όπου μελετήθηκαν οι βασικές αρχές της JML, η αναφορά και ταξινόμηση των προγραμματιστικών λαθών και η πρόταση προτύπων ανίχνευσης αυτών των λαθών με την χρήση της JML. Καθώς επίσης και επέκταση της ΑΔΕ του Μαρίνου Αργυρού [1] όπου ασχολήθηκε με τον έλεγχο προδιαγραφών πηγαίου κώδικα με τη χρήση της JML, την εύρεση προτύπων, αλλά και τη δημιουργία plug in το οποίο ενσωμάτωνε αυτά τα πρότυπα.

Μελετώντας τις πιο πάνω διπλωματικές, στην εργασία αυτή θα προσπαθήσουμε να γενικεύσουμε τα ήδη υπάρχοντα πρότυπα που υπήρχαν αλλά και να δημιουργήσουμε καινούργια πρότυπα τα οποία μπορούν να καλύπτουν περισσότερες περιπτώσεις από τις ήδη υπάρχουσες. Κατ' επέκταση στη διπλωματική αυτή θα ασχοληθούμε μη τη δημιουργία λογισμικού που παρέχει την ευχέρεια στο χρήστη να εισάγει αυτόματα προδιαγραφές στον πηγαίο κώδικα. Με τη λειτουργία αυτή θα μπορεί να εξάγει συμπεράσματα κατά πόσο το λογισμικό του πληροί τις προδιαγραφές που απαιτείται.

Πιο αναλυτικά θα δημιουργήσουμε ένα εργαλείο το οποίο θα βοηθά το χρήστη να ελέγχει τις προδιαγραφές του συστήματος από τον πηγαίο κώδικα χρησιμοποιώντας εκφράσεις από την γλώσσα JML .

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο Έρευνας

Με την έλευση της εποχής του αυτοματισμού, σχεδόν κάθε εταιρεία εξαρτάται από ένα ή περισσότερα λογισμικά τα οποία βοηθούν στην επιτυχημένη ανάπτυξη ενός συστήματος. Η σημαντικότητα αυτών των λογισμικών απαιτεί συνεχή ανάπτυξη αλλά και συντήρηση, για να διατηρούν την αρχική τους δυνατότητα και να παρέχουν στις εταιρείες όλα τα επιθυμητά αποτελέσματα.

Αρκετές επιχειρήσεις ανάπτυξης συστημάτων στον καιρό μας, αποτυγχάνουν στην υλοποίηση λογισμικών που να εκπληρώνουν τις αρχικές προδιαγραφές που έχουν δοθεί, με αποτέλεσμα να παράγουν ημιτελής λογισμικά χωρίς την απαιτούμενη ποιότητα. Αυτό έχει ως αποτέλεσμα να μη βοηθήσει μια εταιρεία να κόψει πρώτη το νήμα στην κούρσα του ανταγωνισμού, αλλά αντιθέτως να την τροφοδοτήσει με άσχετες πληροφορίες που θα την αναγκάσουν να κυλήσει στην αποτυχία.

Ένα καλό σύστημα διασφάλισης της ποιότητας αλλά και των προδιαγραφών θα προβλέπει τα μέτρα για αντιμετώπιση για όλες τις πτυχές της λειτουργικότητας που προτίθεται να ασκεί ένα λογισμικό. Βάσει στατιστικών μετρήσεων είναι διαπιστωμένο ότι το 50% του χρόνου και κόστους διαμοιράζεται στην δοκιμή και επαλήθευση του ιδίου του υπό κατασκευή συστήματος για διασφάλιση της ποιότητας του λογισμικού.

Με βάση αυτό καταλαβαίνουμε την σημαντικότητα που έχει στις μέρες μας ο έλεγχος λογισμικού για να έχουμε και το επιθυμητό αποτέλεσμα στην εφαρμογή μας [13].
Λαμβάνοντας υπόψη όλες τις ανάγκες τις σημερινής εποχής, καταλαβαίνουμε ότι η σημερινοί επιχειρηματίες χρειάζονται ουσιαστικότερο και πιο γρήγορο τρόπο για τον έλεγχο των λογισμικών τους. Έτσι σκεπτόμενος την σημαντική προσφορά που μπορεί να δώσω στον τομέα της επιτυχημένης ανάπτυξης συστημάτων και της τεχνολογίας λογισμικού γενικότερα, αισθάνομαι ότι έχω αρκετά κίνητρα για να ασχοληθώ με αυτή την διπλωματική εργασία που, είναι σίγουρο ότι οι γνώσεις που θα πάρω θα μου είναι χρήσιμες στην μετέπειτα πορεία μου στο χώρο της τεχνολογίας λογισμικού αλλά και της επιστήμης της πληροφορικής γενικότερα.

1.2 Σκοπός έρευνας

Σε αυτή την εργασία υπάρχουν πολλοί αλλά σημαντικοί στόχοι που πρέπει να υλοποιηθούν. Αρχικά θα πρέπει να γίνει μια ολοκληρωμένη μελέτη γύρω από το θέμα του ελέγχου λογισμικού. Δηλαδή να βρεθούν οι τρόποι σε θεωρητικό επίπεδο, για έλεγχο του λογισμικού, και επέκταση έλεγχου του πηγαίου κώδικα γραμμένου στη γλώσσα Java, όπου και θα κυμανθεί αυτή η εργασία.

Παράλληλα θα διερευνηθούν η δυνατότητες της Java Modelling Language (JML), η οποία είναι μια γλώσσα περιγραφής συμπεριφοράς και προδιαγραφών, που θα μας βοηθήσει στα παρακάτω στάδια της εργασίας.

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι θα μελετηθεί το plug-in που έγινε από προηγούμενη διπλωματική εργασία του Μ. Αργυρού [1]. Θα μελετηθούν όλες οι περιπτώσεις αυτόματων προδιαγραφών για ανίχνευση των προγραμματιστικών λογικών, μαθηματικών και κατά τον χρόνο εκτέλεσης λαθών, θα γίνει τυχόν διαμόρφωση τους όπου χρειάζεται, για καλύτερη λειτουργία και χρήση. Επίσης θα δημιουργηθούν νέα πρότυπα για νέες περιπτώσεις οι οποίες θα καλύπτουν μαζί με τις προηγούμενες, πιο μεγάλο φάσμα περιπτώσεων, έτσι ο χρήστης να έχει ευελιξία στις επιλογές του.

Τελειώνοντας, και παίρνοντας γνώση από την έρευνα που έχει γίνει μέχρι στιγμής, θα δημιουργηθεί ένα λογισμικό, το οποίο θα ενσωματώνει όλα όσα ειπώθηκαν πιο πάνω και θα αυτοματοποιεί τον έλεγχο. Δηλαδή αυτό το λογισμικό θα σου επιτρέπει να εισάγεις τα πρότυπα που δημιουργήθηκαν σε μια οθόνη και θα υπάρχει το δικαίωμα επεξεργασίας του κώδικα αλλά και αλλαγής του. Μόλις ο χρήστης τελείωσει με τις αλλαγές θα μπορεί να τρέξει το πρόγραμμα και να δει τα αποτελέσματα, τα οποία θα τυπώνονται στην οθόνη. Ο χρήστης θα μπορεί να δει εάν οι αρχικές προδιαγραφές του προγράμματος του ικανοποιούνται, βλέποντας τα δεδομένα εισόδου αλλά και εξόδου που δόθηκαν στο σύστημα. Επίσης το εργαλείο αυτό θα έχει τη δυνατότητα να διαχωρίζει όλες τις δυνατές περιπτώσεις επιτυχίας και αποτυχίας, δίνοντας τυχαίες τιμές, οι οποίες θα παρουσιάζονται στον χρήστη.

1.3 Δομή Διπλωματικής Εργασίας

Το έγγραφο αυτό περιλαμβάνει έξι κεφάλαια. Το Κεφάλαιο 2 που ακολουθεί περιέχει μια εισαγωγή στην σημασία του έλεγχου λογισμικού αλλά και κατανόηση βασικών εννοιών που θα χρειαστούν στα επόμενα κεφάλαια. Επίσης θα περιγραφούν προβλήματα τα οποία θα καλεστούμε να αντιμετωπίσουμε αλλά και η επεξήγηση των διάφορων λαθών και σφαλμάτων τα οποία προκύπτουν σε διάφορα λογισμικά. Θα εξηγηθούν βασικοί ορισμοί όπως ο στατικός και δυναμικός έλεγχος, τα είδη ελέγχου λογισμικού και τα χαρακτηριστικά τους.

Στα επόμενο, Κεφάλαιο 3, θα γίνει μια σύνοψη των απαραιτήτων γνώσεων για την γλώσσα JML, η οποία θα παίξει κρίσιμο ρόλο στην δημιουργία των προδιαγραφών ελέγχου αλλά και την τελική δημιουργία του λογισμικού που θα αναφέρουμε σε πιο κάτω. Επίσης θα γίνει αναφορά στα χαρακτηριστικά της γλώσσας, την σύνταξη με παραδείγματα για πιο εύκολη κατανόηση. Στη συνέχεια θα αναφερθούμε σε μερικά από τα κύρια εργαλεία που θα μας βοηθήσουν να διαχειριστούμε και επεξεργαστούμε τις προδιαγραφές τις JML, τις οποίες θα τις χρησιμοποιήσουμε για να κάνουμε τους ελέγχους που χρειάζεται.

Στο Κεφάλαιο 4, θα παρουσιάσουμε μερικά από τα πρότυπα που θα χρησιμοποιηθούν, εκ των οποίων θα αναφερθούμε στα πρότυπα στα οποία έγιναν κάποιες αλλαγές αλλά και στα επιπρόσθετα πρότυπα που δημιουργήθηκαν για να καλύψουν πιο μεγάλο εύρος από περιπτώσεις που μπορεί να χρησιμοποιήσει ο χρήστης.

Στη συνέχεια με το Κεφάλαιο 5, θα αναφερθούμε στην δημιουργία ενός λογισμικού το οποίο θα έχει τη δυνατότητα να ασκεί έλεγχο προδιαγραφών σε πηγαίο κώδικα. Αρχικά θα αναφερθούμε στα πρότυπα που εισήχθηκαν στο λογισμικό, αλλά και τις δυνατότητες του εργαλείο αυτού. Έπειτα θα αναφέρουμε τον τρόπο που υλοποιήθηκε το συγκεκριμένο εργαλείο, και ποιες περιπτώσεις καλύπτει. Στο τέλος θα δείξουμε λεπτομερώς τον τρόπο που μπορούμε να εργαστούμε με το λογισμικό αυτό και θα δείξουμε τα αποτελέσματα στις περιπτώσεις που θα επιλέξουμε.

Τελειώνοντας στο Κεφάλαιο 6, θα αναπτύξουμε τα συμπεράσματα τα οποία δημιουργήθηκαν από την όλη έρευνα της εργασίας αυτής. Επίσης θα αναφερθούμε στο λογισμικό το οποίο δημιουργήθηκε και θα παρουσιάσουμε τα πλεονεκτήματα αλλά και τα μειονεκτήματά του. Στο τέλος του κεφαλαίου θα προταθούν μερικές μελλοντικές εργασίες που θα ήταν ωφέλιμο να υλοποιηθούν για περεταίρω επέκταση αυτής της εργασίας αλλά και του ίδιου του λογισμικού

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Έλεγχος Λογισμικού

Σαν ορισμό του έλεγχου λογισμικού, μπορούμε να πούμε ότι είναι μια εμπειρική μελέτη γύρω από ένα προϊόν, για το οποίο γίνεται μια συλλογή πληροφοριών, για να διευκρινιστεί η ποιότητα του προϊόντος με βάση το περιβάλλον αλλά και τον σκοπό που προορίζεται να εκτελέσει.
Επίσης ο έλεγχος λογισμικού μπορεί να προσφέρει μια αντικειμενική και ανεξάρτητη άποψη του λογισμικού έτσι ώστε η αγορά να κατανοήσει ακριβώς την ποιότητα αλλά και την προσφορά του λογισμικού για το έργο που έχει να αναλάβει.

Σε ένα πιο τεχνικό επίπεδο ο έλεγχος που γίνεται έχει ως άμεσο στόχο την εύρεση προγραμματιστικών λαθών, τα οποία μπορούν να οδηγήσουν το λογισμικό σε

απρόσμενες ενέργειες και ως εκ τούτου να κοστίσουν στο χρήστη πολύτιμο εργασιακό χρόνο.
Ασχολείται επίσης με την διαδικασία επικύρωσης και επαλήθευσης ενός λογισμικού με βάση τα πρωταρχικά στάδια της ανάλυσης των αναγκών αλλά και της σχεδίασης του συστήματος. Κατοχυρώνει δηλαδή ότι το λογισμικό πληροί όλα όσα πρέπει να ισχύουν από αυτά που αναμένονται να γίνουν και αυτά που έγιναν μετά την υλοποίηση του συστήματος.

Ο έλεγχος λογισμικού μπορεί να γίνει σε οποιοδήποτε στάδιο αλλά κυρίως εφαρμόζεται μετά από τον ορισμό τον προδιαγραφών στο σύστημα, δηλαδή μετά από την καταγραφή των αναγκών που πρέπει να πληροί το σύστημα όταν υλοποιηθεί.

2.2 Ποιότητα Λογισμικού

Ένα λογισμικό που ορίζεται ως υψηλής ποιότητας, είναι το λογισμικό το οποίο παραδίδεται στον χρήστη, στην προκαθορισμένη του ώρα, δεν κοστίζει περισσότερο από όσο υπολογίστηκε στην αρχή και το πιο βασικό ότι δουλεύει σωστά.
Με την φράσει δουλεύει σωστά εννοούμε ότι το λογισμικό έχει όσο το δυνατόν πιο λίγα “bugs” , αλλά και ικανοποιεί όλες τις απαιτήσεις αλλά και προδιαγραφές που ορίστηκαν και συμφωνήθηκαν από τον χρήστη.

2.3 Δυνατότητες Ελέγχου Λογισμικού

Οι δυνατότητες στον έλεγχο του λογισμικού κυμαίνονται σε συγκεκριμένα πλαίσια τα οποία αποτελούνται από διάφορες τεχνικές καθοδήγησης αλλά και εύρεσης του λάθους. Αυτές η τεχνικές λαμβάνουν υπόψη τους κατά κύριο λόγο τις προδιαγραφές του συστήματος όπως αυτές ορίστηκαν, συγκρίνουν με όμοια προϊόντα που κυκλοφορούν στην αγορά αλλά και πάνω στο ίδιο το προϊόν ασκούνται έλεγχοι δίνοντας εισόδους στο σύστημα και παρακολουθώντας τα αποτελέσματα που θα βγάλει ως έξοδο κ.α. Σε αυτή τη φάση πρέπει να διευκρινίσουμε ότι είναι αδύνατον να βρεθούν όλα τα λάθη και όλες οι ατέλειες κάποιου λογισμικού, αφού η λογική του ελέγχου λογισμικού είναι να συγκρίνει την συμπεριφορά του ίδιου του προϊόντος με βάση τις μεθόδους που

αναφέραμε πιο πάνω και να καταλήγει σε συμπεράσματα εάν το λογισμικό χρειάζεται επιπλέον δουλειά για να διορθωθεί και να μπορεί να δουλεύει όπως προβλέπεται.[14]

2.4 Λάθη, σφάλματα και τρόπος αντιμετώπισης

Ο έλεγχος λογισμικού έρχεται πολλές φορές σε διάφορα λάθη ή σφάλματα που χρήζουν διαφορετικής αντιμετώπισης για κάθε ξεχωριστή κατηγορία. Λάθη μπορεί να εμφανιστούν στον κώδικα και να είναι είτε συντακτικής είτε λογικής φύσεως. Τα συντακτικά λάθη αντιμετωπίζονται απευθείας με τη χρήστη κάποιου διερμηνέα ή μεταγλωττιστή. Συγκεκριμένα ένα πρόγραμμα δε μπορεί να μεταγλωττιστεί εάν δεν είναι συντακτικά ορθό, αλλά από την άλλη ο διερμηνέας δε μπορεί να βρει το λάθος εάν δεν το διερμηνεύσει.

Τα λογικά λάθη συμπίπτουν με την κατηγορία που ασχολείται κυρίως ο έλεγχος λογισμικού. Σε αυτή την κατηγορία εμπίπτουν τα λάθη που προκύπτουν από ελλειπείς προδιαγραφές του συστήματος, που απορρέουν από ανολοκλήρωτες απαιτήσεις του πελάτη στα αρχικά στάδια.

Σε αυτή την περίπτωση θα πρέπει το σύστημα να ελεγχθεί με βάση της προδιαγραφές που θα δοθούν και κρίνοντας έτσι, πού υπάρχουν σφάλματα και παραλείψεις στο σύστημα που υλοποιήθηκε. Ατέλειες μπορεί να υπάρχουν και στη σχεδίαση του συστήματος με αποτέλεσμα να υπάρχουν χαρακτηριστικά που να μην αντανακλούν την λειτουργικότητα του συστήματος όσον αφορά τις αρχικές προδιαγραφές.

Όλα αυτά μπορούν να αντιμετωπιστούν με τεχνικές που αναφέραμε και προηγουμένως, αλλά δεν γίνεται να εγγυηθούν την πλήρη απασφαλμάτωση του συστήματος λόγω των πολλών περιπτώσεων που θα πρέπει να ελεγχθούν.

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι τα λογικά λάθη δεν είναι το μόνο δύσκολο πρόβλημα που καλείται να λύσει ο έλεγχος λογισμικού. Μπορεί να υπάρξουν σημεία στον κώδικα που να είναι λανθασμένα, αλλά για να τα ενεργοποιήσουμε θα πρέπει να τρέξουμε το συγκεκριμένο σημείο. Εάν δεν δώσουμε αρκετά δεδομένα εισόδου τότε υπάρχει περίπτωση αυτά τα σημεία να μην εντοπιστούν κατά τη διάρκεια του ελέγχου.

2.5 Static Testing

Είναι μια μορφή ελέγχου για λογισμικό όπου το λογισμικό δεν εκτελείται, και αυτή είναι η βασική του διαφορά από τον δυναμικό έλεγχο που θα εξηγήσουμε πιο κάτω. Μπορούμε να πούμε ότι γενικά δεν είναι ένας λεπτομερής έλεγχος του λογισμικού, αλλά ελέγχει κυρίως την λογική του κώδικα και του αλγόριθμου. Ουσιαστικά είναι έλεγχος για την σύνταξη του προγράμματος και η εξέταση του κώδικα για τυχόν λάθη. Αυτός ο τύπος του ελέγχου πολλές φορές γίνεται από τον ίδιο τον προγραμματιστή που δημιούργησε το πρόγραμμα, απομονώνοντας κομμάτια κώδικα, για να ευκολύνει την διαδικασία. Επίσης στον στατικό έλεγχο μπορούν να χρησιμοποιηθούν οι μέθοδοι των reviews,των walkthroughs ή των inspections. Στη σημερινή εποχή υπάρχουν μέθοδοι οι οποίοι μπορούν να ευκολύνουν και να αυτοματοποιήσουν την διαδικασία του static testing. Μερικές από αυτές είναι οι διερμηνευτές ή οι μεταγλωττιστές προγραμμάτων οι οποίοι αναλύουν το πρόγραμμα και αναφέρουν στον προγραμματιστή για τυχών λάθη που θα εντοπίσουν.

Τα λάθη που εντοπίζονται σε αυτό το στάδιο της ανάπτυξης συστήματος είναι λιγότερο ακριβά, από μεταγενέστερα στάδια, στα οποία τα λάθη μπορεί να στοιχίσουν περισσότερο χρόνο αλλά και χρήμα.

2.6 Dynamic Testing

Είναι ο όρος που χρησιμοποιείται στον κλάδο της τεχνολογίας λογισμικού για να περιγράψει τον έλεγχο του κώδικα σε επίπεδο εκτέλεσης. Η δυναμική ανάλυση που γίνεται αναφέρεται στην εξέταση της αντίδρασης κάποιου συστήματος, το οποίο έχει μεταβλητά στοιχεία τα οποία δεν είναι σταθερά και αλλάζουν με την διάρκεια του χρόνου. Στον δυναμικό έλεγχο το λογισμικό θα πρέπει να μεταγλωττιστεί και να τρέξει. Συγκεκριμένα ο δυναμικός έλεγχος, συνεργάζεται με το πρόγραμμα, δίνοντας του δεδομένα εισόδου, τα οποία παράγονται σε δεδομένα εξόδου από το σύστημα και αυτά με την σειρά τους ελέγχονται εάν είναι τα αναμενόμενα. Δηλαδή γίνεται εκτέλεση του κώδικα με βάση ένα σύνολο από σενάρια ελέγχου Test Cases (TC). Τον δυναμικό έλεγχο μπορούμε να τον ασκήσουμε και κατά την ολοκλήρωση του λογισμικού εφόσον μπορούν να ελεγχθούν ξεχωριστά διάφορες ενότητες ή απλά διαδικασίες του προγράμματος.

2.7 Black Box Testing

Το Black Box testing είναι μια μέθοδος ελέγχου λογισμικού η οποία ελέγχει την λειτουργικότητα μιας εφαρμογής χωρίς να βασίζεται στις εσωτερικές δομές του προγράμματος. Δηλαδή δε χρειάζεται να ξέρουμε τον κώδικα, ούτε χρειάζεται να έχουμε προγραμματιστικές γνώσεις.
Τα Test Cases (TC) που χρησιμοποιούνται, δημιουργούνται με βάση την εξωτερική περιγραφή του συστήματος λαμβάνοντας υπόψη τις προδιαγραφές, αλλά και την σχεδίαση του συστήματος. Για παράδειγμα η απάντηση στην ερώτηση «Τι πρέπει να κάνει η συγκεκριμένη εφαρμογή» . Οι έλεγχοι που γίνονται μπορούν να είναι λειτουργικοί αλλά και μη λειτουργικοί. Ο ελεγκτής, επιλέγει και δοκιμάζει έγκυρα αλλά και λανθασμένα δεδομένα εισόδου και αποφασίζει τα ορθά αποτελέσματα εξόδου.
Η μέθοδος του Black Box, μπορεί να εφαρμοστεί σε όλα τα επίπεδα του λογισμικού, όπως Unit Testing, Integration Testing, System Testing [14]κ.α.

2.7.1 Unit Testing

Στον προγραμματισμό το unit testing είναι η μέθοδος που ελέγχει ξεχωριστά κομμάτια κώδικα για να κρίνει εάν είναι έτοιμα για να χρησιμοποιηθούν. Το unit testing είναι το μικρότερο κομμάτι που μπορεί να ελεγχθεί από μια συγκεκριμένη εφαρμογή.
Ο στόχος του Unit testing είναι να απομονώσει κάθε κομμάτι του προγράμματος και να αποφασίσει εάν τα κομμάτια αυτά είναι ορθά.

2.7.2 Integration Testing

Είναι ο έλεγχος κατά τον οποίον, ξεχωριστές ενότητες από το λογισμικό ενώνονται και ελέγχονται σαν σύνολο. Το Integration Testing γίνεται μετά από πολλαπλά unit tests και πριν από το system testing. Παίρνει όλα τα inputs που χρησιμοποιήθηκαν σε κάθε unit ξεχωριστά, τα συνενώνει με τρόπο τέτοιο ώστε να δημιουργηθούν inputs τέτοια ώστε να ανταποκρίνονται σε ένα πλάνο το οποίο θέλει να ελέγξει το κομμάτι του integration.

2.7.2 System Testing

Το system testing είναι ο έλεγχος που διεξάγεται σε ένα ολοκληρωμένο, σύστημα με σκοπό να αξιολογήσει εάν ακολουθεί τις προδιαγραφές του συστήματος έτσι όπως έχουν λεχθεί. Ο έλεγχος αυτός γίνεται στα επίπεδα του black box testing και έτσι δεν είναι απαραίτητη η γνώση για την εσωτερική σχεδίαση του κώδικα ή της λογικής του.
Σαν κανόνα το system testing, παίρνει ως δεδομένα εισόδου όλα τα δεδομένα που χρησιμοποιήθηκαν πιο πριν στα κομμάτια που πέρασαν τον έλεγχο του integration testing.

Ο σκοπός του integration testing είναι να ανιχνεύσει ασυνέπειες στην συνένωση του τελικού συστήματος αλλά και σε υποσυστήματα που αποτελούν το τελικό προϊόν

2.8 White Box Testing

Το λεγόμενο white box testing ή αλλιώς glass box testing είναι η μέθοδος κατά την οποία ελέγχουμε τις εσωτερικές δομές του λογισμικού, τους αλγορίθμους και γενικά την λογική του κώδικα που χρησιμοποιείται στο λογισμικό. Τα Test Cases (TC) σε αυτή τη περίπτωση, απαιτούν από τον ελεγκτή να έχει προγραμματιστική γνώση με την οποία θα δώσει στο λογισμικό δεδομένα εισόδου, για να επεξεργαστή όλες τις πιθανές εκδοχές του προγράμματος.

Υπάρχουν διάφοροι τύποι White Box Testing, που μπορούν να χρησιμοποιηθούν. Είναι το Application Programming Interface (API) Testing όπου γίνεται έλεγχος του λογισμικού με την χρήση APIs, αλλά και το code coverage όπου δημιουργούνται TC τα οποία καλύπτουν συγκεκριμένα κριτήρια για κάλυψη του κώδικα κ.α.

Το white box testing μπορεί να χρησιμοποιηθεί σε πολλές περιπτώσεις όπως Unit Testing, Integration Testing, System Testing κ.α. Αυτή η μέθοδος μπορεί να ανακαλύψει πολλά προβλήματα όμως υστερεί στην εύρεση λαθών, η ακόμα και ατελείωτων κομματιών που αφορούν τα στοιχεία των προδιαγραφών

2.9 Grey box testing

Αυτός ο τύπος ελέγχου λογισμικού είναι η ένωση του Black box και του White box Testing. Προσπαθεί να προσαρμόσει τα πλεονεκτήματα του κάθε τύπου και να τα ενώσει σε ένα ολοκληρωμένο τύπο που είναι ανώτερος από τα ανεξάρτητα στοιχεία που το απαρτίζουν. Το Grey box Testing μπορεί να πάρει την εύκολη προσέγγιση του Black box και να την ενσωματώσει στην ιδέα του White box.

Πιο συγκεκριμένα στο Grey box testing, ο ελεγκτής ελέγχει το σύστημα από έξω όπως κάνουμε και στο Black box, όμως με αυτή την τακτική γνωρίζουμε τις δομές και τους αλγορίθμους του προγράμματος , έτσι δίνουμε καλύτερα δεδομένα εισόδου που μπορούν να μας δώσουν πιο σημαντικά αποτελέσματα για να κρίνουμε το σύστημά μας.

Κεφάλαιο 3

Java Modelling Language και Προγράμματα

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε περιληπτικά μια γλώσσα η οποία θα μας είναι απολύτως απαραίτητη για την ανάπτυξη αυτής της εργασίας τόσο για της έννοιες που μπορεί να προσφέρει, αλλά και για την δημιουργία του ίδιου του λογισμικού. Αυτή η γλώσσα προγραμματισμού ονομάζεται Java Modelling Language (JML) η οποία είναι μία γλώσσα προδιαγραφών συμπεριφοριστικής διαπροσωπίας. Χρησιμοποιεί εκφράσεις οι οποίες μπορούν να χαρακτηριστούν ως pre conditions και postconditions, όπου preconditions είναι εκφράσεις οι οποίες πρέπει να είναι αληθείς ακριβώς πριν από την εκτέλεση κάποιου κομματιού κώδικα, ενώ postconditions πρέπει να είναι αληθείς άμεσος μετά την εκτέλεση αυτού του κομματιού. Η JML σαν γλώσσα έχει την δυνατότητα να προσφέρει τη σημασιολογία και τα μέσα που χρειάζεται για να περιγράψει πλήρως όλη την συμπεριφορά ενός τμήματος κώδικα γραμμένο σε Java, χωρίς καμία ασάφεια ούτως ώστε κάθε προγραμματιστής να μπορεί να καταλαβαίνει ακριβώς τι εκφράζουν οι προδιαγραφές αυτές γραμμένες σε JML. Κάθε προδιαγραφές μπορούν να οριστούν σε JML, είτε σε ξεχωριστό αρχείο όπου να συγκεντρώνονται εκεί όλες οι λεπτομέρειες των προδιαγραφών και στο ίδιο το αρχείο με αυτό του πηγαίου κώδικα σε JAVA. Εντούτοις εάν επιλέξουμε να γράψουμε τις προδιαγραφές αυτές στο αρχείο μέσα στον κώδικα θα πρέπει να τις βάλουμε εντός σχολίων ούτως ώστε να μην επηρεάζουν τον υπάρχοντα κώδικα. Η ανάγκη αυτή οδήγησε στην δημιουργία εργαλείων, για τα οποία θα αναφερθούμε πιο μετά, τα οποία να μπορούν να καταλαβαίνουν τους περιορισμούς αυτούς, ούτως ώστε να γίνεται εφικτή η χρήση των συμπεριφοριστικών πληροφοριών που παρέχει η JML. Επίσης η JML είναι γλώσσα η οποία βασίζει της ιδέες της από τους Eiffel [5], Larch η οποία είναι μία προσέγγιση για αυστηρή διατύπωση των προδιαγραφών για ενότητες προγραμμάτων και Refinement Calculus η οποία είναι μία αυστηρότερα δομημένη μορφή της μεθόδου βήμα προς βήμα εκλέπτυνσης κατά την δημιουργία ενός προγράμματος, με το σκεπτικό να προσφέρει αυστηρές και τυπικές σημασιολογίες, με τις οποίες να μπορεί να εκφράζει κάθε προδιαγραφή που θα της δοθεί από οποιονδήποτε προγραμματιστή της Java. Επίσης η

JML ακολουθεί την μέθοδο Design by Contract [5] η οποία είναι μια μέθοδος για σχεδίαση ενός λογισμικού. Την μέθοδο αυτή μπορούμε να την μεταφράσουμε ως Σχεδίαση με Συμβόλαιο, όπου μια κλάση διατηρεί ένα συμβόλαιο το οποίο έχει δυο πτυχές Η πρώτη πτυχή είναι ότι το κάλεσμα της κλάσης πρέπει να γίνει με τρόπο τέτοιο έτσι ώστε να ικανοποιούνται τα συμβόλαια που είναι σχεδιασμένα για αυτή την κλάση. Η δεύτερη πτυχή είναι ότι η κλάση με τη σειρά της κατά την επιστροφή των δεδομένων της θα πρέπει να ικανοποιεί τα συμβόλαια που της έχουν δοθεί για τα αποτελέσματα εξόδου. Αυτά τα συμβόλαια είναι τα precondition αλλά και τα post condition τα οποία έχουμε αναφέρει.[1][3][4][6][7][10]

3.2 Σύνταξη της JML και παραδείγματα

Η JML όπως έχουμε αναφερθεί μέχρι στιγμής είναι μια αυστηρή γλώσσα προδιαγραφών της JAVA. Προσδιορίζει την συμπεριφορά μιας κλάσης της JAVA, για τις αποφάσεις που πάρθηκαν και πρέπει να ακολουθηθούν κατά τη φάση της σχεδίασης αλλά και της υλοποίησης. Ο καθορισμός των προδιαγραφών γίνεται με την εισαγωγή κώδικα preconditions, post conditions και invariants. Για να μπορεί ο κώδικας της JML να χρησιμοποιηθεί εύκολα οι εκφράσεις αυτές εισάγονται σε σχόλια στο .java αρχείο μεταξύ των: /*@ … @*/ ή μετά από //@. Μερικές από της λέξεις κλειδιά που χρησιμοποιείς στην σύνταξη της η JML είναι τα (`requires`, `ensures`, `signals`, `assignable`, `pure`, `invariant`, `non null`) και μερικές άλλες εκφράσεις οι οποίες θα αναφερθούν (\old, \forall, \result , \exists) [1][11][12].

3.2.1 Preconditions και Postconditions

Τα preconditions μιας μεθόδου είναι οι προδιαγραφές οι οποίες πρέπει να ισχύουν πριν να καλεστεί η μέθοδος. Η λέξη κλειδί που καθορίζει μια precondition είναι η λέξη requires. Η σύνταξη είναι ως εξής :

```
requires amount >= 0;
```

Τα postconditions είναι οι προδιαγραφές οι οποίες πρέπει να ισχύουν μετά που θα καλεστεί η μέθοδος. Η λέξη κλειδί που καθορίζει μια precondition είναι η λέξη ensures. Η σύνταξη είναι ως εξής :

```
ensures balance == \old(balance-amount) && \result == balance;
```

Συνταχτικά τα preconditions τοποθετούνται στην αρχή και τα postconditions στο τέλος, όπως το παράδειγμα:

```
/*@ requires amount >= 0;
ensures true;
@*/
public int debit(int amount) {
...
}
```

Έκτατα postconditions μπορούν να δηλωθούν μετά τη δήλωση των ensures, με τη χρήση των signals, όπως φαίνεται πιο κάτω:

```
/*@ requires amount >= 0;
ensures true;
signals (ISOException e)amount > balance
```

3.2.2 invariants

Τα invariants μιας κλάσης είναι προδιαγραφές οι οποίες πρέπει να διατηρούνται από όλες τις μεθόδους της κλάσεις, τα οποία έχουν την ιδιότητα να βοηθούν στην κατανόηση του κώδικα . Μπορούμε να πούμε ότι τα invariants συμπεριλαμβάνονται σε όλα τα preconditions και postconditions. Επίσης τα invariants πρέπει να διατηρούνται ακόμα και όταν παρατηρείται κάποια εξαίρεση. Ένα παράδειγμα είναι το πιο κάτω:

```
public class Wallet {
public static final short MAX_BAL = 1000;
private short balance;
/*@ invariant 0 <= balance &&   balance <= MAX_BAL;
@*/
...
```

3.2.3 non_null

Μερικά από τα invariants τα preconditions και postconditions χρησιμοποιούνται για να δηλώσουν ότι μια μεταβλητή δεν προδιαγράφεται να είναι null. Η έκφραση non_null είναι μια εναλλακτική πιο γρήγορη μέθοδος που μπορεί να κάνει ακριβώς το ίδιο. Ένα παράδειγμα για το non_null είναι:

```
private /*@ non null @*/ File[] files;
void createSubdir(/*@ non null @*/ String name){
...
Directory /*@ non null @*/ getParent(){
...
```

3.2.4 Assert

Τα Asserts καθορίζουν μια προδιαγραφή η οποία πρέπει να τηρηθεί σε ένα συγκεκριμένο σημείο μέσα στον πηγαίο κώδικα. Η λέξη κλειδί assert έχει συμπεριληφθεί και στην java, εντούτοις στη JML μπορούμε να εκφραστούμε περισσότερο. Για παράδειγμα σε ένα for, while, if κτλ

```
if (i <= 0 || j < 0) {
...
} else if (j < 5) {
//@ assert i > 0 && 0 < j && j < 5;
...
} else {
//@ assert i > 0 && j > 5;
...
}
```

3.2.5 Assignable

Με την έκφραση Assignable, περιορίζουμε τις πιθανές αναθέσεις που μπορεί να συμβούν σε κάποια μέθοδο. Για παράδειγμα πιο κάτω βλέπουμε ότι στη μέθοδο debit

μπορεί να γίνει ανάθεση μόνο στο πεδίο balance. Η προεπιλεγμένη τιμή για το Assignable είναι το \everything, το οποίο σημαίνει ότι έχουμε το δικαίωμα να αναθέσουμε τιμές σε όλα τα πεδία.

```
/*@ requires amount >= 0;
assignable balance;
ensures balance == \old(balance)-amount;
@*/
public int debit(int amount) {
...
```

3.2.6 Pure

Το modifier pure αφορά μόνο μεθόδους και κατασκευαστές. Δηλώνει ότι κατά την εκτέλεση της μεθόδου κανένα από τα πεδία της δε μπορεί να πάρει τιμή και να επιφέρει κάποια αλλαγή στην κατάσταση του προγράμματος. Με άλλα λόγια η μέθοδος μπορεί να χαρακτηριστεί ως και Assignable είναι το \nothing.

```
public /*@ pure @*/ int getBalance(){...
Directory /*@ pure non null @*/ getParent(){...
```

3.2.7 Visibility

Η JML υποστηρίζει όλες τις βασικές ορατότητες που υποστηρίζει και η JAVA. Συγκεκριμένα υποστηρίζει τις ορατότητες public, protected και private. Η JML έχει την δυνατότητα να χαλαρώνει της ορατότητες κάποιων στοιχείων της JAVA για να μπορέσει να τις χρησιμοποιήσει για λόγους προδιαγραφών. Συγκεκριμένα όταν ένα στοιχείο είναι private, μπορεί να τα κάνει σε public βάζοντας μπροστά την έκφραση spec_public. Φυσικά το ίδιο ισχεί και για τις άλλες ορατότητες.

```
public int pub;
```

```
private /*@ spec public @*/ int priv;
//@ requires i <= pub && i <= priv; // OK
public void pub2(int i) { ... }
```

3.2.8 Εκφράσεις JML

Οι εκφράσεις τις JML είναι ένας επιπλέον τρόπος που μας δίνει την δυνατότητα να μπορούμε να αλληλεπιδράσουμε με τις διάφορες κλάσεις της Java, τις μεθόδους αλλά και τους τελεστές που χρησιμοποιούνται. Με αυτόν τον τρόπο μπορούμε να περιγράψουμε το πρόγραμμα μας με μια πιο δυνατή σύνθεση προδιαγραφών. Μερικές από τις εκφράσεις είναι:

3.2.8.1 \old

Με την έκφραση αυτή μπορούμε να συγκρατήσουμε στη μνήμη την τιμή μιας μεταβλητής η οποία εισίλθε στην μέθοδο.

3.2.8.2 \forall

Είναι ένας ποσοτικός τελεστής που ελέγχει την συνθήκη που θα ακολουθήσει και εάν επαληθεύεται τότε επιστρέφει true, αλλιώς επιστρέφει false.

3.2.8.3 \result

Με την έκφραση αυτή μπορούμε να συγκρατήσουμε το αποτέλεσμα της επιστροφής μιας μεθόδου. Δε χρειάζεται να ορίσουμε τον τύπο της, αφού παίρνει τον τύπο του αντικειμένου ή της μεταβλητής που επιστρέφεται.

3.2.8.4 \exists

Ο ποσοτικός τελεστής που επιστρέφει true εάν έστω για μία τιμή μέσα στο πεδίο τιμών ισχύει το κατηγόρημα που ορίζεται.

3.3 Εργαλεία για JML

Για να μπορέσουμε να επεξεργαστούμε τις προδιαγραφές σε JML έπρεπε κατά κύριο λόγο να βρούμε κάποια εργαλεία τα οποία να μπορούν να κατανοούν την γλώσσα προδιαγραφών συμπεριφοριστικής διαπροσωπίας JML, και να εντοπίζουν εάν το πρόγραμμα που θα ακολουθεί πληροί τις προδιαγραφές που του έχουν δοθεί. Παρόλο που η JML μπορεί να χρησιμοποιηθεί για στατική ανάλυση των προδιαγραφών, η οποία ανάλυση δεν χρειάζεται να μεταγλωττίσει τον κώδικα JAVA που ακολουθεί εντούτοις μπορεί να χρησιμοποιηθεί για να επαληθεύσει τα αποτελέσματα του πηγαίου κώδικα σε μια εφαρμογή, έτσι υπάρχουν και εργαλεία τα οποία ασκούν αυστηρό δυναμικό έλεγχο στα διάφορα αποτελέσματα του κώδικα, τα οποία αφορούν τη δυναμική αλλαγή μεταβλητών χειριστηρίων κ.α. [10][11].

3.3.1 JML2 Eclipse Plug-In

Το εργαλείο JML2 Eclipse Plugin είναι το εργαλείο το οποίο χρησιμοποιήθηκε περισσότερο στην εργασία αυτή. Ο λόγος είναι ότι έχει ενσωματωμένα όλα τα βασικά εργαλεία της JML στο περιβάλλον ανάπτυξης λογισμικού Eclipse. Τα εργαλεία τα οποία μπορούν να χρησιμοποιηθούν είναι τα jmlrac, jmlc, jmldoc, τα οποία και θα εξηγηθούν πιο κάτω. Όλα τα εργαλεία που υπάρχουν μέσα στο Eclipse IDE είναι ενσωματωμένα με τέτοιο τρόπο ώστε να μπορούν να αποδίδουν πλήρως στις απαιτήσεις της γλώσσας προδιαγραφών JML σε συνδυασμό πάντα με τον κώδικα γραμμένο σε JAVA [10][12].

3.3.2 Jmlc

Ο JML Compiler είναι το βασικό εργαλείο το οποίο παίρνει σαν είσοδο του ένα αρχείο κώδικα JAVA που περιέχει προδιαγραφές γραμμένες σε JML και μεταγλωττίζει ολόκληρο το αρχείο. Μεταγλωττίζει τα JML statements τα οποία τα μετατρέπει σε runtime assertion checks για να μπορούν να χρησιμοποιηθούν κατά τη λειτουργία του προγράμματος. Επίσης μεταγλωττίζει και τον κώδικα Java, στον οποίου ασκεί τους ελέγχους που βρήκε από την JML. Εάν υπάρχει παραβίαση τον assertions αυτών τότε ενημερώνει τον προγραμματιστεί ότι υπήρχε λάθος στις προδιαγραφές.

3.3.3 Jmlrac

Με τη χρήση του jmlrac μπορούμε να συνδυάσουμε τις απαιτούμενες βιβλιοθήκες για ανίχνευση των παραβιάσεων των προδιαγραφών, που έχουν καταγραφεί με τον JML Compiler, και επίσης δίνει την δυνατότητα στο χρήστη να τρέξει το εκτελέσιμο αρχείο με τις προδιαγραφές για ανίχνευση των λαθών.

3.3.4 Jmldoc

Το εργαλείο αυτό λειτουργεί ως εναλλακτικό του Javadoc, δηλαδή είναι ένα εργαλείο το οποίο μετατρέπει τα χαρακτηριστικά των κλάσεων, τα σχόλια και τις δηλώσεις JML σε αρχεία .html τα οποία τεκμηριώνουν τον κώδικα και τις ιδιότητες του.

3.3.5 AutoJML

Με το AutoJML μπορούμε να δημιουργήσουμε αυτόματα προδιαγραφές σε JML, βάσει του σκελετού του πηγαίου κώδικα σε JAVA αλλά και από άλλες προδιαγραφές που μπορεί να υπάρχουν σε JML. Το πρόγραμμα αυτό παράγει αυτόματα τις προδιαγραφές, χρησιμοποιώντας μία υψηλότερου επιπέδου γλώσσα καθορισμού συμπεριφορών όπως τα πρωτόκολλα προδιαγραφών ασφάλειας αλλά και τα Διαγράμματα καταστάσεων UML.

3.3.6 ESC/Java2

Το ESC/Java2 είναι ένα εργαλείο το οποίο χρησιμοποιεί την μέθοδο του *Extended Static Checker for Java version2*) Αυτό το προγραμματιστικό εργαλείο προσπαθεί να ανιχνεύσει κοινά λάθη τα οποία προκύπτουν κατά τον χρόνο εκτέλεσης αναλύοντας στατικά τον πηγαίο κώδικα Java και τις αυστηρές προδιαγραφές JML που έχουν ορισθεί. Οι χρήστες μπορούν να ελέγξουν το μέγεθος και τα διάφορα είδη ελέγχου που πραγματοποιεί το ESC/Java2 συμβολίζοντας τα προγράμματα με σχόλια συγκεκριμένης τροποποίησης που λέγονται *pragmas* [9][10][11].

3.4 Άλλα Εργαλεία

Εκτός από τα εργαλεία τα οποία χρειαστήκαμε για να δουλέψουμε με τις προδιαγραφές της JML, υπήρχαν κι άλλα εργαλεία τα οποία αποδείχτηκαν εξίσου χρήσιμα και αποδοτικά για να έρθει εις πέρας η εργασία αυτή, αυτά είναι το NetBeans IDE, και το FileTypesMan.

3.4.1 NetBeans IDE

Το NetBeans (Integrated Development Environment) είναι μια μια ολοκληρωμένη εφαρμογή ανοιχτού κώδικα, για παραγωγή προγραμμάτων. Προσφέρει όλα τα εργαλεία που χρειάζονται για να δημιουργήσει επαγγελματικές εφαρμογές που αφορούν ιστοσελίδες , επιχειρησιακά προγράμματα, εφαρμογές για κινητά τηλέφωνα κ.α. Το NetBeans IDE, προσφέρει την δυνατότητα προγραμματισμού σε γλώσσες προγραμματισμού όπως:Java , JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, κ.α. Στη περίπτωση μας το NetBeans, επιλέχτηκε για την ικανότητα του να προγραμματίζει σε JAVA αλλά και τα ενσωματωμένα εργαλεία του τα οποία σου προσφέρουν ένα ολοκληρωμένο σύνολο επιλογής από εργαλεία για την κατασκευή διαπροσωπιών λογισμικού.

3.4.2 FileTypesMan

Το FileTypesMan είναι μια εφαρμογή η οποία σου εμφανίζει μια λίστα με όλες τις επεκτάσεις των αρχείων και του τύπους τους εγκατεστημένα στον ηλεκτρονικό υπολογιστή. Για κάθε τύπου αρχείο το πρόγραμμα αυτό σου επιτρέπει να δεις αρκετές πληροφορίες όπως την περιγραφή του, το όνομα του τύπου του, τον αντιληπτό τύπο του κάποιες σημαίες που χρησιμοποιεί κ.α. Με το FileTypesMan μπορούμε εύκολα να επεξεργαστούμε τις ιδιότητες και τις σημαίες που προδιαγράφουν κάθε τύπο προγράμματος καθώς επίσης μας επιτρέπει να εισάγουμε ή και να διαγράψουμε κάποιες ενέργειες που θέλουμε για οποιοδήποτε τύπο αρχείου.

Κεφάλαιο 4

Προτεινόμενα Πρότυπα και προγραμματιστικά λάθη

4.1 Εισαγωγή

Ο μεγάλος όγκος από τα λάθη και από τις απροσεξίες που γίνονται στον πηγαίο κώδικα ενός προγράμματος, που πολλά από αυτά προέρχονται κυρίως από τις ίδιες αιτίες που

συνεχώς επαναλαμβάνονται, οδήγησε στην ανάγκη για δημιουργία κάποιου λογισμικού που να μπορεί να βοηθήσει στον ελαχιστοποίηση αυτών των προβλημάτων. Ένα λογισμικό το οποίο να έχει αυτοματοποιημένες μεθόδους οι οποίες να μπορούν να χρησιμοποιηθούν εύκολα και γρήγορα. Ο K. Ιωάννου [8] και ο Μ. Αργυρού [1] πρότειναν σε προηγούμενες μελέτες πρότυπα ανίχνευσης αυτών των λαθών με την χρήση της JML. Αυτά τα πρότυπα μπορούν να ενσωματωθούν εύκολα και γρήγορα στον κώδικα, με απλές αυτοματοποιημένες μεθόδους. Επίσης με την ιδέα αυτή μπορούμε να ενσωματωθούν σε ένα λογισμικό οι κυριότερες αιτίες λαθών [2] και ο προγραμματιστής να μπορεί να ελέγχει ταυτόχρονα πολλές αιτίες λαθών που μπορεί να προκαλούνται στο πρόγραμμα.

Πιο κάτω θα ξαναθυμηθούμε τα πρότυπα που προτάθηκαν και εξηγήθηκαν από τους K. Ιωάννου [8] και ο Μ. Αργυρού [1], τα οποία θα χρησιμοποιηθούν στο λογισμικό που θα παρουσιαστεί σε μετέπειτα ενότητα. Επίσης τα πρότυπα αυτά μπορεί να υπέστηκαν κάποιες αλλαγές οι οποίες θα εξηγηθούν.
Οι πρόταση αυτών των προτύπων, οδήγησε σε περισσότερη έρευνα, για εύρεση περισσότερων προτύπων και περιπτώσεων που μπορεί να βοηθήσουν τον χρήστη. Τα καινούργια πρότυπα θα εξηγηθούν με πιο πολλή λεπτομέρεια στο τέλος της ενότητας αυτής.

4.2 Προτεινόμενα Πρότυπα Λαθών σε JML

Όπως έχει αναφερθεί, σε αυτήν την ενότητα θα εξηγηθούν τα πρότυπα που έχουν προταθεί από προηγούμενες μελέτες ανίχνευσης λαθών με την χρήση της JML, έτσι ώστε να γίνει κατανοητή η λειτουργία τους στο λογισμικό που θα υλοποιηθεί.

4.2.1 Πρότυπο Division By Zero

Το πρότυπο αυτό ανιχνεύει το λάθος υπολογισμού (**Computation error**) που μπορεί να γίνει όταν διαιρέσουμε με το μηδέν σε μια μαθηματική πράξη. Ο τρόπος ανίχνευσης του είναι με το να θέσει περιορισμό ότι η μεταβλητή που διαιρείται δε μπορεί να έχει την τιμή μηδέν, καθώς επίσης και δε δέχεται η μέθοδος να επιστρέψει τιμή μηδέν, επειδή εάν υπάρχει το μηδέν τότε υπάρχει και το ενδεχόμενο να δημιουργηθεί το λάθος.

4.2.2 Πρότυπο Array Index Out Of Range

Αυτό το πρότυπο ανιχνεύει το λάθος ελέγχου ορίων (**Boundary checking errors**) . Συγκεκριμένα ανιχνεύει την προσπάθεια πρόσβασης σε θέση μεγαλύτερη του μεγέθους ενός πίνακα. Ο τρόπος που λειτουργεί η ανίχνευση είναι, πρώτα με τον περιορισμό στο χειριστήριο να έχει null τιμή. Έπειτα ελέγχει εάν ο βρόγχος που θα υλοποιηθεί στην μέθοδο είναι εντός των πλαισίων του πίνακα και η τελευταία τιμή στον πίνακα να δοθεί ως αποτέλεσμα σε κάποια μεταβλητή. Σημειώνεται ότι εδώ έγινε διόρθωση στην έκφραση που μόλις σχολιάστηκε, επειδή η τιμή που δινόταν στο τέλος δεν ελεγχόταν σωστά.

4.2.3 Πρότυπο String Index Out Of Range

Πρότυπο το οποίο ανιχνεύει λάθος ελέγχου ορίων(**Boundary checking errors**), δηλαδή την προσπάθεια για πρόσβαση σε θέση μεγαλύτερη από το μήκος μία συμβολοσειράς. Ο τρόπος που δουλεύει η συγκεκριμένη ανίχνευση, είναι με τον περιορισμό ότι η μεταβλητή τύπου string, δεν πρέπει να έχει null τιμή καθώς επίσης και με την εισαγωγή μιας έκφρασης assert η οποία ειδοποιεί τον προγραμματιστή εάν το όρια της συμβολοσειράς ξεπεραστούν.

4.2.4 Πρότυπο Array Store Exception

Το πρότυπο αυτό ανιχνεύει λάθη τύπων δεδομένων (Data Type) και συγκεκριμένα ανιχνεύει την προσπάθεια για καταχώρηση δεδομένων σε πίνακα μη συμβατού τύπου. Το πρότυπο παρακολουθεί τον μετρητή να είναι μικρότερος από το μέγεθος του πίνακα, τον βρόγχο της μεθόδου να είναι εντός των πλαισίων καθώς επίσης ο τύπος του πίνακα να είναι ο τύπος που ορίστηκε. Επίσης ανιχνεύει ότι ο τύπος της μεταβλητής θα είναι συμβατός με αυτόν του πίνακα.

4.2.5 Πρότυπο Dereferencing Null

Στη συνέχεια το πρότυπο Dereferencing Null ανιχνεύει λάθη Εγκυρότητας Δεδομένων (**Data Validation**). Δηλαδή είναι υπεύθυνο να παρακολουθεί για τυχόν μη

αρχικοποίηση δημιουργίας αντικειμένου μέσα στην μέθοδο. Ελέγχει ότι κατά τη διάρκεια που χρησιμοποιείται η μέθοδος, η μεταβλητή που είναι υπεύθυνη για να αρχικοποιήσει τον πίνακα θα κυμαίνεται μεταξύ του μηδενός και του μεγέθους του πίνακα που θέλουμε να αντιγράψουμε. Σαν τελικό έλεγχο το πρότυπο είναι υπεύθυνο για να ανιχνεύει τη μέθοδο να μην επιστρέψει τιμή null, δηλαδή να αποτύχει στην αρχικοποίηση του χειριστηρίου του πίνακα.

4.2.6 Πρότυπο Stack Overflow Area

Το επόμενο πρότυπο ανιχνεύει λάθη ελέγχου ορίων (**Boundary checking errors**) που στην περίπτωση αυτή ελέγχει εάν έγινε πρόσβαση σε θέση μεγαλύτερη από το μέγεθος μίας στοίβας. Συγκεκριμένα στο πρότυπο αυτό η μέθοδος προσπαθεί να βρει το παραγοντικό ενός αριθμού. Εάν όμως προσπαθήσουμε να βρούμε το παραγοντικό κάποιου αριθμού μεγαλύτερου του 12, τότε το αποτέλεσμα δε θα μπορεί να φυλαχτεί στη μνήμη ενός integer, αφού όπως είναι γνωστό ο μεγαλύτερος αριθμός που μπορεί να φυλαχτεί στη μνήμη ενός integer είναι ο 2^{32}-1. Στη περίπτωση μας για να έχουμε σωστά αποτελέσματα οι εκφράσεις JML, προδιαγράφουν το σύστημα να μην υπολογίζει το παραγοντικό που είναι μικρότερο του ενός ή μεγαλύτερο του 13 για να μην υπάρξουν σφάλματα.. Επίσης υπάρχει ο περιορισμός ο οποίος ανιχνεύει ότι το αποτέλεσμα θα είναι μικρότερο από την μεγαλύτερη τιμή που μπορεί να πάρει μια integer μεταβλητή, δηλαδή το 4294967295.

4.4.7 Πρότυπο Unsorted Table

Είναι υπεύθυνο για λάθη διαχείρισης δεδομένων(**Data Handling**) όπου ανιχνεύει ένα μη ταξινομημένο πίνακα. Πιο αναλυτικά στο πρότυπο αυτό έχουμε την εισαγωγή ενός πίνακα στην μέθοδο ο οποίος προδιαγράφεται να είναι ταξινομημένος. Έτσι υπάρχουν οι εκφράσεις που παρακολουθούν το χειριστήριο του πίνακα να μην είναι null και η το μέγεθος του πίνακα να είναι μεγαλύτερο από το μηδέν. Επίσης ο πίνακας ελέγχεται εάν τα στοιχεία του είναι ταξινομημένα , για να μπορεί να προχωρήσει χωρίς πρόβλημα η δυαδική αναζήτηση που ακολουθεί.

4.2.8 Πρότυπο Infinite Loop Due To *Mathematical* Operator

Ττο πρότυπο αυτό διαχειρίζεται λάθη αποφάσεων συνθήκης **(Decision statements)** αφού συναντάμε πρόκληση ατέρμονου βρόγχου λόγω λανθασμένης χρήσης μαθηματικού τελεστή μέσα στον βρόγχο του while. Για να ελεγχθεί το λάθος, αυτό το πρότυπο δήλωσε εκφράσεις οι οποίες ελέγχουν ότι οι δυο μεταβλητές που θα χρησιμοποιηθούν στην συνθήκη είναι μεγαλύτερες του μηδενός. Επιπλέον υπάρχει η συνθήκη όπου η αφαίρεση των δύο αριθμών δεν πρέπει να μας δίνει αρνητικό αριθμό, οπόταν βεβαιώνουμε ότι ο μετρητής είναι μεγαλύτερος από το μέγεθος που ορίσαμε και ότι ο μετρητής μειώνεται κατά ένα κάθε φορά μέσα στον βρόγχο.

4.2.9 Πρότυπο Infinite Loop Due To Logical Operator

Στη συνέχεια το πρότυπο που ακολουθεί διαχειρίζεται λάθη αποφάσεων συνθήκης **(Decision statements)** αφού συναντάμε πρόκληση ατέρμονου βρόγχου λόγω λανθασμένης χρήσης λογικού τελεστή. Ο προγραμματιστής θέλει να μετρήσει πόσα στοιχεία του πίνακα είναι μεγαλύτερα του 5, και στη περίπτωση αυτή ο λογικός τελεστής έχει τοποθετηθεί λανθασμένα. Ο τρόπος αντιμετώπισης του λογικού αυτού σφάλματος πρώτα με την προϋπόθεση ότι το χειριστήριο του πίνακα δεν είναι null. Επίσης ελέγχεται το αποτέλεσμα της μεθόδου ότι ο μετρητής του βρόγχου θα είναι ενδιάμεσα στο μήκος του πίνακα καθώς επίσης ελέγχει την ποσότητα των στοιχείων που βρέθηκαν να είναι μεγαλύτερα από τον αριθμό που δόθηκε και εάν εν τέλει μηδενίστηκαν τα στοιχεία που δεν είναι μεγαλύτερα από αυτόν τον αριθμό.

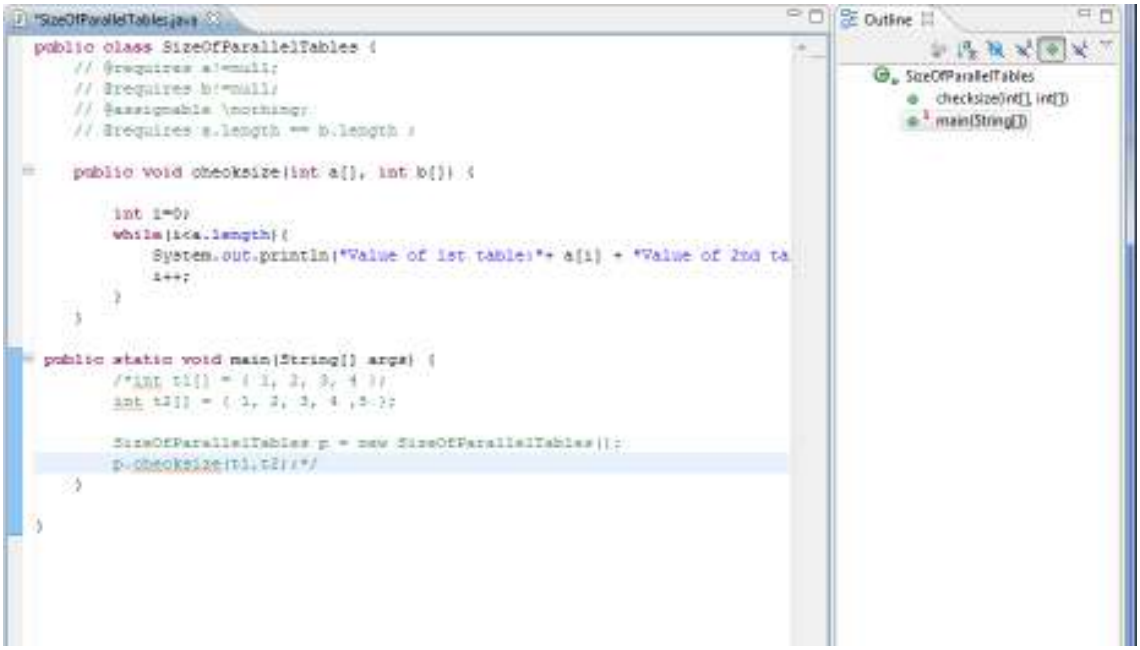
4.2.10 Πρότυπο Negative Array Size

Ακολούθως θα χρησιμοποιηθεί και το πρότυπο negative array size, το οποίο ανιχνεύει λάθη Ελέγχου(Control errors) αλλά πιο συγκεκριμένα την εισαγωγή αρνητικού μεγέθους σαν θέση στον πίνακα. Αυτό επιτυγχάνεται με την χρήση εκφράσεων που ελέγχουν ότι η τιμή που θα δοθεί στην αρχικοποίηση της τιμής του πίνακα θα είναι μεγαλύτερη του μηδενός, καθώς επίσης ότι το χειριστήριο που θα επιστρέψει σαν αποτέλεσμα η μέθοδος δεν θα είναι null.

Τα επόμενα πρότυπα που θα ακολουθήσουν είναι τα πρότυπα που δημιουργήθηκαν σε αυτή την διπλωματική για της ανάγκες της δημιουργίας του λογισμικού που θα ακολουθήσει. Παρακάτω θα δοθεί μια λεπτομερής αναφορά στα πρότυπα αυτά εξηγώντας τα λάθη που ανιχνεύουν, τον τρόπο λειτουργίας και γενικά τα προβλήματα που αντιμετωπίζουν.

4.2.11 Πρότυπο Diff Size of Parallel Table

Στο πρότυπο αυτό αντιμετωπίζουμε την πιθανότητα δημιουργίας και χρησιμοποίησης δυο παράλληλων πινάκων με αλληλοεξαρτημένα δεδομένα, που να μην έχουν το ίδιο μήκος με κίνδυνο λάθους στο πρόγραμμα.



Σχήμα 4.1 : Πρότυπο για Diff Size of Parallel Table

Όπως φαίνεται πιο πάνω ο προγραμματιστής δημιουργεί τους πίνακες , τον ένα μετά τον άλλο, αλλά στον ένα δίνει πιο πολλά δεδομένα. Στη συνέχεια, στέλνει τους δυο πίνακες στην μέθοδο, και προσπαθεί να τυπώσει παράλληλα τα αποτελέσματα. Στο τέλος του βρόγχου θα δημιουργηθεί λάθος επειδή θα ξεπεράσει τα όρια του πρώτου πίνακα με τα πιο λίγα στοιχεία. Για τον έλεγχο αυτού του λάθους ορίζονται προδιαγραφές στο πρόγραμμα με εκφράσεις JML.

Για να ήμαστε σίγουροι ότι τα χειριστήρια έχουν κάποια τιμή, χρησιμοποιούμε τις εκφράσεις **requires a!=null** και **requires b!=null** που ορίζουν τα pre-conditions της μεθόδου, καθορίζοντας πως για να εκτελεστεί η μέθοδος πρέπει να ισχύουν οι συνθήκες αυτές. Οπότε αποκλείεται να έχουμε τιμή null λόγω αυτών των προδιαγραφών. Επίσης χρησιμοποιείται και η έκφραση **requires a.length == b.length**, η οποία ελέγχει ότι οι δύο πίνακες έχουν το ίδιο μήκος, έτσι θα μπορούμε να εργαστούμε πάνω σε αυτούς παράλληλα χωρίς να υπάρχει κάποιο σφάλμα.

4.2.12 Πρότυπο Diff Type Equation Exception

Ακολούθως φθάνουμε στην περίπτωση θέλουμε να αντιμετωπίζουμε το συχνό φαινόμενο κάποιας πράξης μεταξύ δύο αταίριαστου και διαφορετικού τύπου μεταβλητών, που είναι υπεύθυνο για την δημιουργία ενός σφάλματος που μπορεί να περάσει απαρατήρητο.

Σχήμα 4.2 : Πρότυπο για Diff Type Equation Exception

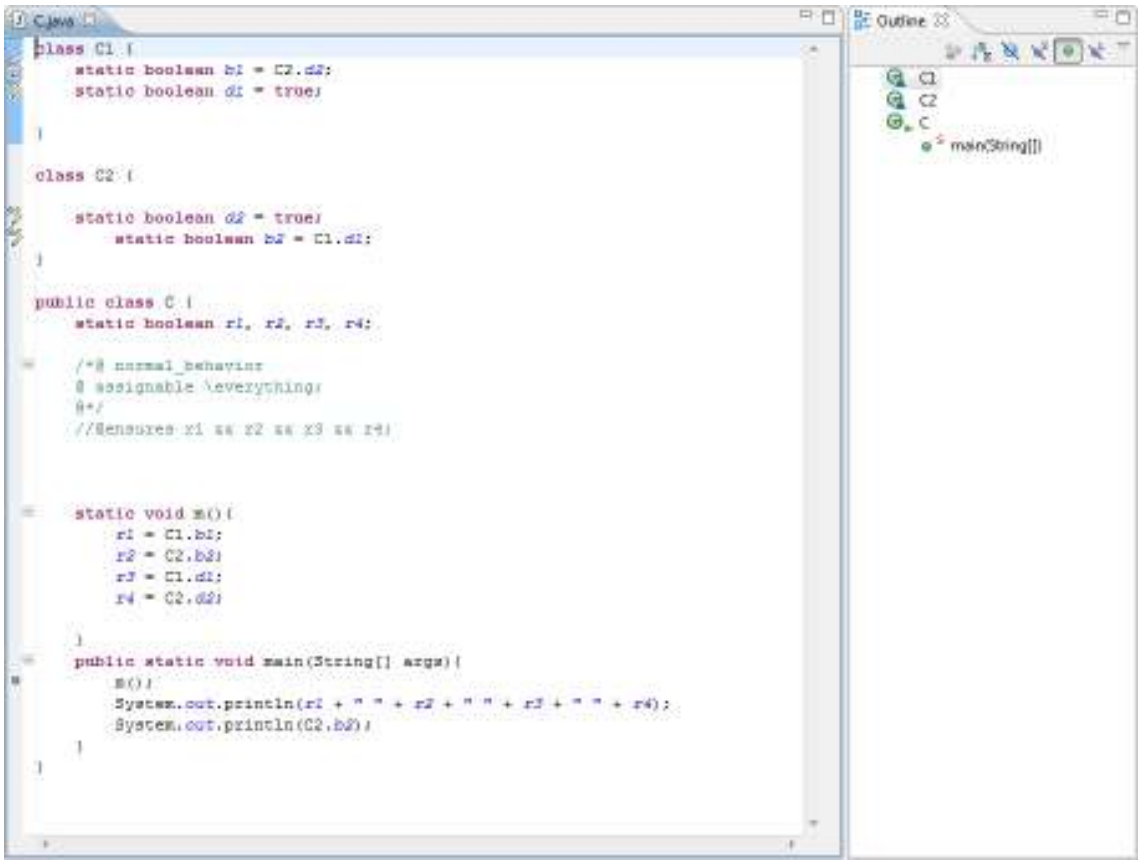
Στον πιο πάνω πηγαίο κώδικα, υπάρχει μια μεταβλητή τύπου String που εμπεριέχει ένα αριθμό, και μια μεταβλητή τύπου Integer που και αυτή αρχικοποιείται με έναν αριθμό. Στο πιο πάνω κώδικα, βλέπουμε ότι ο προγραμματιστής προσπαθεί να προσθέσει τις δυο μεταβλητές. Το αποτέλεσμα που θα πάρει όμως δεν θα είναι το άθροισμα των δυο

αριθμών αλλά οι συνένωση τους σαν ένα string. Οι προδιαγραφές τους προγράμματος όμως ορίζουν ότι αυτές οι δυο μεταβλητές, δε πρέπει να έχουν διαφορετικό τύπο μεταξύ του, για να μπορέσει να γίνει σωστά η μαθηματική πράξη.

Για να το πετύχουμε αυτό χρησιμοποιούμε τις μεταβλητές στο πάνω μέρος της κλάσης, και ελέγχουμε τον τύπο τους με βάση το pre-condition της μεθόδου όπου τοποθετούμε την έκφραση **/*@requires \typeof(num) <: \typeof(str);@*/** ,το οποίο καθορίζει πώς για να εκτελεστεί η μέθοδος. Κατά τον έλεγχο της συγκεκριμένης κλάσης οι τιμές που χρησιμοποιεί η μέθοδος είναι βέβαιο ότι θα είναι είναι ίδιου τύπου σύμφωνα πάντα τους κανόνες που προσφέρει η JML.

4.2.13 Πρότυπο Static Initialization

Σε αυτό το πρότυπο παρουσιάζεται ο κώδικας σε Java και JML για το λάθος χρόνου εκτέλεσης που παρατηρείται όταν θέλουμε να αρχικοποιήσουμε static μεταβλητές, που εάν δεν ήταν static τότε δεν θα είχαμε το πρόβλημα αυτό.



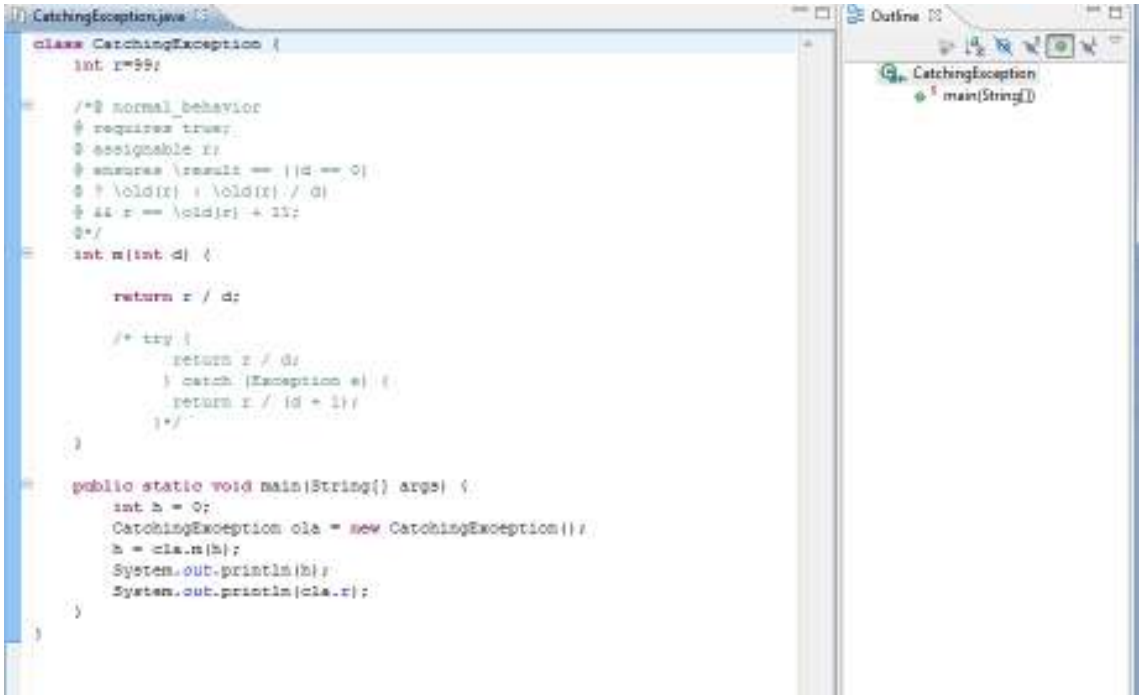
Σχήμα 4.3 : Πρότυπο για Static Initialization

Στο πιο πάνω πρότυπο βλέπουμε ότι υπάρχουν τρεις διαφορετικές κλάσεις. Στην κεντρική κλάση την C δηλώνουμε τέσσερεις Boolean μεταβλητές τις οποίες τις αρχικοποιούμε με βάση τις τιμές που εμπεριέχονται στις άλλες κλάσεις. Η λογική που ακολουθείται στην αρχικοποίηση των μεταβλητών στο πιο παράδειγμα είναι η εξής: Αρχικά η πρώτη μεταβλητή αρχικοποιείται από την μεταβλητή b1 που είναι στην C1. Η b1 με την σειρά της μόλις καλεστεί καλεί την κλάση C2. Στην C2 το d2 παίρνει την τιμή true και το b2 παίρνει την τιμή από το d1 που είναι στην C1 και δεν έχει καλεστεί ακόμα, άρα παίρνει την τιμή false. Προχωρώντας στο επόμενο το r2 την τιμή του b2 η οποία δηλώθηκε προηγουμένως ως false. Το r3 αρχικοποιείται με την τιμή του d1 η οποία είναι true και τέλος το r4 παίρνει την τιμή του d2 η οποία είναι true. Όπως βλέπουμε εύκολα μπορεί ο προγραμματιστής να κάνει κάποιο λάθος με την διαδικασία αρχικοποίησης στις static αυτές μεταβλητές.

Έτσι δηλώνουμε μια post-condition έκφραση **//@ensures r1 && r2 && r3 && r4;** που τοποθετείται πριν από την μέθοδο και προδιαγράφει ότι το αποτελέσματα της μεθόδου αυτής θα πρέπει να είναι true, άρα και κατά συνέπεια όλες οι μεταβλητές πρέπει να αρχικοποιηθούν με την σωστή μεταβλητή που είναι η true.

4.2.14 Πρότυπο Catching Exception

Στην περίπτωση αυτή , ελέγχεται εάν κάποιες εξαιρέσεις που μπορεί να προκαλέσουν πρόβλημα στο πρόγραμμα διαχειρίζονται καταλλήλως από τον προγραμματιστή για να παρεμποδίσουν το σφάλμα.



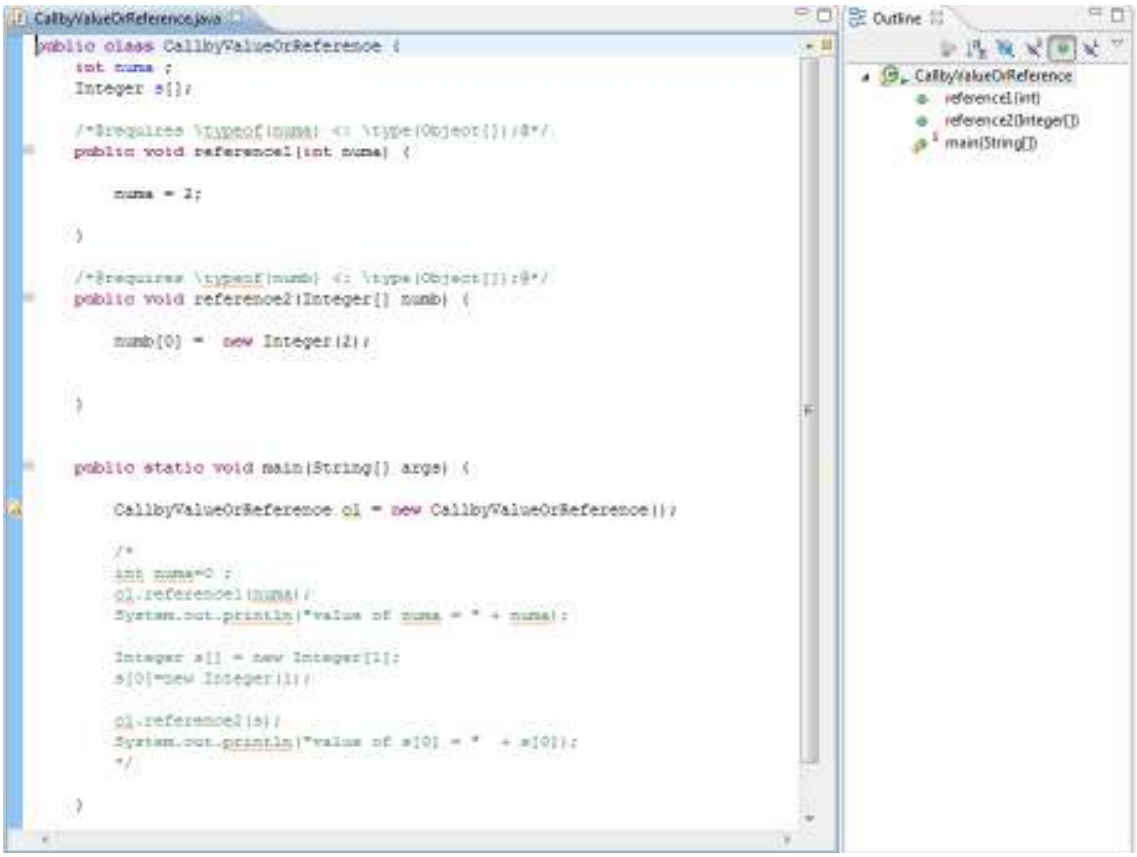
Σχήμα 4.4 : Πρότυπο για Catching Exception

Η εξαίρεση στον πιο πάνω κώδικα όπως φαίνεται είναι η περίπτωση όπου η μεταβλητή παίρνει την τιμή μηδέν και δημιουργεί το σφάλμα διαίρεσης κάποιου αριθμού με το μηδέν. Οι προδιαγραφές του προγράμματος αυτού απαιτούν ότι αυτή η πιθανότητα πρέπει να αποκλειστεί και να μην μπορεί να επηρεάσει το όλο σύστημα. Έτσι απαιτείται η χρήση κώδικα exception. Στην περίπτωση που φαίνεται στον κώδικα η πρώτη προσπάθεια φαίνεται ότι δεν πληροί τις προδιαγραφές, αφού η περίπτωση του μηδενός δεν αντιμετωπίζεται. Ο σωστός τρόπος αποφυγής του λάθους είναι με τη χρήση του κώδικα που είναι σε σχόλιο.

Για να ελέγξουμε αυτή την περίπτωση χρησιμοποιούμε μια post-condition έκφραση την: **//@ensures \result == ((d == 0) ? \old(r) : \old(r) / d) ;** η οποία τοποθετείται πριν από την μέθοδο και προδιαγράφει το αποτέλεσμα της ότι η διαχείριση της συγκεκριμένης περίπτωσης θα γίνει σωστά και το πρόγραμμα δε θα έχει σφάλματα.

4.2.15 Πρότυπο Calling by Value or Reference

Στο πρότυπο αυτό αντιμετωπίζουμε την πιθανότητα αλλαγής της τιμής μιας μεταβλητής μέσα σε κάποια μέθοδο, χωρίς όμως να μην επέλθει καμία αλλαγή εκτός της μεθόδου και η μεταβλητή να παραμένει με την τιμή που είχε πριν σταλεί στην μέθοδο για επεξεργασία.



Σχήμα 4.5 : Πρότυπο για Calling by Value or Reference

Στον πιο πάνω πηγαίο κώδικα έχουμε δυο περιπτώσεις. Η πρώτη περίπτωση όπου η μέθοδος δέχεται σαν όρισμα μια μεταβλητή τύπου int, στην οποία δίνει κάποια τιμή μέσα στην μέθοδο. Αυτή η τιμή όμως όταν τελείωσει η μέθοδος δεν θα υφίσταται αφού η μεταβλητή παρέμενε ανεπηρέαστη. Στη δεύτερη περίπτωση η μέθοδος δέχεται ως όρισμα ένα αντικείμενο τύπου integer και η αλλαγή που γίνεται στην τιμή έχει ως αποτέλεσμα να αλλάξει και η τιμή μετά το τέλος της μεθόδου.Η συγκεκριμένη post-condition έκφραση που χρησιμοποιούμε είναι η: **/*@requires \typeof(numa) <: \type(Object[]);@*/** για την πρώτη περίπτωση και η **/*@requires \typeof(numb) <: \type(Object[]);@*/** για τη δεύτερη περίπτωση, όπου το μόνο που αλλάζει είναι το όνομα της μεταβλητής που χρησιμοποιούμε κάθε φορά.

Κεφάλαιο 5

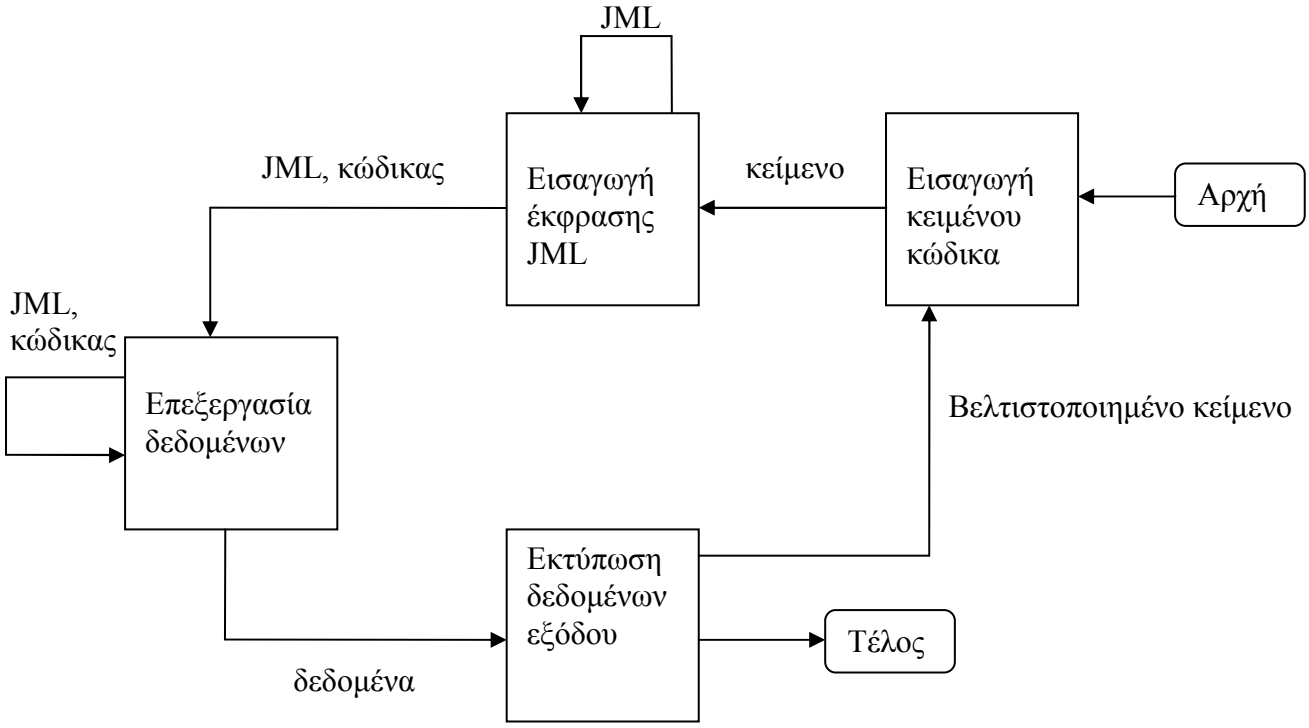
Παρουσίαση Μεθοδολογίας και Λογισμικού

5.1 Εισαγωγή

Σ' αυτήν την ενότητα θα γίνει η παρουσίαση της μεθοδολογίας που χρησιμοποιήθηκε για την αποπεράτωση αυτής της ατομικής διπλωματικής εργασίας. Πιο συγκεκριμένα, θα δούμε περισσότερες λεπτομέρειες για την μεθοδολογία υλοποίησης που ακολουθήθηκε για να γίνει το πρόγραμμα, θα αναφερθούμε στα προγράμματα τα οποία χρησιμοποιήθηκαν αλλά και θα αναλύσουμε την λειτουργικότητα του λογισμικού που υλοποιήθηκε. Θα απαρτίσουμε ξεχωριστά όλες τις λειτουργίες και για κάθε μια θα σχολιαστεί ο τρόπος με τον οποίο χρησιμοποιείται αλλά και τα αποτελέσματα που παράγονται. Επίσης θα σχολιάσουμε τα δεδομένα εξόδου από το σύστημα και με ποιο τρόπο ο χρήστης μπορεί να καταλάβει εάν το πρόγραμμα με το οποίο ασχολείται πλήροί τις προδιαγραφές που έχουν εισαχθεί στο σύστημα. Περαιτέρω θα μιλήσουμε για την ευελιξία του λογισμικού αλλά και τις διάφορες επιλογές που μπορούν να χρησιμοποιηθούν. Τελειώνοντας θα αναφερθούμε στα μειονεκτήματα αλλά και στα

πλεονεκτήματα αυτής της εφαρμογής και θα προτείνουμε τρόπους επέκτασης τόσο σε λογική όσο και σε επιλογές.

Πριν αρχίσει η περιγραφή πρέπει να πούμε ότι τα εργαλεία που χρησιμοποιήθηκαν ήταν το λογισμικό Netbeans IDE, το οποίο προσφέρει ένα ολοκληρωμένο περιβάλλον για να δημιουργηθεί η διαπροσωπία του συστήματος. Επίσης ο κύριος λόγος που χρησιμοποιήθηκε το Netbeans IDE, είναι η δυνατότητα του για δημιουργία ενός ανεξάρτητου λογισμικού. Αυτό το λογισμικό θα μπορεί να τρέχει παράλληλα με κάποιο άλλο πρόγραμμα που θα αποφασίσει ο προγραμματιστής για να συντάσσει τον κώδικα του ή και εντελώς ανεξάρτητο εάν ο προγραμματιστής επιθυμεί να εργαστεί εξολοκλήρου στην εφαρμογή αυτή. Η γλώσσα που χρησιμοποιεί και υποστηρίζει το Netbeans IDE είναι η Java, η οποία είναι η γλώσσα προγραμματισμού που επιλέξαμε μαζί με την γλώσσα προδιαγραφών συμπεριφοριστικής διαπροσωπίας (JML) για να δημιουργήσουμε το λογισμικό αυτό.



Σχήμα 5.1: Σχεδιάγραμμα Μεθοδολογίας

Πιο πάνω στο σχήμα 5.1 βλέπουμε το σχεδιάγραμμα μεθοδολογίας που εκφράζει την εφαρμογή. Όπως βλέπουμε αρχικά γίνεται η εισαγωγή του κώδικα, που όπως θα

μάθουμε σε μετέπειτα στάδιο αυτό μπορεί να γίνει με ποικίλους τρόπους. Στο επόμενο στάδιο εισάγονται και οι εκφράσεις της JML, οι οποίες εκφράζουν τις προδιαγραφές και κατ' επέκταση τους περιορισμούς που θέλουμε να τηρηθούν στο πρόγραμμα και μπορεί να τις επιλέξει ο χρήστης. Εδώ πρέπει να αναφέρουμε ότι ο κώδικας αλλά και οι JML εκφράσεις μπορούν να δοθούν ταυτόχρονα χωρίς να υπάρχει κάποιο πρόβλημα στη συνέχεια του προγράμματος.

Μόλις γίνει η εισαγωγή ολόκληρου του κειμένου, τότε τα δεδομένα μεταβαίνουν στο επόμενο στάδιο όπου γίνεται η επεξεργασία τους. Εκεί βλέπουμε ότι υπάρχει ένας βρόγχος ο οποίος θα καθορίζεται από τις μεταβλητές που θα δώσει ο χρήστης αλλά και από το κείμενο που δόθηκε για επεξεργασία. Το λογισμικό σε αυτή την φάση, θα τρέξει το όλο κείμενο από κώδικα που θα του δοθεί όσες φορές θελήσει ο χρήστης. Σε κάθε επανάληψη που θα γίνεται θα δοκιμάζονται διαφορετικές μεταβλητές κάθε φόρα, που αυτές θα καθορίζονται από μια συνάρτηση η οποία θα εξάγει από ένα πλαίσιο τιμών, τυχαίους αριθμούς. Όπως είναι λογικό το λογισμικό θα κάνει πιο πολύ ώρα να επεξεργαστεί τα δεδομένα και κατ' επέκταση να βγάλει αποτελέσματα, εάν ο χρήστης δώσει αριθμό επαναλήψεων αρκετά μεγάλο.

Στο τέλος , όταν το λογισμικό έχει έτοιμα τα αποτελέσματα, θα αρχίσει να τα τυπώνει στην οθόνη, με τη σειρά που βρέθηκαν οι εκφράσεις των προδιαγραφών στο σύστημα.
Όταν τα δεδομένα τυπωθούν στην οθόνη τότε ο χρήστης θα μπορεί να βγάλει τα δικά του συμπεράσματα ανάλογα πάντα, εάν τηρήθηκαν οι προδιαγραφές που έθεσε στο πρόγραμμα, πριν την εκτέλεση του.

Αφού έχουν βγει τα αποτελέσματα, και ο χρήστης ανέλυσε τα δεδομένα και πήρε τις αποφάσεις του, τότε μπορεί να προσθέσει ή και να αλλάξει τον ήδη υπάρχον κώδικα για να ξανατρέξει το πρόγραμμα και να πάρει νέα αποτελέσματα. Επίσης ο χρήστης έχει τη δυνατότητα να αλλάξει και τις JML εκφράσεις όπως αυτός νομίζει για να καθοδηγήσει καλύτερα και πιο αποτελεσματικά την ροή του προγράμματος.

5.2 Διαδικασία ανοίγματος του λογισμικού

Το λογισμικό ονομάζεται SoftwareTesting.jar και προσφέρεται μαζί με τον φάκελο lib ο οποίος περιέχει τις τρεις βιβλιοθήκες που χρειάζεται για να τρέξει πρόγραμμα την AbsoluteLayout, την toplink-essentials και την toplink-essentials-agent. Όπως βλέπουμε ο τύπος του λογισμικού είναι Jar , που είναι το εκτελέσιμο αρχείο που μπορεί να προσφέρει το Netbeans. Έχοντας εγκαταστήσει το πρόγραμμα της java στον υπολογιστή μας το λογισμικό αυτό μπορεί απλά να τρέχει χωρίς να επέμβουμε κάπου εμείς, απλά κάνοντας διπλό κλικ στο αρχείο αυτό.

Εάν το λογισμικό δεν ανοίγει αυτό σημάνει ότι υπάρχουν κάποια registries που δεν αναγνωρίζουν σωστά τα .jar αρχεία. Ο τρόπος αντιμετωπίσεως παρέχεται στο άλλο αρχείο που έρχεται με το λογισμικό και ονομάζεται fix_jar_associations.reg. Το συγκεκριμένο αρχείο τροποποιεί τα registries του υπολογιστή έτσι να αναγνωρίζει σωστά τα αρχεία που μας ενδιαφέρουν. Κάνουμε διπλό κλικ στο αρχείο αυτό και έπειτα μπορούμε να τρέξουμε το λογισμικό μας κανονικά χωρίς να πρέπει να ξανακάνουμε τη διαδικασία αυτή. Ο κώδικας που χρησιμοποιήσαμε για αυτή την περίπτωση παρουσιάζεται πιο κάτω:

```
[HKEY_CLASSES_ROOT\.jar]
@="jarfile"
[HKEY_CLASSES_ROOT\jarfile]
@="Executable Jar File"
[HKEY_CLASSES_ROOT\jarfile\shell]
[HKEY_CLASSES_ROOT\jarfile\shell\open]
[HKEY_CLASSES_ROOT\jarfile\shell\open\command]
@="\"C:\\Program Files (x86)\\Java\\jre6\\bin\\javaw.exe\" -jar \"%1\" %*"
```

Εάν για κάποιο λόγο βρισκόμαστε σε υπολογιστή, στον οποίον δεν έχουμε πρόσβαση για να αλλάξουμε τα registries, τότε θα πρέπει να χρησιμοποιήσουμε το κάποιο κέλυφος γραμμής εντολών έτσι ώστε να το τρέξουμε από εκεί. Συγκεκριμένα για να τρέξουμε το λογισμικό πρέπει να ανοίξουμε ένα κέλυφος, να πάμε στον φάκελο που βρίσκεται το λογισμικό μας και να γράψουμε την πιο κάτω εντολή:

```
java -jar SoftwareTesting.jar
```

5.3 Περιγραφή της εφαρμογής

Για την υλοποίηση αυτής της διπλωματικής εργασίας, χρειάστηκε να δημιουργηθεί ένα λογισμικό το οποίο να μπορεί να κάνει δυναμικό έλεγχο στον πηγαίο κώδικα ενός λογισμικού, με την χρήση της JML. Ο έλεγχος αυτός θα πρέπει να δίνει αποτελέσματα στον χρήστη έτσι ώστε να μπορεί ο ελεγκτής να συμπεράνει εύκολα εάν το σύστημα του πληροί τις απαιτήσεις που δόθηκαν ή όχι.

Σε γενικά πλαίσια το λογισμικό χωρίζεται σε τέσσερα μέρη. Πρώτα την επιλογή και την εισαγωγή στην οθόνη ενός προτύπου, από εκείνα που δόθηκαν στις προηγούμενες μελέτες, αλλά και από τα καινούργια που δημιουργήθηκαν για να ενισχύσουν την υλοποίηση αυτής της μελέτης και να δώσουν πιο πολλές επιλογές στον χρήστη. Στα πρότυπα αυτά συμπεριλαμβάνεται τόσο ο κώδικας όσο και οι δηλώσεις της JML, οι οποίες δίνονται ξεκάθαρα και μπορεί ο χρήστης εύκολα να κατανοήσει πού αποσκοπούν οι δηλώσεις αυτές. Πρέπει να πούμε ότι αυτά τα πρότυπα είναι με τέτοιον τρόπο δοσμένα έτσι ώστε ο χρήστης να μπορεί να κάνει πάνω σε αυτά αλλαγές και να τα προσαρμόσει στα δικά του πλαίσια, σύμφωνα πάντα με την καθοδήγηση του από το πρόγραμμα. Επίσης αυτά τα πρότυπα δίνονται σαν έτοιμα παραδείγματα έτσι ώστε να γίνει πιο κατανοητή η λειτουργικότητα του προγράμματος, αφού υπάρχει η δυνατότητα να τρέξεις τα πρότυπα και να πάρεις τα αποτελέσματα στην οθόνη.

Η δεύτερη επιλογή που μπορεί να κάνει ο χρήστης είναι να επιλέξει μέσα από αρκετές ατελείς δηλώσεις JML, οι οποίες βρίσκονται σε μια λίστα πάνω δεξιά στο λογισμικό. Εφόσον ο χρήστης επιλέξει μια από αυτές τις δηλώσεις τότε θα εμφανιστούν οι δηλώσεις αυτές σε δυο θέσεις στην οθόνη. Η πρώτη δήλωση που υπάρχει στην περιοχή του κειμένου με τα παραδείγματα, θα τυπώσει ένα πλήρης παράδειγμα της έκφρασης που έχει δηλώσει ο χρήστης, έτοιμη να εκτελεστεί. Η δεύτερη δήλωση θα εμφανιστεί, ακριβώς στην γραμμή του κώδικα που έχει τον δείκτη του ποντικιού ο χρήστης πριν επιλέξει μια έκφραση από την λίστα. Ο λόγος που γίνεται αυτό είναι για να μπορεί ο χρήστης να βάζει σε όποιο σημείο στον κώδικα επιθυμήσει, τις δηλώσεις JML που του χρειάζονται. Το πρόγραμμα παρέχει από μόνο του μια καθοδήγηση για τον τρόπο που θα πρέπει να δηλώνονται οι JML εκφράσεις. Εκτός από το παράδειγμα που θα δίνεται, η δεύτερη έκφραση θα εμφανίζεται ημιτελής, όπου ο χρήστης θα πρέπει να

συμπληρώνει τα σημεία που δίνονται με τετράγωνες αγκύλες. Ο λόγος που γίνεται αυτό είναι για να μπορεί το πρόγραμμα να αναγνωρίζει μέσα από την πληθώρα των εντολών και μέσα από τους τόσους πολλούς τρόπους που μπορείς να γράψεις μια έκφραση, να μπορεί να την αναγνωρίζει.

Η τρίτη επιλογή του προγράμματος είναι και πάλη μια επιλογή από τα πρότυπα που αναφέραμε προηγουμένως, αλλά αυτή τη φορά θα είναι μόνο ο κώδικας στη Java, χωρίς να είναι ενσωματωμένες οι δηλώσεις της JML. Παρόλα αυτά υπάρχει η δυνατότητα εισαγωγής των δηλώσεων αυτών από την λίστα με τις δηλώσεις της JML. Με αυτό τον τρόπο ο χρήστης θα έχει τα πρότυπα που είναι οι γενικές περιπτώσεις που εισηγήθηκαν, και θα μπορεί και αυτός με την σειρά του να εισάγει όποιο JML statement επιθυμήσει, για να ελέγξει ότι χρειάζεται, βλέποντας τις απαντήσεις στην οθόνη.

Η τέταρτη και τελευταία επιλογή βρίσκεται πάνω αριστερά στην οθόνη, όπου υπάρχει η δυνατότητα εισαγωγής στο λογισμικό, οποιοδήποτε αρχείο με κώδικα που βρίσκεται φυλαγμένο στον υπολογιστή. Έτσι με αυτό τον τρόπο γίνεται εφικτή η αλληλεπίδραση του λογισμικού με το γύρω περιβάλλον. Παράλληλα ο χρήστης μπορεί και πάλι να εισάγει στο πρόγραμμα τα JML statement, που αναφέραμε και προηγουμένως.

Εκτός από τις επιλογές που αναφέραμε, στο λογισμικό υπάρχουν διάφοροι χώροι κειμένου στους οποίους αναγράφονται διαφορετικά δεδομένα. Αρχικά υπάρχει ο χώρος όπου εισάγεται ο κώδικας μαζί με τα JML statements, από πάνω του υπάρχει ο χώρος στον οποίο φαίνονται τα παραδείγματα των JML. Δίπλα τους είναι τοποθετημένοι οι χώροι στους οποίους εμφανίζονται τα τελικά αποτελέσματα. Αυτοί οι χώροι είναι ανακατανεμημένοι σε δυο στήλες. Την στήλη με τις δηλώσεις των JML για τις οποίες βρέθηκε ότι πληρούν τις προδιαγραφές και η στήλη η οποία αναγράφεται το αντίθετο. Κάθε στήλη χωρίζεται στα δυο, υπάρχει ο χώρος πάνω στον οποίο μπορούμε να δούμε τις κλάσεις ισοδυναμίας τις οποίες θα τις εξηγήσουμε πιο μετά, για κάθε έκφραση και ακριβώς από κάτω εμφανίζονται τα αποτελέσματα μαζί με τα δεδομένα εισόδου αλλά και τα δεδομένα εξόδου του προγράμματος.

Το επόμενο που θα αναφέρουμε είναι οι επιλογές που υπάρχουν είναι αρχικά η επιλογή των επαναλήψεων που θέλουμε να τρέξει το πρόγραμμα, με τα στοιχεία που θα το

δώσουμε, περιμένοντας ακριβώς τόσα δεδομένα εξόδου όσα και οι επαναλήψεις, με εξαίρεση στην περίπτωση στην οποία τα αποτελέσματα δε μπορούν να είναι πέραν του ενός. Σε αυτή την περίπτωση θα πάρουμε μόνο ένα αποτέλεσμα. Επίσης υπάρχει η επιλογή του διαστήματος τιμών που θέλουμε να χρησιμοποιηθεί στο σύστημα, όπου και εδώ θα το αναλύσουμε πως λειτουργεί σε περαιτέρω ενότητα.

Σημαντικό λειτουργικό σημείο της εφαρμογής είναι τα δυο κουμπιά που διαφαίνονται στο κάτω μέρος. Εκεί υπάρχουν τα λειτουργικά κουμπιά Run και Clear All με τα οποία ο χρήστης αλληλεπιδρά άμεσα. Το κουμπί Run είναι αυτό που θα πιέζεται και θα εκκινεί την διαδικασία του ελέγχου. Θα διαβάζει όλα τα δεδομένα που έχει δώσει ο χρήστης, το κείμενο με τον κώδικα, τα JML statements, το εύρος των τιμών αλλά και τον αριθμό των επαναλήψεων. Θα τα επεξεργάζεται και θα εμφανίζει στις διάφορες οθόνες τα αποτελέσματα που θα εξαχθούν. Το δεύτερο κουμπί που υπάρχει είναι το Clear All, το οποίο καθαρίζει όλες τις οθόνες από τα δεδομένα που υπάρχουν και δίνει την ευχέρεια στον χρήστη να εφαρμόσει την διαδικασία από την αρχή. Εδώ πρέπει να αναφέρουμε ότι όλες οι οθόνες είναι επεξεργάσιμες ανά πάσα στιγμή, και ο χρήστης μπορεί να αλλάξει ότι νομίζει δεν είναι σωστό. Παρόλα αυτά η μόνη οθόνη που θα επηρεάσει το σύστημα εάν κάνει αλλαγή θα είναι αυτή στο σημείο που εισάγεται ο κώδικας.

Τελειώνοντας, για να βγει έξω από το λογισμικό, ο χρήστης επιλέγει το Open και έπειτα το Exit. Διαφορετικά μπορεί αν πιέσει το κουμπί «X» που υπάρχει σε όλα τα παράθυρα.

5.4 Περιγραφή υλοποίησης

Για να δημιουργηθεί αυτή η εφαρμογή, έπρεπε να βρεθεί κάποιος τρόπος με τον οποίο να μπορούμε να περνούμε τα αποτελέσματα του εκτελέσιμου κομματιού του κώδικα γραμμένου σε Java, και παράλληλα να μεταγλωττίζουμε τα κομμάτια του κώδικα της JML για να κατανοούμε την λογική τους. Αν και υπάρχουν πολλά προγράμματα τα οποία μπορούν να μεταγλωττίζουν τα JML statements, εντούτοις δεν υπήρχε τρόπος να παίρνουμε τις πληροφορίες που θέλαμε έτσι ώστε να μπορέσουμε να τα επεξεργαστούμε και να τα αναλύσουμε όπως έπρεπε. Επίσης το ίδιο συνέβηκε και στην έρευνα για κάποιο πρόγραμμα το οποίο να μπορούσε να μεταγλωττίσει κομμάτια Java αρχείου και παράλληλα να μας επιτρέπει να δίνουμε από έξω τιμές στις μεταβλητές που θα επιλέγαμε, αλλά και να μας επιστρέφει τις τιμές των μεταβλητών που χρειαζόμασταν για να εξάγουμε τα αποτελέσματα.

Εν τέλει ο τρόπος που αποφασίστηκε και ακολουθήθηκε ήταν η δημιουργία Parser, τόσο για τον κώδικα γραμμένο σε Java, όσο και για τις εκφράσεις της JML. Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι ο Parser για την JML φτιάχτηκε για να αναγνωρίζει ξεχωριστές εκφράσεις που θα χρησιμοποιούνται σαν περιορισμοί για το λογισμικό το οποίο θα θέλουμε να ελέγξουμε. Ο Parser της Java, είναι κατά κύριο λόγο υποσύνολο αυτού της JML, αφού αρχικά το πρόγραμμα ανιχνεύει πρώτα τις εκφράσεις τις JML, και αν κριθεί απαραίτητο ελέγχει και τον κώδικα της Java. Συγκεκριμένα το λογισμικό θα διαβάσει όλες τις εκφράσεις JML, και εάν η έκφραση είναι τύπου requires, τότε ο Parser της Java θα επικεντρωθεί στις μεταβλητές που αναγράφονται τόσο στη JML όσο και στον κώδικα της Java. Στις περιπτώσεις που το JML statement είναι τύπου ensures, που αυτό σημαίνει ότι το αποτέλεσμα που θα ελεγχθεί θα παραχθεί εξολοκλήρου από τον κώδικα της μεθόδου που ακολουθεί, τότε θα ενεργοποιηθεί ο Parser της Java θα διαβάσει τον κώδικα και θα τον τροποποιήσει με τρόπο έτσι ώστε να τον καταλαβαίνει για να μπορεί να παράγει τα απαραίτητα δεδομένα.

Ο κύριος τρόπος που με τον οποίο δουλεύουν οι Parsers που ανέφερα προηγουμένως είναι με τη χρήση κανονικών εκφράσεων. Οι κανονικές εκφράσεις βοηθούν στην αναγνώριση συγκεκριμένων εκφράσεων που θα ορίσουμε. Όλη η αναγνώριση των JML statements έγινε με κανονικές εκφράσεις που εισάχθηκαν στην αρχή του

προγράμματος. Για κάθε JML statement, και κατ' επέκταση κάθε κανονική έκφραση υπάρχει συγκεκριμένη λογική που τρέχει κατά την αναγνώριση των εκφράσεων. Επίσης μπορούμε να αναφέρουμε ότι χρησιμοποιήθηκαν κανονικές εκφράσεις μέσα σε κανονικές εκφράσεις για να αναγνωρίζουμε τον κώδικα της Java που θα αναγνωριστεί μετά από τα JML.

```
Pattern start = Pattern.compile("(//@ *|/\\*@ *)");
Pattern end = Pattern.compile(";");
Pattern white = Pattern.compile("[\\s]*");
Pattern anisotites = Pattern.compile("(!=|==|>=|<=|<|>)");
Pattern variable = Pattern.compile("[a-zA-Z0-9]+");
Pattern digits = Pattern.compile("([\\d]+)");
Pattern checkvar = Pattern.compile(white + "" + start +"requires"+ white +variable2 + white
+ anisotites + white + digits + white + end);
```

Πιο πάνω βλέπουμε ένα παράδειγμα μίας κανονικής έκφρασης που χρησιμοποιεί άλλες κανονικές εκφράσεις για να αναγνωρίσει το ζητούμενο κώδικα. Η κανονική έκφραση checkvar, που φαίνεται πιο πάνω αναγνωρίζει εκφράσεις του τύπου *//@requires a!=0;* Όπου η αρχή μπορεί να γραφτεί είτε //@ είτε /*@ , που είναι οι τρόποι οι οποίοι αρχίζουν τα JML statements, στη θέση του a μπορεί να μπει οποιαδήποτε μεταβλητή, επίσης μπορούμε να βάλουμε οποιαδήποτε ανίσωση, αλλά και οποιονδήποτε αριθμό στο τέλος. Επίσης όπως φαίνεται και στον κώδικα πιο πάνω στην κανονική έκφραση αναγνωρίζει όλα τα white spaces που βρίσκει.

Με βάση τη λειτουργία της κάθε κανονικής έκφρασης, που για κάθε JML statement υπάρχει τουλάχιστον μια, υπάρχει και δική του μέθοδος στον κώδικα που τυπώνει ξεχωριστά τα δεδομένα για την συγκειμένη προδιαγραφή στον κώδικα που εισήγαμε. Δηλαδή κάθε JML έκφραση έχει την δική της επεξεργασία για να τυπώσει τα αποτελέσματα ξεχωριστά από τις άλλες.

Υπάρχουν όμως διάφοροι περιορισμοί, μέσα από τις προδιαγραφές που θα οριστούν, να χρησιμοποιούν τις ίδιες μεταβλητές, και η ροή των μεταβλητών στο σύστημα να είναι σημαντική για όλες τις εκφράσεις της JML. Σε αυτές τις περιπτώσεις για κάθε JML statement υπάρχουν δυο σημεία στον κώδικα. Το πρώτο που ορίζει ότι το JML

statement αυτό είναι αυτόνομο και δεν παίρνει τιμές από κανένα άλλο σημείο στον κώδικα και το δεύτερο σημείο που είναι υπεύθυνο να διαμοιράζει τις τιμές σε όλες τις εκφράσεις του συστήματος. Την πρώτη περίπτωση την συναντάμε στην εισαγωγή ανεξάρτητων JML statements, όπου ο χρήστης έχει την ευχέρεια να ελέγξει την συγκεκριμένη έκφραση χωρίς να βάλει οποιοδήποτε κώδικα γραμμένο σε Java. Την δεύτερη περίπτωση την συναντάμε στο σημείο με τα έτοιμα Templates που εμπεριέχουν και JML statements, και οι εκφράσεις μεταξύ τους, τις πιο πολλές φορές αλληλεπιδρούν και διαμοιράζονται τις τιμές των μεταβλητών για να τρέξει το πρόγραμμα με τα σωστά δεδομένα.

Εκτός από τον παράγοντα της αναγνωρίσεις του κώδικα υπάρχει και ο τομέας τις επεξεργασίας. Όλα τα δεδομένα που αντλούνται από το λογισμικό μπαίνουν σε hash table, στο οποίο μπαίνει ως κλειδί το όνομα της μεταβλητής και ως δεδομένο η τιμή της. Σε περίπτωση που υπάρχουν πολλά δεδομένα τότε χρησιμοποιούνται λίστες στις οποίες φυλάσσονται όλα τα δεδομένα, όσες φορές προσδιορίσει ο χρήστης. Αυτές οι λίστες στέλλονται στο τέλος για επεξεργασία και εξαγωγή στις μεθόδους που είναι υπεύθυνες για να τυπώσουν τα αποτελέσματα.

Μια άλλη σημαντική λειτουργία του λογισμικού είναι ο τρόπος με τον οποίο δημιουργεί τα δεδομένα τα οποία τα δίνει στις μεταβλητές του προγράμματος. Αρχικά η πρώτη σκέψη ήταν να γίνει η τροφοδοσία του συστήματος, βάσει γενετικών αλγορίθμων, όπου θα ήταν υπεύθυνοι να αναλύουν το πρόγραμμα και να δίνουν στις μεταβλητές τιμές οι οποίες εξήχθηκαν με βάση τη λογική του γενετικού αλγόριθμου. Η σκέψη αυτή δεν έγινε στην πράξη αφού ήταν δύσκολη η ενσωμάτωση γενετικού αλγόριθμου στο πρόγραμμα. Εντούτοις δημιουργήσαμε μια λογική η οποία δημιουργεί τα δεδομένα αυτά.

Στο λογισμικό υπάρχει η επιλογή range, αυτή η τιμή είναι υπεύθυνη για να καθορίζει τον τρόπο με τον οποίο θα αναπαράγονται τα δεδομένα στο σύστημα. Η πρώτη λειτουργία της μεταβλητής αυτής που καθορίζεται από τον χρήστη είναι, στην περίπτωση όπου η μεταβλητές που θέλουμε να ασχοληθούμε είναι σταθερός αριθμός. Στην περίπτωση αυτή ο χρήστης θα δώσει ένα αριθμό ο οποίος θα διαιρεθεί με το δυο, και θα παίρνει τιμές από το αρνητικό του αριθμό μέχρι και τον θετικό του. Για

παράδειγμα εάν δώσουμε την τιμή 10, τότε το οι αριθμοί που θα παίρνει το πρόγραμμα θα είναι τυχαίοι αριθμοί μεταξύ από -5 μέχρι 5. Στην περίπτωση όπου έχουμε κάποιο χειριστήριο το οποίο θα πρέπει να πάρει τιμές εκ των οποίων η μια θα είναι null. Τότε το πρόγραμμα θα δίνει αριθμούς εκ των οποίων το 10% του δοθέντος αριθμού θα είναι null. Στην περίπτωση όπου έχουμε String τότε το πρόγραμμα θα παίρνει από ένα τυχαίο αριθμό από το μηδέν μέχρι τον αριθμό που του έχουμε δώσει και θα τον μετατρέπει σε χαρακτήρα με βάση το ASCI, με την εξαίρεση ότι και πάλι το 10% των τιμών θα είναι Null, για να ικανοποιήσουμε όλες τις περιπτώσεις.

Εκτός από τις τυχαίες μεταβλητές που δημιουργούνται στο σύστημα με βάση κάποια όρια που καθορίζονται από τον χρήστη, το λογισμικό έχει την δυνατότητα να δημιουργεί κλάσεις ισοδυναμίας για κάθε περίπτωση. Δηλαδή αναλύει τα JML statements ξεχωριστά και βρίσκει τις περιπτώσεις στις οποίες το JML statement είναι ψευδές άρα το σύστημα δεν τηρεί τις προδιαγραφές και τις περιπτώσεις στις οποίες επαληθεύεται. Τις κλάσεις αυτές τις τυπώνει, πάνω από τις δοκιμές των τιμών, έτσι ώστε ο χρήστης γνωρίζει από πριν ποιά δεδομένα θα επαληθευτούν και ποια δεδομένα θα αποτοιχίσουν. Ένα παράδειγμα από κλάσεις ισοδυναμίας είναι το ακόλουθο:

Statement:
/*@requires 1 <= n;
Case 1: 1 <= n - PASS
Case 2: 1 > n - FAIL

Statement:
//@requires top < elems.length;
Case 1: top == elems.length – FAIL
Case 2:top < elems.length - PASS
Case 3:top > elems.length - FAIL

Ο τρόπος με τον οποίο οι κλάσεις ισοδυναμίας αναπαράγονται, έχει άμεση σχέση με τον τρόπο που δουλεύουν οι κανονικές εκφράσεις. Μόλις ένα JML statement αναγνωρισθεί από τον Parser, τότε επεξεργάζεται, βρίσκει σε ποια σημεία υπάρχουν αριθμοί, ισοδυναμίες, αλλά και μεταβλητές και παράγει τα συγκεκριμένα

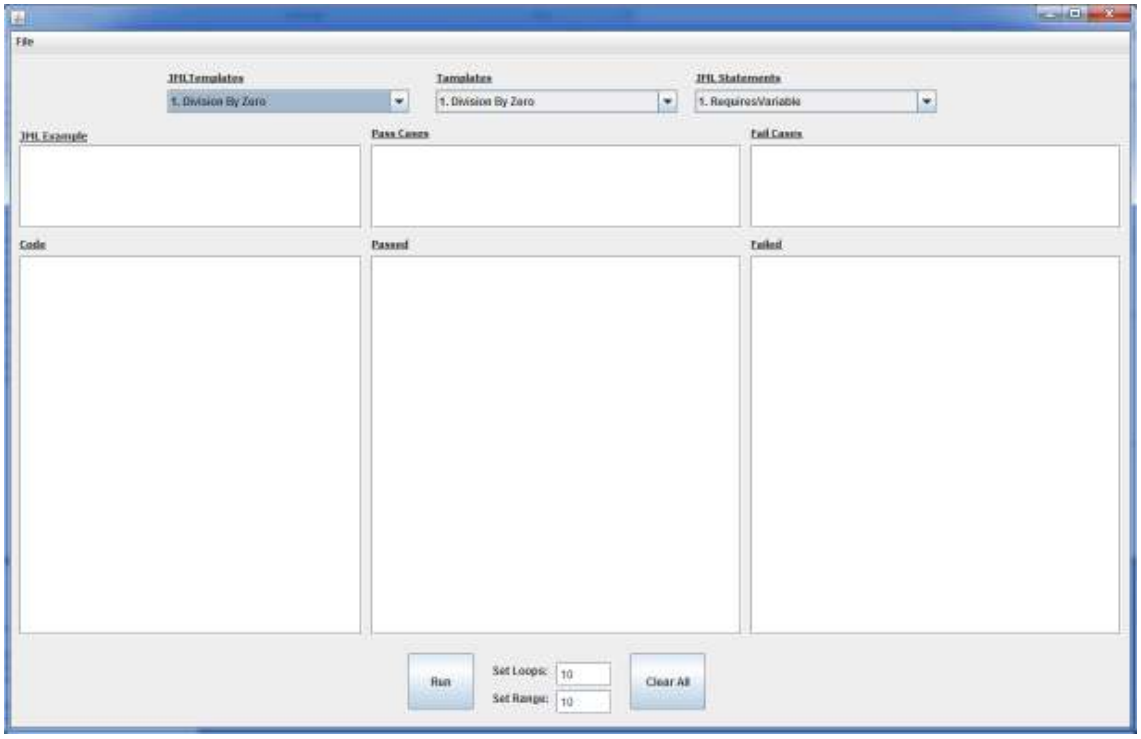
αποτελέσματα. Στο τέλος τα στέλνει στη υπεύθυνη συνάρτηση για να τυπωθούν. Επίσης πρέπει να αναφέρουμε ότι οι κλάσεις ισοδυναμίας που βρίσκονται καλύπτουν όλες τις πιθανές περιπτώσεις που μπορούν να υπάρξουν, ακόμη και αν ο τυχαίος τρόπος που αναπαράγονται οι μεταβλητές δε δώσει κάποια τιμή σε μια συγκεκριμένη κλάση. Ο λόγος που κρατήθηκε ο τυχαίος τρόπος αναπαραγωγής στοιχείων και μετά τη δημιουργία των κλάσεων ισοδυναμίας είναι για να υπάρχει ποικιλία από τιμές τις οποίες θα ορίζει με τον τρόπο που αναφέραμε ο χρήστης.

Μια άλλη δυνατότητα του λογισμικού είναι η ευχέρεια για ευελιξία ως προς τον τρόπο που αναγνωρίζει διάφορα μέρη στον κώδικα που διεξάγουν σημαντικό ρόλο. Μερικά από αυτά είναι οι μεταβλητές, οι οποίες αναγνωρίζονται σε κάθε σημείο στον κώδικα και μπορεί ο χρήστης να δώσει όποιο όνομα μεταβλητής επιθυμεί. Επίσης σε διάφορες εκφράσεις σημαντικό ρόλο έχουν οι ανισότητες, οι οποίες τις περισσότερες φορές κρίνουν το τελικό αποτέλεσμα, τόσο των κλάσεων ισοδυναμίας αλλά και το αποτέλεσμα του προγράμματος. Αναγνωρίζονται επίσης οι περισσότερες δεσμευμένες λέξεις, τόσο στη Java όσο και στην JML, έτσι ώστε, να μπορεί ο Parser να αναγνωρίζει τη διαφορά μεταξύ μεταβλητής και δεσμευμένης λέξης. Κανονικές εκφράσεις δημιουργήθηκαν για εκφράσεις όπως του while, if else, for κ.α.

Τελειώνοντας πρέπει να αναφέρουμε ότι παρά την δυνατότητα για αναγνώριση πολλών εκφράσεων εντούτοις δεν ήταν εφικτή η αναγνώριση σε όλες τις εκφράσεις που υπάρχουν έτσι το πρόγραμμα περιορίστηκε στην αναγνώριση των πιο βασικών περιπτώσεων.

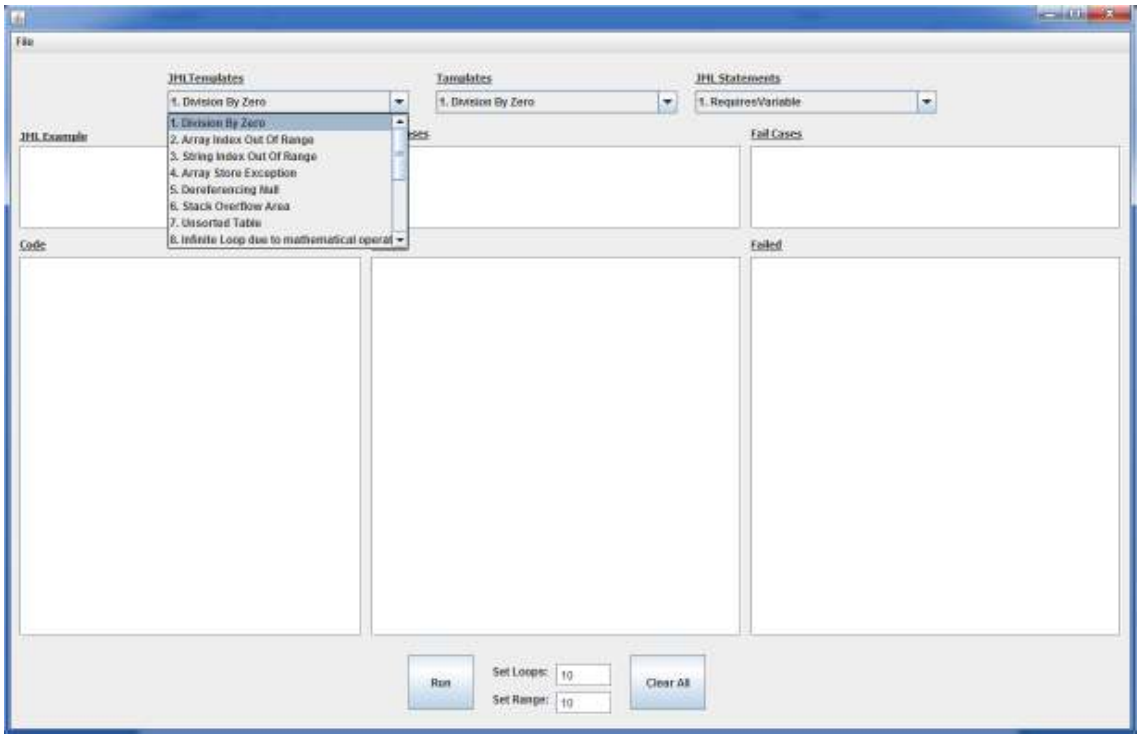
5.5 Περιγραφή Λογισμικού

Σε αυτό το σημείο θα αρχίσουμε την περιγραφή του λογισμικού, δείχνοντας εικόνες και εξηγώντας με λεπτομέρεια τη γίνεται στο σύστημα. Όπως έχουμε αναφέρει μέχρι τώρα, για να βάλουμε σε εφαρμογή το λογισμικό πρέπει να τρέξουμε το αρχείο SoftwareTesting.jar. Μόλις ενεργοποιηθεί το λογισμικό ανοίγει στην οθόνη μας η διαπροσωπία που φαίνεται στο σχήμα 5.2.

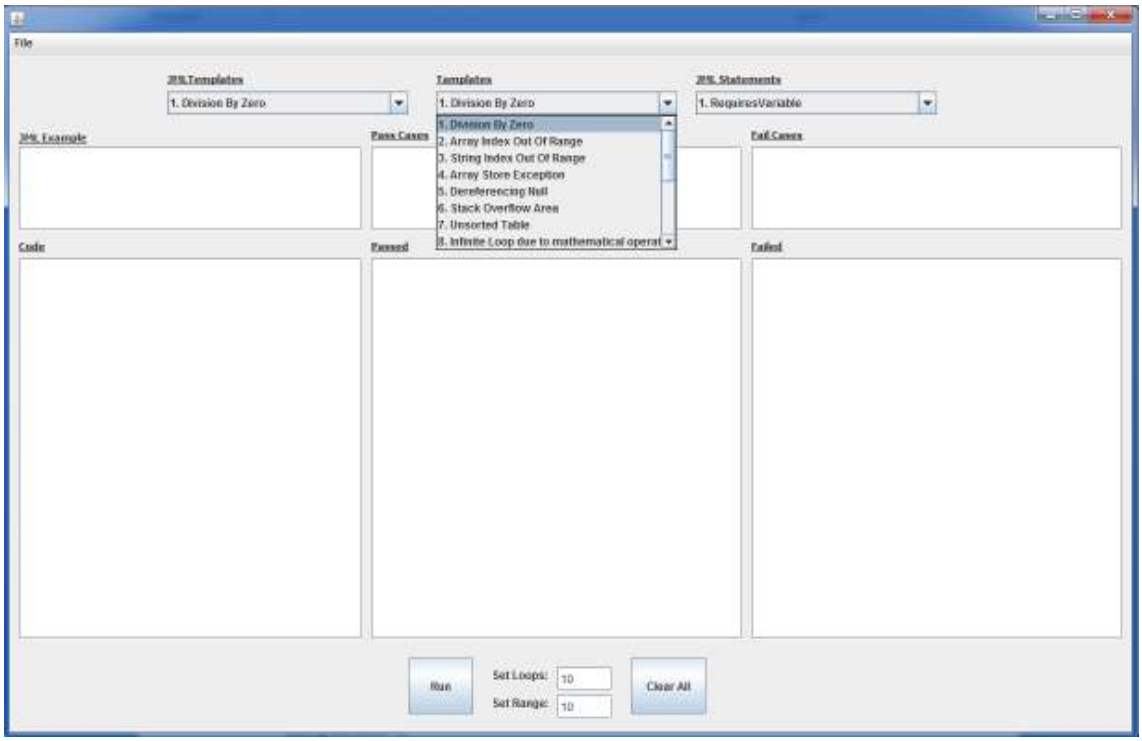


Εικόνα 5.2: Διαπροσωπία λογισμικού SoftwareTesting

Όπως φαίνεται στην εικόνα υπάρχουν τρεις λίστες στο πάνω μέρος. Η πρώτη λίστα ονομάζεται JML Templates, και περιέχει όλα τα Templates που φτιάχτηκαν και αντιπροσωπεύουν ένα μεγάλο ποσοστό των περιπτώσεων κοινών λαθών που γίνονται, με JML εκφράσεις, για να ελέγξουν κατά πόσο οι προδιαγραφές πληρούνται. Η δεύτερη λίστα περιέχει τα Templates, χωρίς τις εκφράσεις JML και οι τρίτη και τελευταία λίστα περιέχει ανεξάρτητες εκφράσεις JML τις οποίες μπορούμε ανά πάσα στιγμή να προσθέσουμε σε όποιο σημείο του κώδικα επιθυμούμε. Οι τρεις λίστες φαίνονται πιο κάτω.



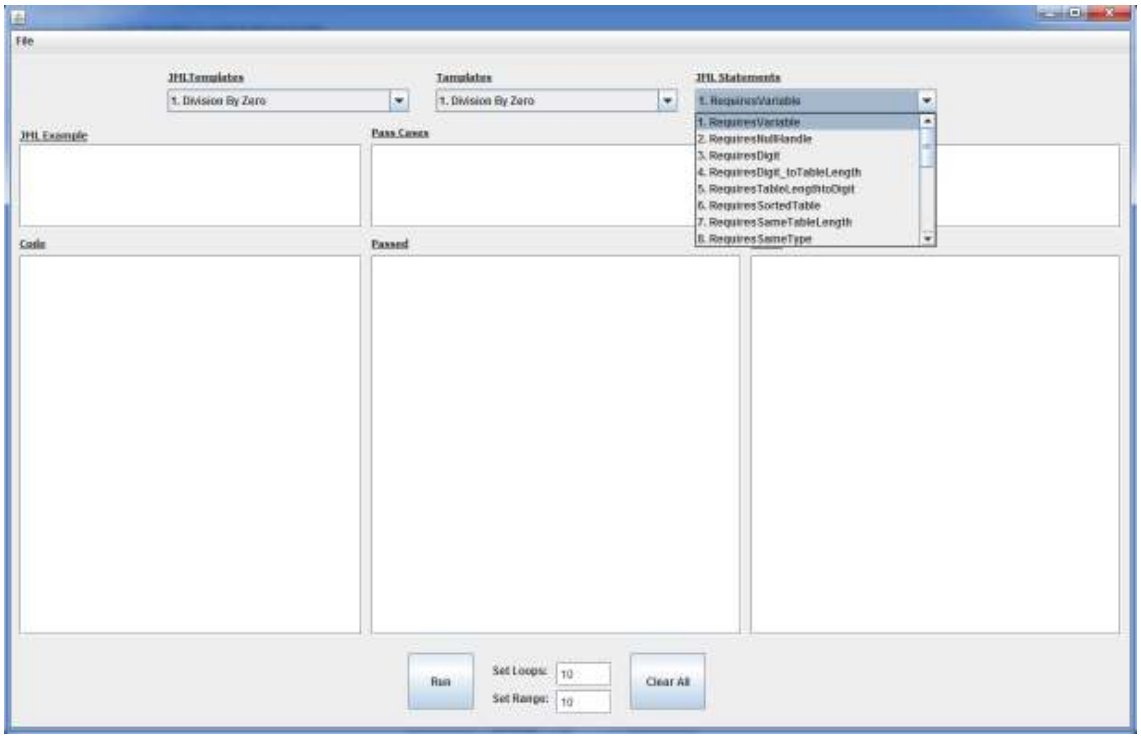
Εικόνα 5.3: Λίστα JML Templates



Εικόνα 5.4 Λίστα Java Templates

Στις εικόνες 5.3 και 5.4 φαίνονται οι λίστες με τα Templates που θα μπορούν να χρησιμοποιηθούν. Στην πρώτη εικόνα υπάρχουν ενσωματωμένα και οι εκφράσεις τις JML. Τα Templates που χρησιμοποιούνται είναι τα εξής:

1. Division By Zero, 2. Array Index Out Of Range, 3. String Index Out Of Range,
4. Array Store Exception, 5. Dereferencing Null, 6. Stack Overflow Area,
7. Unsorted Table, 8. Infinite Loop due to mathematical operator,
9. Error due to Logical operator, 10. Negative Array Size, 11. SizeOfParallelTables,
12. DiffTypeEquationException, 13. StaticInitialisation, 14. CatchingException
και 15. CallbyValueOrReference



Εικόνα 5.5: Λίστα JML Statements

Στην εικόνα 5.5 εμφανίζεται η λίστα με τα JML Statements τα οποία είναι τα πιο κάτω:

1. RequiresVariable, 2. RequiresNullHandle, 3. RequiresDigit
4. RequiresDigit_toTableLength, 5. RequiresTableLengthtoDigit,
6. RequiresSortedTable, 7. RequiresSameTableLength, 8. RequiresSameType
9. RequiresCallingbyObject_Reference, 10. AssertCounter_insideTableBoundaries
11. AssertWhile_Counters, 12. VariableTypes_insideTableTypes,

13. CheckingCounter_TobeInsideTableSize,

14 Invariant_CheckTableLength_CheckType, , 15. Factorial

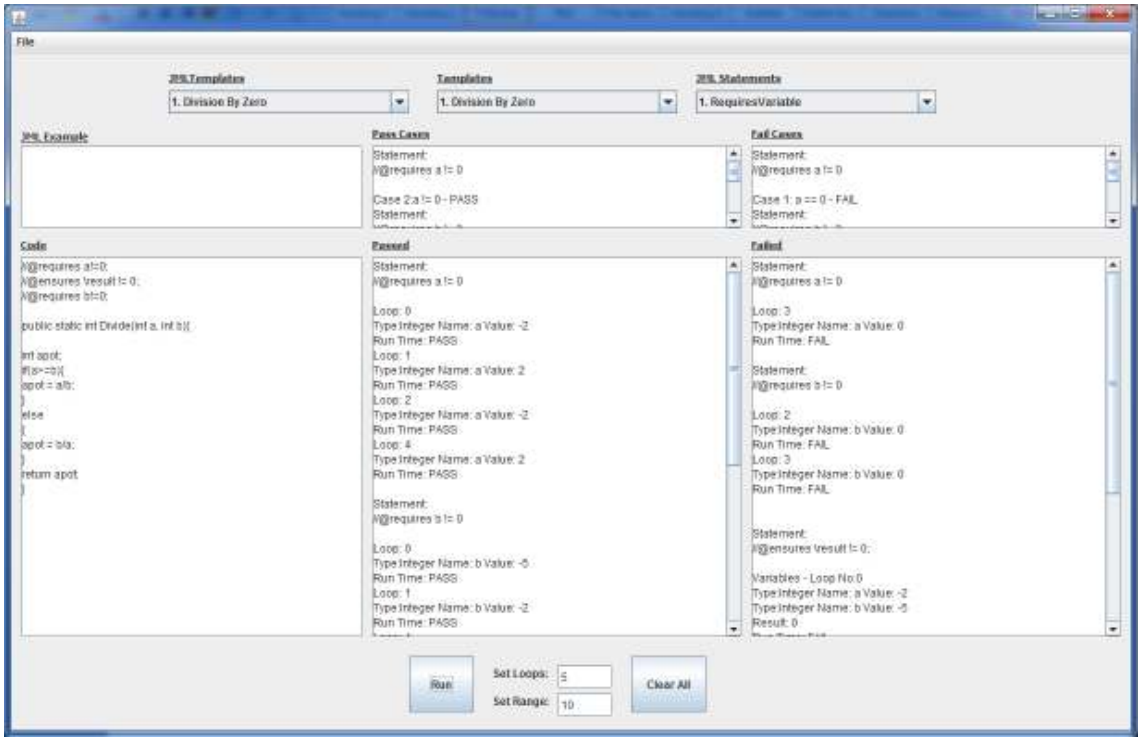
16. EnsuresTableHasValues, 17. EnsuresFindMinValuesFromTable

18. EnsuresResultNotZero, 19. EnsuresInitializeTable & 20. EnsuresExceptions

5.5.1 Περιγραφή των JML Templates

Πιο κάτω θα αρχίσουμε να αναλύουμε πιο αναλυτικά τις περιπτώσεις που συναντούμε στο πρόγραμμα, δίνοντας για λόγους παρουσίασης Loops:5 και Range:10. Επίσης να αναφέρουμε ξανά ότι όλα τα δεδομένα εισόδου αλλά και εξόδου θα εμφανίζονται στην οθόνη. Αριστερά όλα τα δεδομένα που είναι επιτυχής και πληρούν τις προδιαγραφές που δόθηκαν και δεξιά, όλα όσα απέτυχαν. Επίσης στο πάνω μέρος κάθε περιπτώσεις θα διαγράφονται αναλυτικά όλες οι κλάσεις ισοδυναμίας για κάθε έκφραση.

Αρχικά έχουμε επιλέξει το πρώτο Template Division By Zero, το οποίο και εμφανίζεται στο χώρο κειμένου με τίτλο Code. Στη συνέχεια πιέζουμε το κουμπί Run και βλέπουμε ότι το πρόγραμμα αναλύει το πρόγραμμα και εξάγει τα αποτελέσματα στις οθόνες που βλέπουμε.

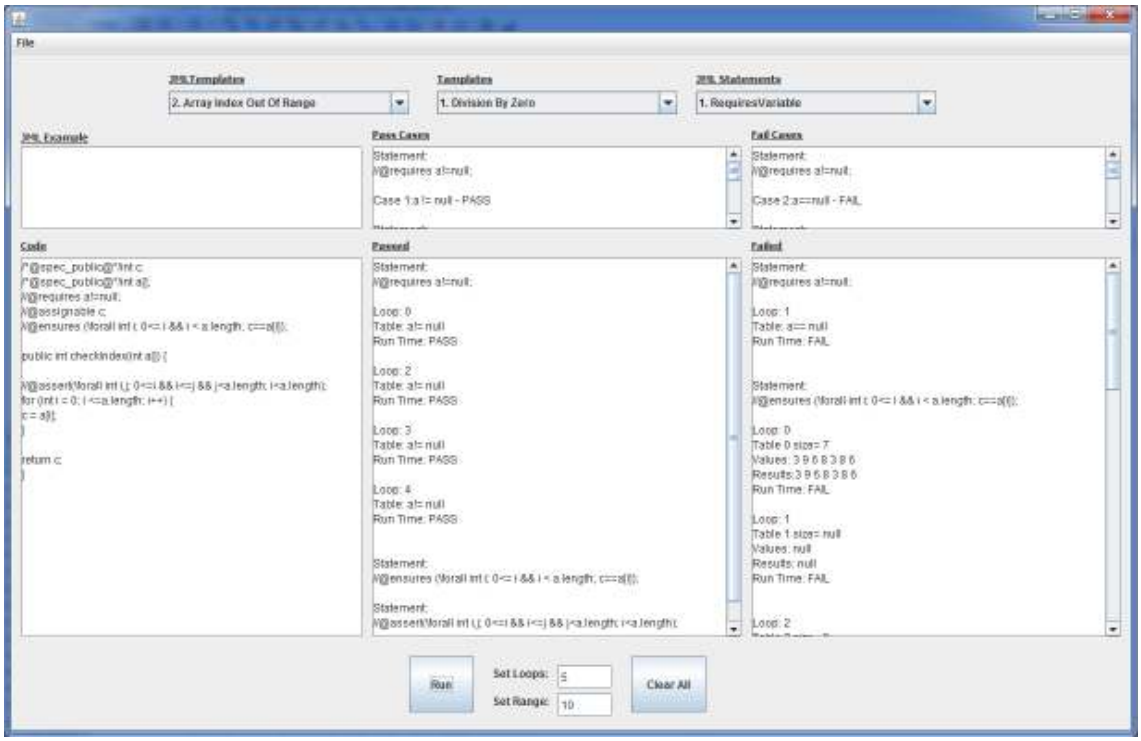


Εικόνα 5.6: Επιλογή 1ου Template και Ενεργοποίηση του Λογισμικού

Καταρχάς βλέπουμε ότι υπάρχουν 3 JML Statements, τα οποία παίρνουν τιμές από το πρόγραμμα. Οι κλάσεις ισοδυναμίας που εμφανίζονται σε κάθε JML Statement ξεχωριστά είναι:

Για την έκφραση**: //@requires a != 0** και **//@requires b != 0**, έχουμε ότι στην περίπτωση που το a ή το b δεν ισούνται με μηδέν τότε η έκφραση επαληθεύεται, ενώ αντίθετα εάν ισούνται με μηδέν τότε η έκφραση είναι ψευδής και οι προδιαγραφές του συστήματος δεν επαληθεύονται.
Η έκφραση **//@ensures \result != 0;** σχηματίζει τις εξής κλάσεις ισοδυναμίας. Στην πρώτη περίπτωση το αποτέλεσμα της μεθόδου πρέπει να μην ισούται με μηδέν, όπου σε αυτή την περίπτωση η έκφραση επαληθεύεται και κατ' επέκταση οι προδιαγραφές. Στην δεύτερη περίπτωση όταν το αποτέλεσμα είναι μηδέν τότε θα δεν επαληθεύεται η έκφραση.

Για τις πιο πάνω κλάσεις δίνονται τιμές στο πρόγραμμα και αναλόγως τυπώνονται στις οθόνες με βάση την σειρά που βρέθηκαν και αναλόγως εάν οι εκφράσεις JML επαληθεύτηκαν ή όχι.



Εικόνα 5.7: Επιλογή 2ου Template και Ενεργοποίηση του Λογισμικού

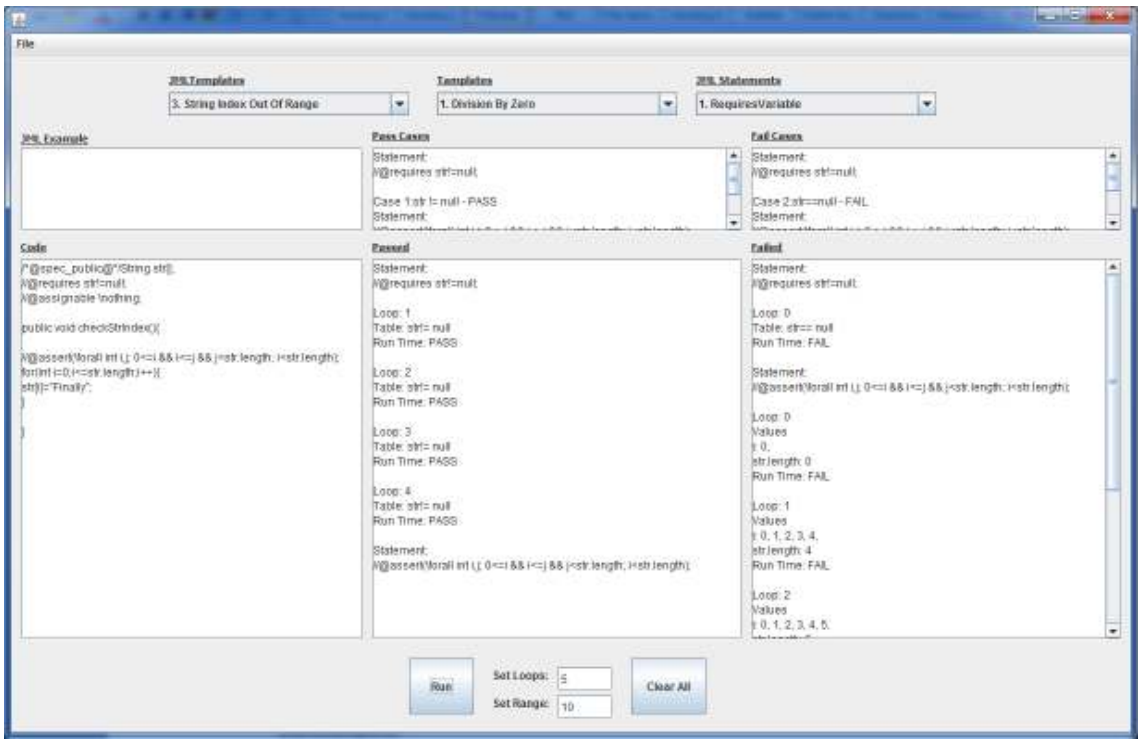
Στην περίπτωση του Template: Array Index Out Of Range, υπάρχουν τρεις εκφράσεις οι οποίες θα πάρουν τιμές και θα κρίνουν εάν το πρόγραμμα τηρεί τις προδιαγραφές του. Αυτές οι εκφράσεις είναι θα αναλυθούν πιο κάτω:

Στην πρώτη περίπτωση η έκφραση **//@requires a!=null;** Απαιτεί ότι η μεταβλητή a που σε αυτήν την περίπτωση είναι ένα χειριστήριο κάποιου πίνακα, δεν πρέπει να είναι Null. Έτσι οι κλάσεις ισοδυναμίας χωρίζονται σε δυο περιπτώσεις στην περίπτωση που το χειριστήριο είναι null και στην περίπτωση που δεν είναι.

Η έκφραση **//@ensures (\forall int i; 0<= i && i < a.length; c==a[a.length-1]);** είναι μια πιο σύνθετη έκφραση η οποία χωρίζεται σε τέσσερεις κλάσεις ισοδυναμίας όπου για να είναι αληθής και να ικανοποιεί τις προδιαγραφές του προγράμματος θα πρέπει να συμβαίνουν τα εξής: Η μεταβλητή i θα πρέπει να κυμαίνεται μεταξύ του μηδενός και του μεγέθους του πίνακα χωρίς να παίρνει την τελευταία τιμή. Επίσης κατά την έξοδο της μεθόδου η μεταβλητή c πρέπει να πάρει την τελευταία τιμή του πίνακα a. Υπάρχουν ακόμα τρείς κλάσεις ισοδυναμίας οι οποίες δεν πληρούν τις προδιαγραφές. Στην πρώτη κλάση η μεταβλητή i ξεκινάει με αριθμό μικρότερο του μηδενός, στη δεύτερη παίρνει

τιμής μεγαλύτερες ή ίσες με το μήκος του πίνακα και στην τρίτη κλάση το c στο τέλος του βρόχου δεν παίρνει την τελευταία τιμή που υπάρχει στον πίνακα.

Η επόμενη έκφραση **//@assert(\forall int i,j; 0<=i && i<=j && j<a.length; i<a.length);** , στην περίπτωση μας θα είναι πάντα ψευδής και αυτό οφείλεται στο κείμενο του κώδικα που δόθηκε. Οι κλάσεις ισοδυναμίας στην περίπτωση αυτή είναι τρεις. Η περίπτωση που επαληθεύει την έκφραση είναι όταν η μεταβλητή i, στο πρόγραμμα κυμαίνεται από μηδέν έως το μέγεθος του πίνακα. Εάν είναι μικρότερο του μηδενός η μεγαλύτερο του μεγέθους του πίνακα τότε αποτυγχάνει. Όμως όπως έχει ειπωθεί στην περίπτωση αυτή λόγω του κώδικα που υπάρχει, συγκεκριμένα στην έκφραση **for (int i = 0; i <= a.length; i++),** όπου βλέπουμε ότι αν το i παίρνει τιμή ίση του μεγέθους του πίνακα, δεν αφήνει την έκφραση να επαληθευτεί. Αυτό μπορούμε να το αλλάξουμε στον κώδικα και να βάλουμε στη θέση της ανίσωσης το μεγαλύτερο και να διαγράψουμε το ίσο. Στην περίπτωση αυτή το πρόγραμμα θα δίνει σωστά αποτελέσματα.

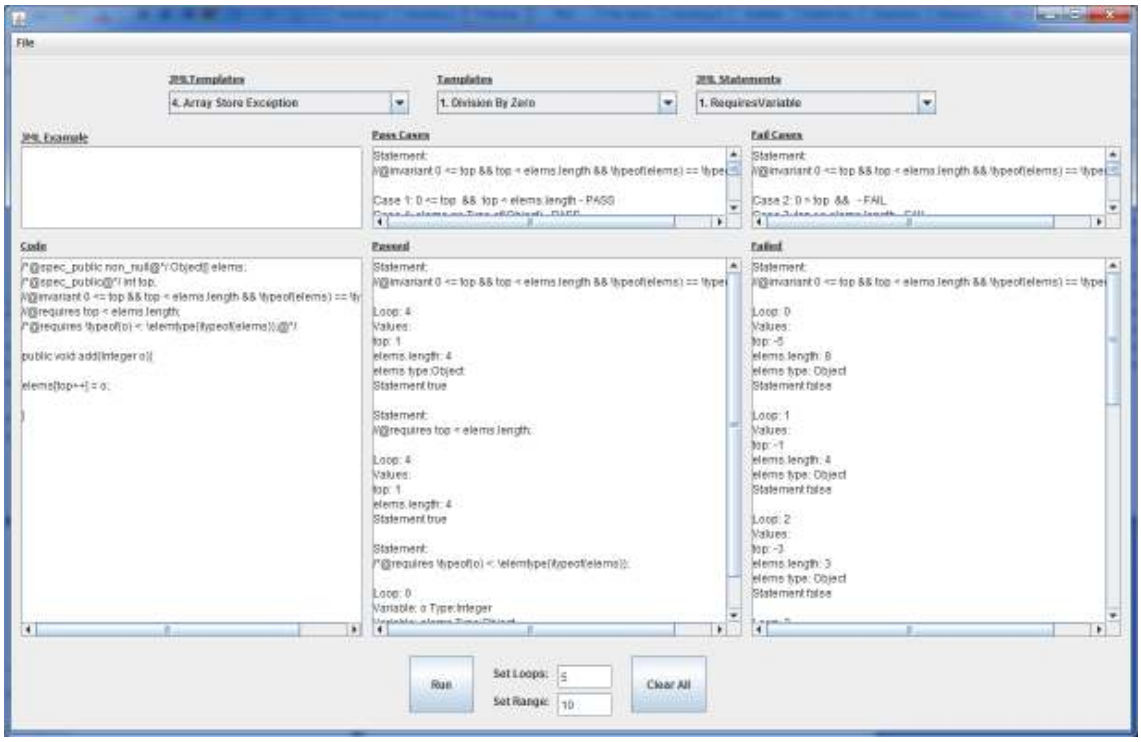


Εικόνα 5.8: Επιλογή 3ου Template και Ενεργοποίηση του Λογισμικού

Όταν επιλέξουμε το Template: String Index Out Of Range, και πιέσουμε το run, τότε βλέπουμε ότι υπάρχουν δυο εκφράσεις στις οποίες το λογισμικό δίνει τιμές.

Η πρώτη έκφραση είναι η **//@requires str!=null;** Η οποία περιορίζει το πρόγραμμα να πάρει η μεταβλητή str τιμή και να μην μένει με την τιμή null. Και σε αυτή την περίπτωση οι κλάσεις ισοδυναμίας είναι η περίπτωση που το str έχει κάποια τιμή και επαληθεύεται η έκφραση αλλά και η περίπτωση η οποία μένει με το null.

Η δεύτερη έκφραση που μας ενδιαφέρει είναι η **//@assert(\forall int i,j; 0<=i && i<=j && j<str.length; i<str.length);,** η οποία για να επαληθευτεί θα πρέπει η μεταβλητή i να κυμαίνεται από μεταξύ και του μεγέθους της μεταβλητής str. Αν είναι πιο μικρό η μεγαλύτερο ή ίσο από το str τότε η έκφραση αποτυγχάνει. Στην περίπτωση αυτή η έκφραση αποτυγχάνει πάντα επειδή το i παίρνει τιμές έως και ίσες της μεταβλητής str, έτσι θα πρέπει να αλλάξουμε τον κώδικα ένα θέλουμε να έχουμε σωστά αποτελέσματα.



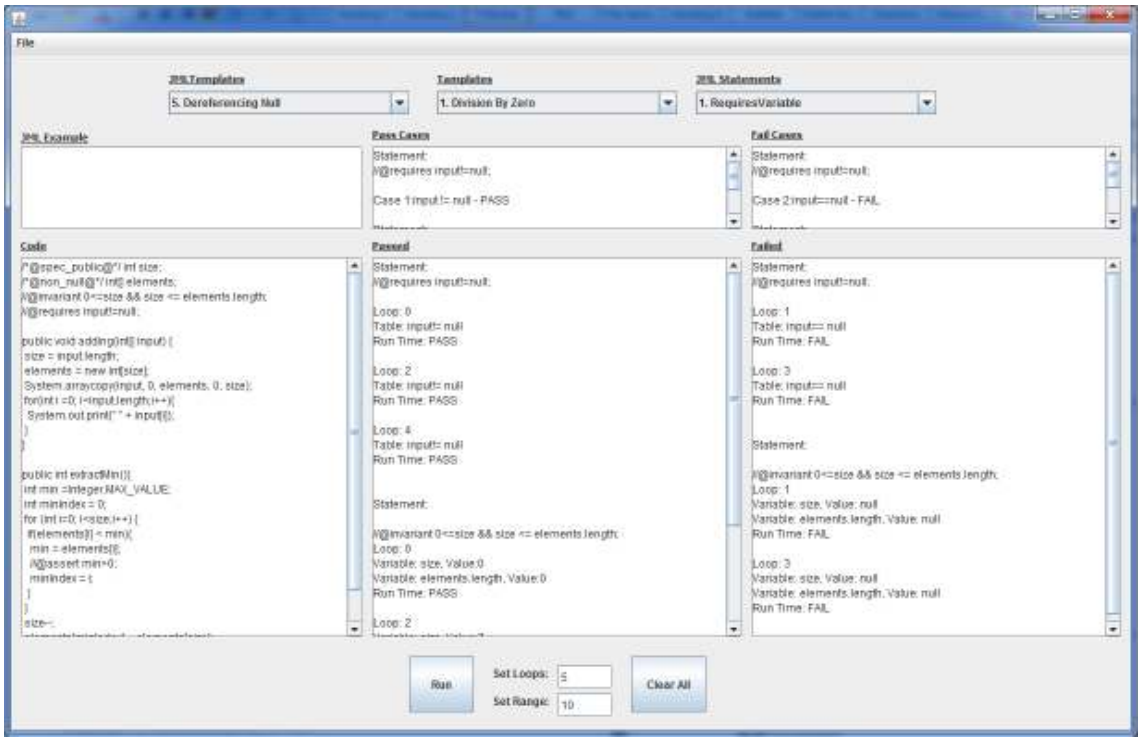
Εικόνα 5.9: Επιλογή 4ου Template και Ενεργοποίηση του Λογισμικού

Στο 4ο Template: Array Store Exception, υπάρχουν τρεις εκφράσεις οι οποίες τέθηκαν ως προδιαγραφές

Η πρώτη είναι η **//@invariant 0 <= top && top < elems.length && \typeof(elems) == \type(Object[]);** η οποία επαληθεύεται μόνο εάν ισχύει ότι η μεταβλητή top κυμαίνεται από το μηδέν μέχρι και χωρίς συμπεριλαμβανομένου του μεγέθους του πίνακα elems, και επίσης πρέπει να ισχύει ότι ο τύπος του πίνακα elems συμπεριλαμβάνεται στον τύπο Object. Εάν το top είναι μικρότερο του μηδενός ή είναι μεγαλύτερο ή ίσο του μεγέθους του πίνακα τότε η έκφραση αποτυγχάνει. Επίσης στην περίπτωση που ο τύπος του πίνακα δε συμπεριλαμβάνεται στο Object, τότε δεν επαληθεύεται η έκφραση. Σε αυτή την περίπτωση ο τύπος είναι object, έτσι αυτό το σκέλος είναι πάντα αληθές, και για να παρατηρήσουμε το αντίθετο θα πρέπει να επέμβουμε στον κώδικα, αλλάζοντας τον τύπο.

Η δεύτερη έκφραση που μας ενδιαφέρει είναι η **//@requires top < elems.length;**, η οποία δίνει την προδιαγραφή στο πρόγραμμα ότι η τιμή που θα πάρει η μεταβλητή top να είναι μικρότερη του μεγέθους του πίνακα. Σε αντίθετη περίπτωση η έκφραση αποτυγχάνει.

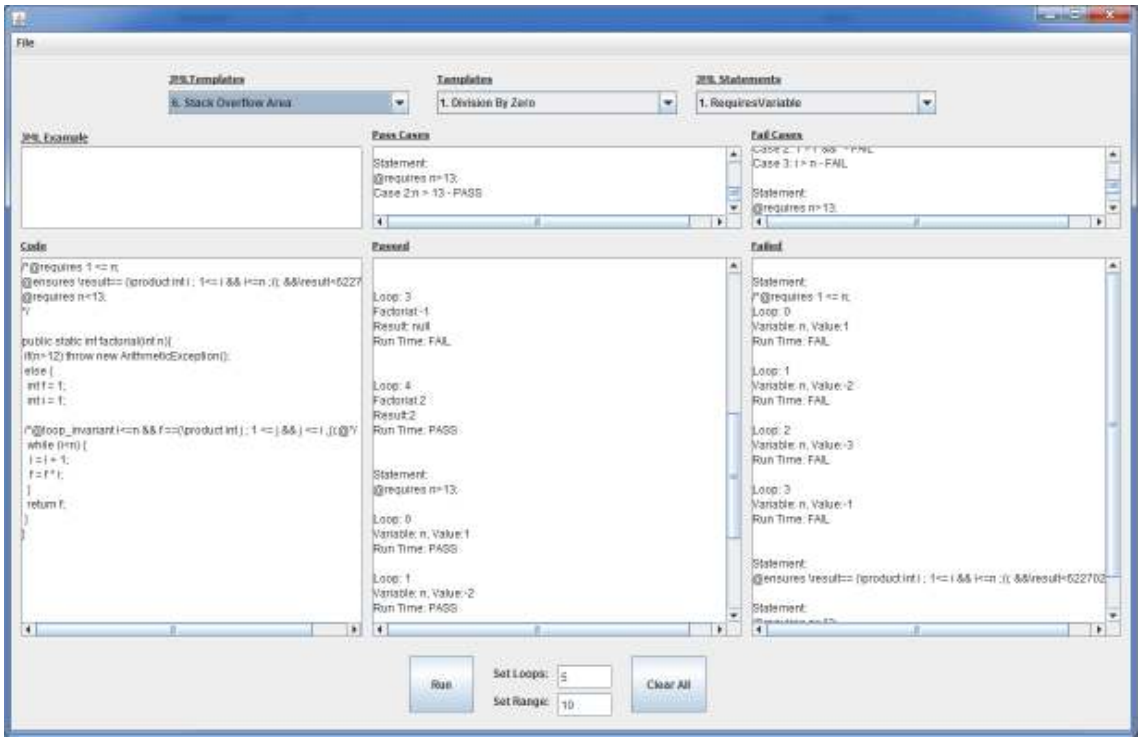
Η Τρίτη και τελευταία έκφραση που συναντάμε σε αυτό το Template είναι η εξής: **/*@requires \typeof(o) <: \elemtype(\typeof(elems));@*/**, η οποία για να επαληθευτεί θα πρέπει ο τύπος δεδομένων της μεταβλητής «ο» να συμπεριλαμβάνεται στον τύπο δεδομένων του πινάκα elems. Εάν δεν συμβαίνει αυτό τότε η έκφραση αποτυγχάνει. Σε αυτή την περίπτωση, ο τύπος του ο που είναι Integer συμπεριλαμβάνεται στο Object έτσι η έκφραση είναι αληθής. Επίσης το αποτέλεσμα τυπώνεται μόνο μια φορά σαν έξοδος επειδή δεν αλλάζει με νέα δεδομένα στον κώδικα, αφού η αλλαγή θα πρέπει να γίνει στο κείμενο που αναγράφεται στο σημείο του τύπου της μεταβλητής.



Εικόνα 5.10: Επιλογή 5ου Template και Ενεργοποίηση του Λογισμικού

Το 5ο Template: Dereferencing Null, έχει δυο προδιαγραφές τις οποίες θα αναλύσουμε. Η πρώτη έκφραση που συναντάμε είναι η **//@invariant 0<=size && size <= elements.length;** η οποία αναλύεται και αποφασίζεται ότι υπάρχουν τρεις κλάσεις ισοδυναμίας, Η πρώτη για την οποία επαληθεύεται η έκφραση απαιτεί όπως η μεταβλητή size να κυμαίνεται μεταξύ του μηδενός και του μεγέθους του πίνακα elements, συμπεριλαμβανομένων των δυο τιμών. Εάν είναι μικρότερο του μηδενός η μεγαλύτερο του πίνακα τότε η έκφραση αποτυγχάνει. Επίσης εάν ο πίνακας είναι null τότε η έκφραση δεν παίρνει θετική τιμή.

Η δεύτερη έκφραση που υπάρχει σε αυτό το πρότυπο είναι η **//@requires input!=null;** όπου βάζει την προδιαγραφή η οποία ορίζει ότι το input πρέπει να έχει κάποια τιμή για να είναι αληθής, και να μην μένει άδειο με τιμή null.



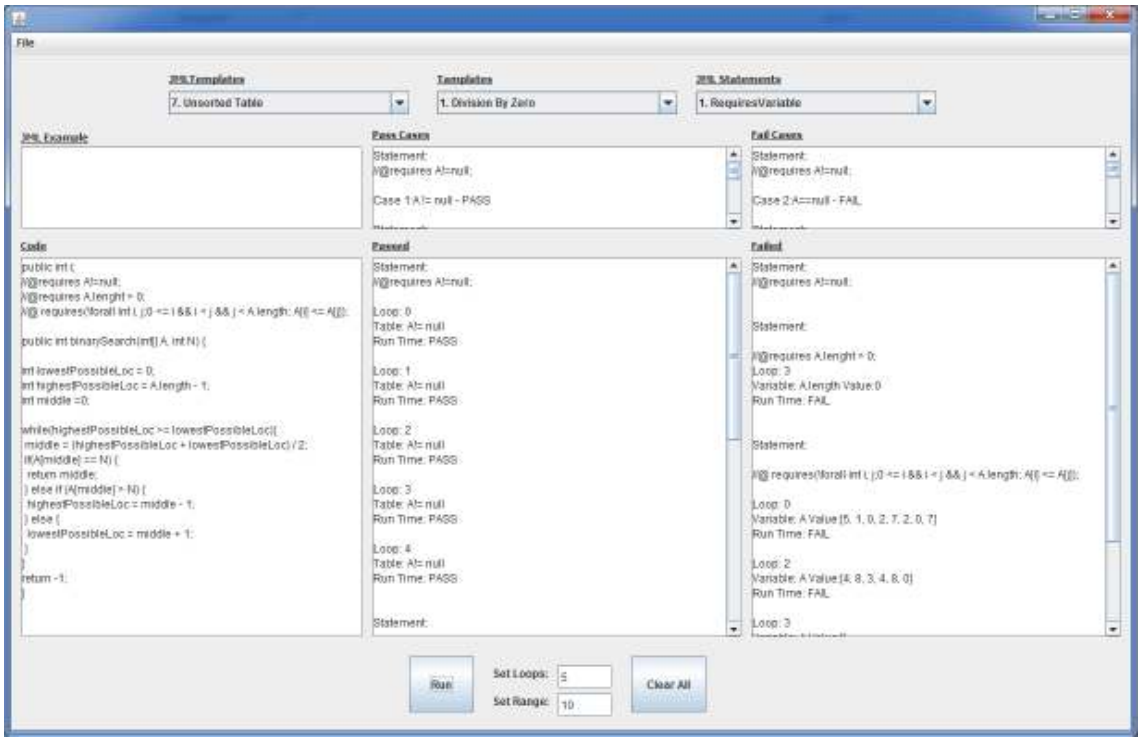
Εικόνα 5.11: Επιλογή 6ου Template και Ενεργοποίηση του Λογισμικού

Στο 6ο Template: Stack Overflow Area, δύνονται τρεις προδιαγραφές οι οποίες αφορούν το παραγοντικό.

Στην περίπτωση τις έκφρασης **/*@requires 1 <= n;** Προδιαγράφεται ότι η μεταβλητή n, θα πρέπει να είναι μεγαλύτερη ή ίση του ενός. Σε αλλιώτικη περίπτωση η έκφραση γίνεται ψευδής.

Η έκφραση αυτή @ensures **\result== (\product int i ; 1<= i && i<=n ;i); &&\result<6227020800;** προδιαγράφει στο πρόγραμμα πρώτον ότι η μεταβλητή i θα πρέπει να κυμαίνεται από το ένα μέχρι και το n. Εάν είναι μικρότερη του 1 ή μεγαλύτερη ή ίση του n τότε η προδιαγραφές του προγράμματος δεν τηρούνται. Επίσης η έκφραση αυτή ζητάει ότι το αποτέλεσμα της μεθόδου να είναι μικρότερο από τον δοθέντα αριθμό.

Τέλος η έκφραση **@requires n<13;** Προδιαγράφει το n να είναι μικρότερο από το 13, για να μην έχουμε προβλήματα υπερχείλισης της μεταβλητής.

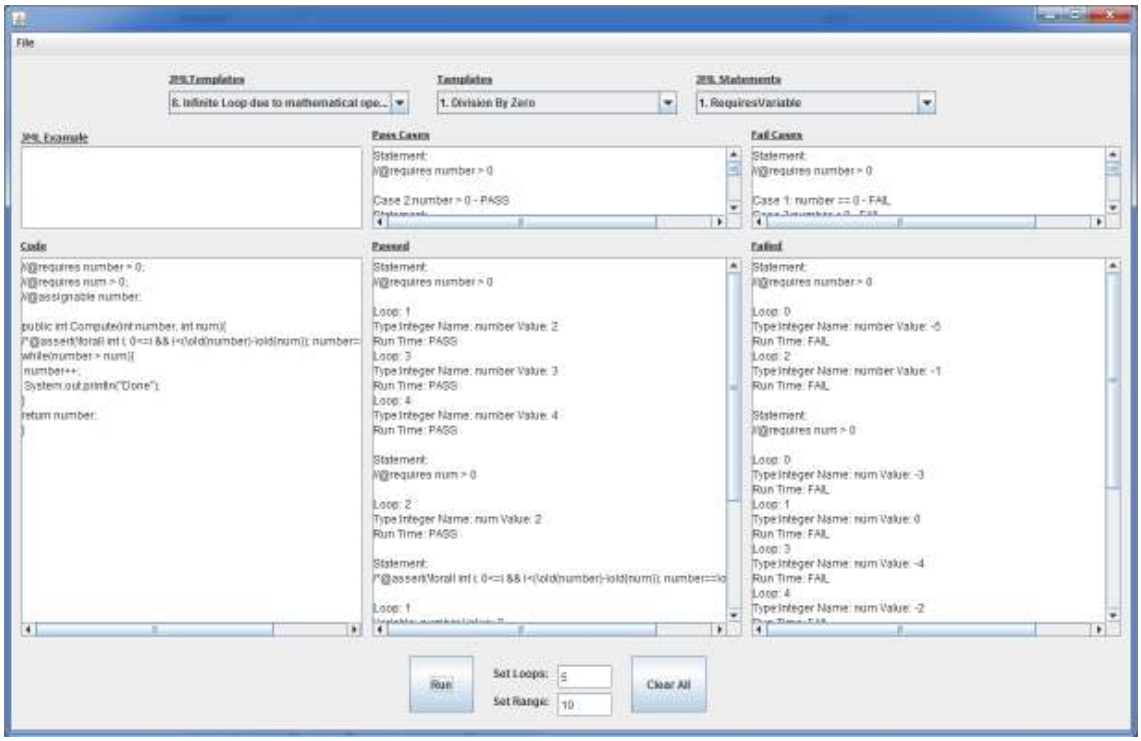


Εικόνα 5.12: Επιλογή 7ου Template και Ενεργοποίηση του Λογισμικού

Στο επόμενο Template υπ' αριθμό 7 και όνομα: Unsorted Table, υπάρχουν τρεις προδιαγραφές οι οποίες

Η πρώτη **//@requires A!=null;** η οποία ορίζει ότι το χειριστήριο του πίνακα δεν θα είναι null αλλά θα έχει κάποια τιμή καθώς επίσης και η έκφραση //@requires A.length > 0; Όπου προδιαγράφεται το μέγεθος του πίνακα να είναι μεγαλύτερο του μηδενός. Σε αντίθετες περιπτώσεις οι εκφράσεις αποτυγχάνουν όπως φαίνεται στο σχήμα 5.510.

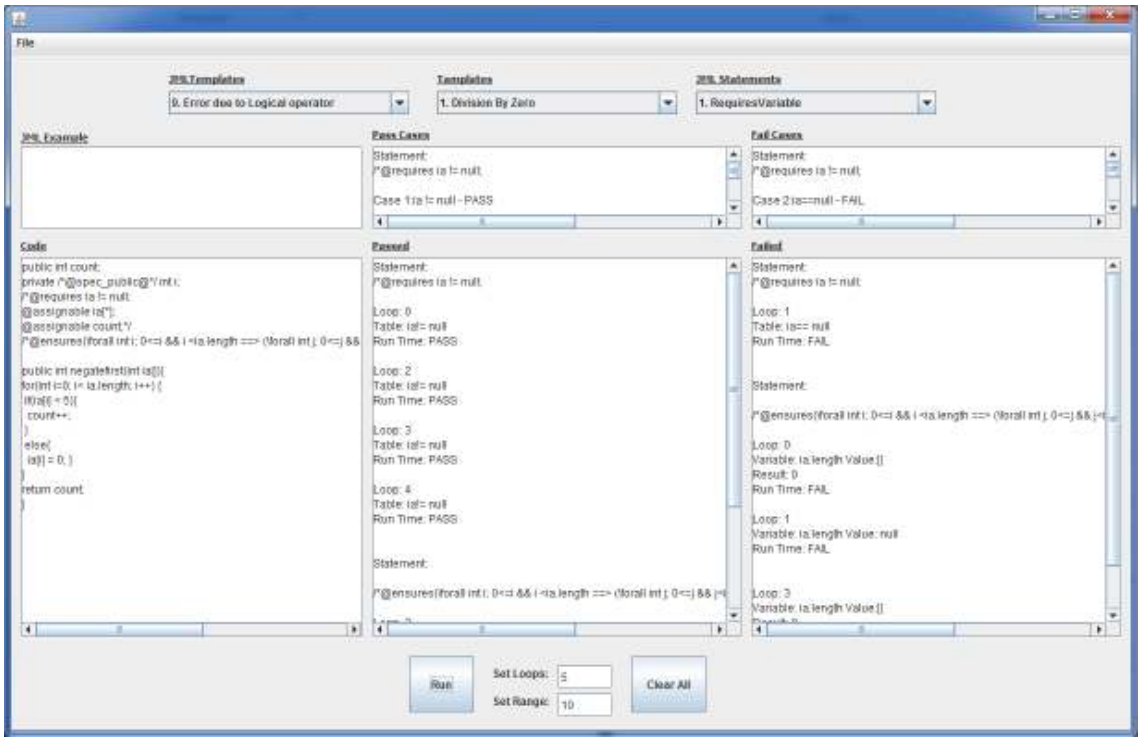
Η επόμενη έκφραση είναι η //@ **requires(\forall int i, j;0 <= i && i < j && j < A.length; A[i] <= A[j]);** όπου καθορίζει ότι η μεταβλητή i θα πρέπει να παίρνει τιμές από μηδέν μέχρι το μήκος του πίνακα μη συμπεριλαμβανομένης της τιμής εκείνης αλλά και ότι το κάθε στοιχείο του πίνακα θα πρέπει να είναι μικρότερο από το προηγούμενο, έτσι ώστε ο πίνακας να είναι ταξινομημένος. Εάν δεν είναι ταξινομημένος ή το i δεν έχει τις τιμές που ειπώθηκαν τότε οι έκφραση κρίνεται ψευδής και εμφανίζονται τα αποτελέσματα στην οθόνη.



Εικόνα 5.13: Επιλογή 8ου Template και Ενεργοποίηση του Λογισμικού

Φτάνοντας στο 8ο Template με όνομα: Infinite Loop due to mathematical operator, παρατηρούμε ότι υπάρχουν δυο request εκφράσεις προδιαγραφών και μια assert.
Οι πρώτες δυο **//@requires number > 0;** και **//@requires num > 0;** είναι ευκολοκατανόητες. Δηλαδή προδιαγράφουν το number και το num να είναι μεγαλύτερα του μηδενός, για να επαληθεύσουν την έκφραση, ενώ σε αντίθετη περίπτωση η έκφραση θα είναι ψευδής και θα φανεί στην οθόνη.

Η άλλη έκφραση που βλέπουμε ορίζεται ως εξής: **/*@assert(\forall int i; 0<=i && i<(\old(number)-\old(num)); number==\old(number)-1);@*/** Αυτή η έκφραση προδιαγράφει ότι οι επαναλήψεις που θα πραγματοποιηθούν στον βρόγχο που ακολουθεί θα είναι ο αριθμός που βγαίνει από την πράξη: number-num, καθώς επίσης και κατά τη διάρκεια του βρόγχου ο αριθμός num να είναι ισοδύναμος με τον προηγούμενο -1. Εάν σε αντίθετη περίπτωση ο αριθμός τον βρόγχων δεν είναι number-num τότε η έκφραση αποτυγχάνει. Επίσης εάν ο αριθμός num, δεν είναι σε κάθε επανάληψη κατά ένα μικρότερος τότε η έκφραση θα είναι ψευδής, και τα αποτελέσματα θα φανούν στην οθόνη. Σε αυτή την περίπτωση θα έχουμε συνεχώς αρνητικά αποτελέσματα επειδή όπως βλέπουμε ο μετρητής αυξάνεται αντί να μειώνεται έτσι έχουμε ατέρμονα βρόγχο λόγω λανθασμένου μαθηματικού τελεστή.



Εικόνα 5.14: Επιλογή 9ου Template και Ενεργοποίηση του Λογισμικού

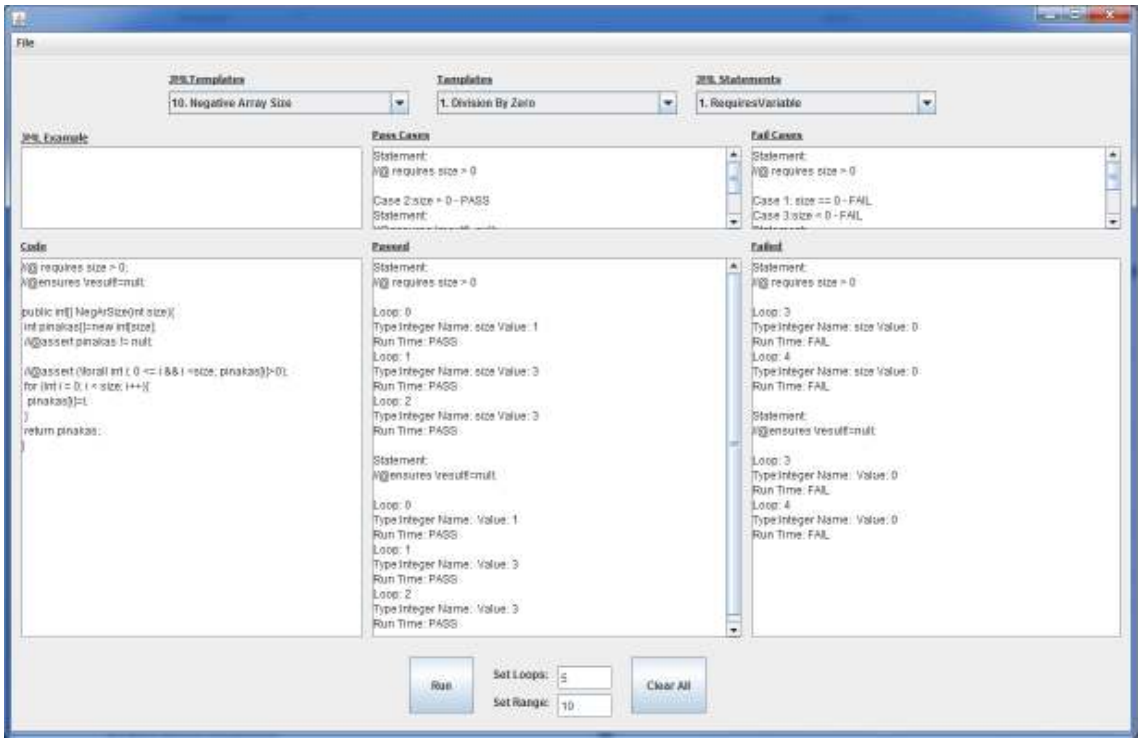
Στο 9ο Template: Error due to Logical operator, υπάρχουν αρκετές JML εκφράσεις, αλλά μόνο δυο από αυτές αποτελούν προδιαγραφές οι οποίες παίρνουν τιμές για να τις επεξεργαστούν.

Η πρώτη έκφραση είναι η **/*@requires ia != null;** η οποία προδιαγράφει το χειριστήριο του πίνακα ia να πάρει κάποια τιμή και να μην είναι null

Η επόμενη έκφραση είναι η **/*@ensures(\forall int i; 0<=i && i <ia.length ==> (\forall int j; 0<=j && j<i ==> (\old(ia[j])>5))) ? (count==count+1) : (ia[i]== 0); @*/** Σε αυτήν την έκφραση οι κλάσεις ισοδυναμίας οι οποίες έχουν θετικό αποτέλεσμα καθορίζονται από την τιμή που θα πάρει το i το οποίο πρέπει να είναι μεταξύ του μηδέν και του μεγέθους του πίνακα χωρίς να παίρνει την τιμή εκείνη.

Επίσης εάν η τιμή που ελέγχεται στον πίνακα ia είναι μεγαλύτερη του 5 τότε το counter πρέπει να αυξηθεί κατά ένα και εάν η τιμή που ελέγχεται στον πίνακα είναι μικρότερη του μηδέν τότε η τιμή πρέπει να μηδενιστεί**.** Σε αντίθετη περίπτωση εάν το I είναι μικρότερο του μηδενός ή μεγαλύτερο ή ίσο από το μέγεθος του πίνακα η έκφραση αποτυγχάνει. Επίσης εάν η τιμή στη θέση του πίνακα που ελέγχεται είναι μεγαλύτερη

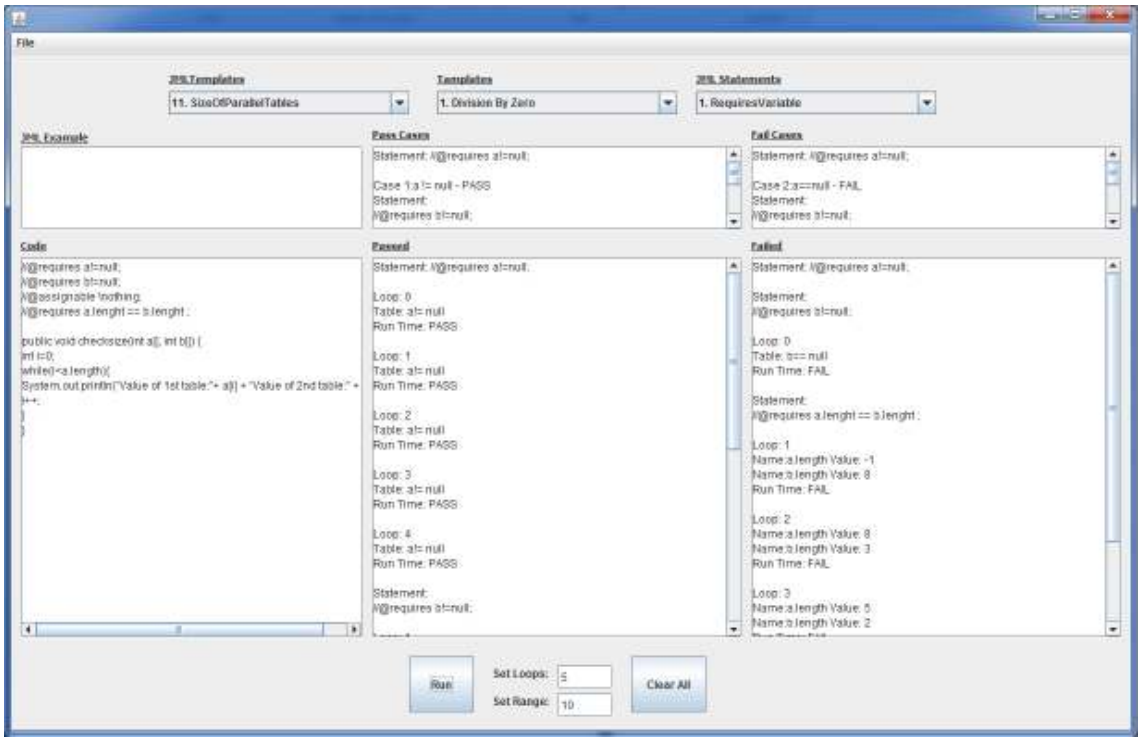
του 5 και το counter δεν αυξηθεί ή εάν η τιμή είναι μικρότερη του 5 και δεν πάρει την τιμή μηδέν, τότε και στις δυο περιπτώσεις έχουμε λάθος στις προδιαγραφές του προγράμματος .



Εικόνα 5.15: Επιλογή 10ου Template και Ενεργοποίηση του Λογισμικού

Στο 10ο Template: Negative Array Size, θα αναφερθούμε σε 2 JML Statements.
Το πρώτο είναι το //@ **requires size > 0;,** όπου προδιαγράφεται η τιμή της μεταβλητής size να είναι μεγαλύτερη του μηδενός, ενώ σε αντίθετη περίπτωση έχουμε αποτυχία.

Επίσης το statement: //**@ensures \result!=null;** το οποίο εγγυάται ότι το αποτέλεσμα της μεθόδου δεν θα είναι null. Οι τιμές δύνονται στο πρόγραμμα με τον τρόπο που αναφέρθηκε στις αρχές του κεφαλαίου και εάν το αποτέλεσμα είναι null, τότε τυπώνεται ως αποτυχία στην οθόνη, ή τυπώνεται ως επιτυχία εάν το αποτέλεσμα δεν είναι null.

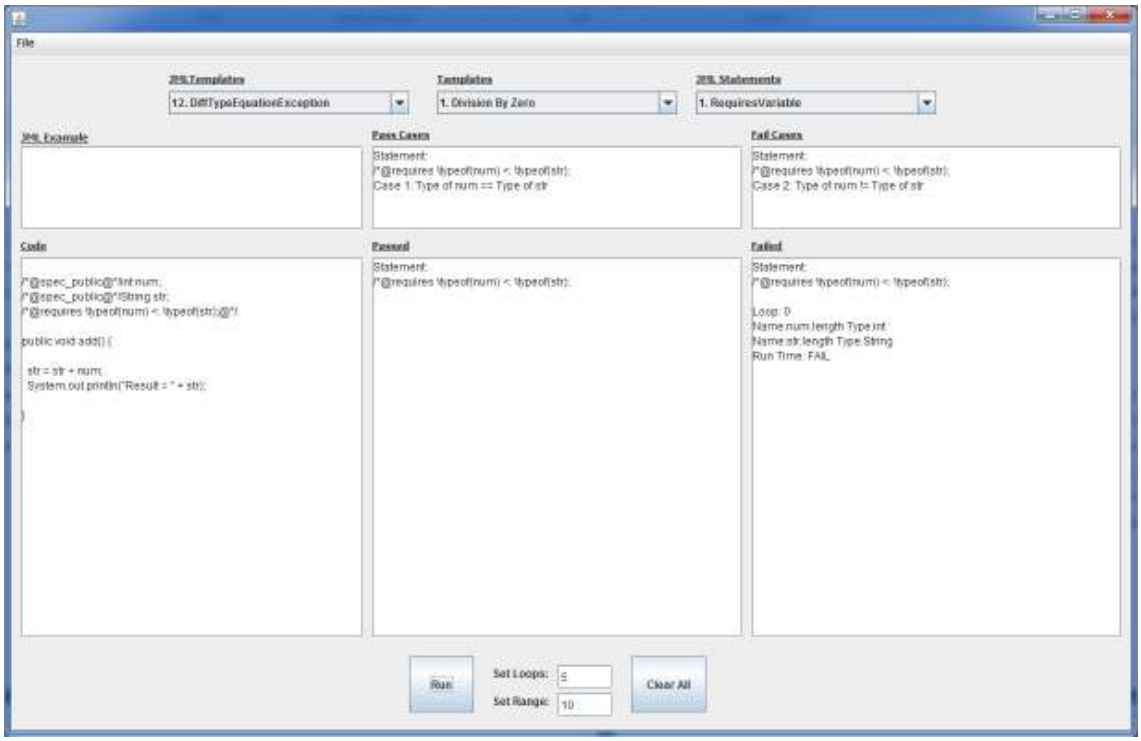


Εικόνα 5.16: Επιλογή 11ου Template και Ενεργοποίηση του Λογισμικού

Φθάνοντας στα πιο πρόσφατα Templates, παρουσιάζουμε το 11ο με όνομα: SizeOfParallelTables, το οποίο προδιαγράφει ότι οι τιμές των δυο παράλληλων πινάκων θα είναι ίσες με τον εξής τρόπο:

Αρχικά προδιαγράφουμε τα χειριστήρια των δυο πινάκων να μην είναι null **//@requires a!=null;** και **//@requires b!=null;** . Στη συνέχεια δίνονται οι τιμές και εάν δεν είναι κανένα από τα δυο null, τότε επαληθεύονται οι δηλώσεις. Σε αντίθετη περίπτωση εάν κάποιο από τα δύο είναι null τότε η συγκεκριμένη δήλωση είναι ψευδής και αυτό φαίνεται στην οθόνη.

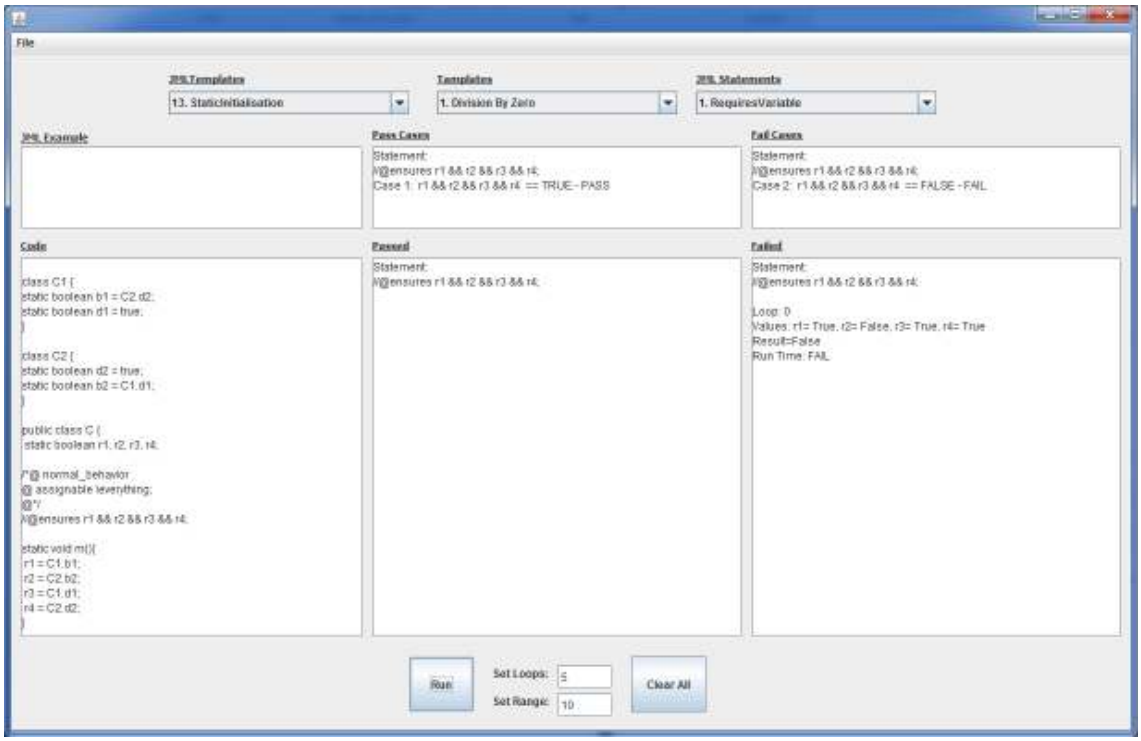
Η Επόμενη προδιαγραφή που θα χρειαστεί να ελέγξουμε είναι η **//@requires a.lenght == b.lenght ;** η οποία ελέγχει εάν τα δυο μεγέθη είναι ίσα, που σε αυτήν την περίπτωση οι προδιαγραφές θα είναι ορθές. Εάν τα μεγέθη δεν είναι ίσα, είτε κάποιο από τα μεγέθη είναι null, τότε απευθείας η έκφραση γίνεται ψευδής. Στο τέλος παρουσιάζεται στην οθόνη στα δεξιά εάν είναι αληθής η πρόταση ή στα αριστερά εάν είναι ψευδής. Τα στοιχεία που παρουσιάζονται είναι η επανάληψη που βρήκε τα αποτελέσματα, οι τιμές των μεταβλητών αλλά και το τελικό αποτέλεσμα.



Εικόνα 5.17: Επιλογή 12ου Template και Ενεργοποίηση του Λογισμικού

Το επόμενο Template έχει αριθμό 12 και ονομάζεται: DiffTypeEquationException, επίσης για αυτό το πρότυπο χρειάστηκε μόνο ένας έλεγχος για να μπουν οι προδιαγραφές.

Η έκφραση που χρησιμοποιήσαμε ήταν η **/*@requires \typeof(num) <: \typeof(str);@*/** , όπου προδιαγράφουν ότι ο τύπος του num, θα είναι ίδιος με τον τύπο της μεταβλητής str. Οι δυο αυτοί τύποι καθορίζονται από τις δηλώσεις στο πάνω μέρος του κώδικα. Επίσης πρέπει να αναφέρουμε ότι σε αυτήν την περίπτωση δεν γίνεται ανάθεση τυχαίων τιμών αφού, παίρνει τις τιμές απευθείας από το κείμενο του κώδικα, έτσι στο παράδειγμα που δώσαμε η έκφραση αυτή θα είναι πάντα ψευδής, μέχρι που να αλλάξουμε τον κώδικα και να βάλουμε δυο ίδιους τύπους.
Η επανάληψη επίσης γίνεται μόνο μια φορά για τον λόγο που προαναφέραμε, ότι δεν υπάρχουν τυχαίες αναθέσεις τιμών αλλά οι προκαθορισμένες τιμές από τον κώδικα.

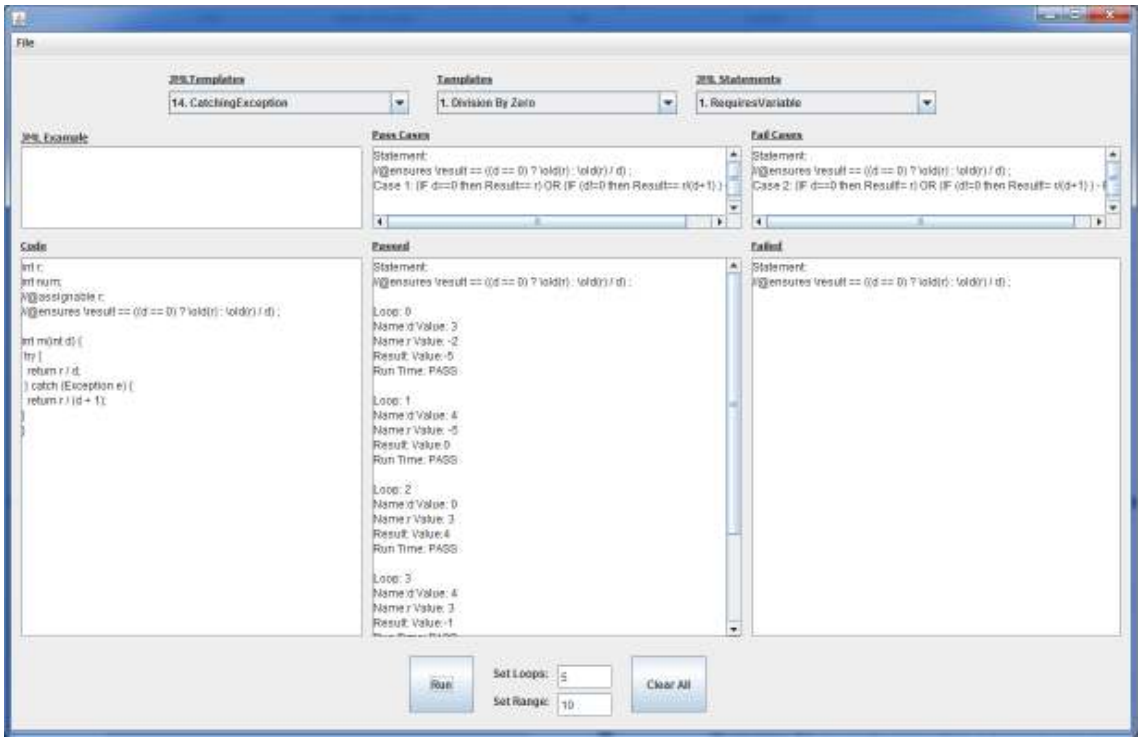


Εικόνα 5.18: Επιλογή 13ου Template και Ενεργοποίηση του Λογισμικού

Συνεχίζουμε με το 13ο Template που ονομάζεται: StaticInitialisation, το οποίο προδιαγράφει ότι οι στατικές μεταβλητές που φαίνονται στο σύστημα, θα αρχικοποιηθούν κανονικά μετά την ανάθεση που γίνεται στην συγκεκριμένη μέθοδο.

Η έκφραση που θα μας απασχολήσει είναι η **//@ensures r1 && r2 && r3 && r4;** η οποία εγγυάται ότι μετά το τέλος της μεθόδου η έκφραση θα πρέπει να έχει την τιμή true, για να ικανοποιηθούν οι προδιαγραφές που θέλουν τις τέσσερις μεταβλητές να αρχικοποιούνται με την τιμή true.

Οι κλάσεις ισοδυναμίας που δημιουργούνται σε αυτήν την περίπτωση είναι η έκφραση r1 && r2 && r3 && r4 να έχει την τιμή true η οποία κάνει αληθές το statement του JML, και εάν το r1 && r2 && r3 && r4 έχει τιμή false, τότε αυτόματα η προδιαγραφές δεν ικανοποιούνται και βγαίνει το ανάλογο μήνυμα στην οθόνη. Το μήνυμα και σε αυτή την περίπτωση βγαίνει μόνο μια φορά ανεξαρτήτως του αριθμού των επαναλήψεων που θα δώσει ο χρήστης επειδή οι τιμές καθορίζονται απευθείας από τον κώδικα.



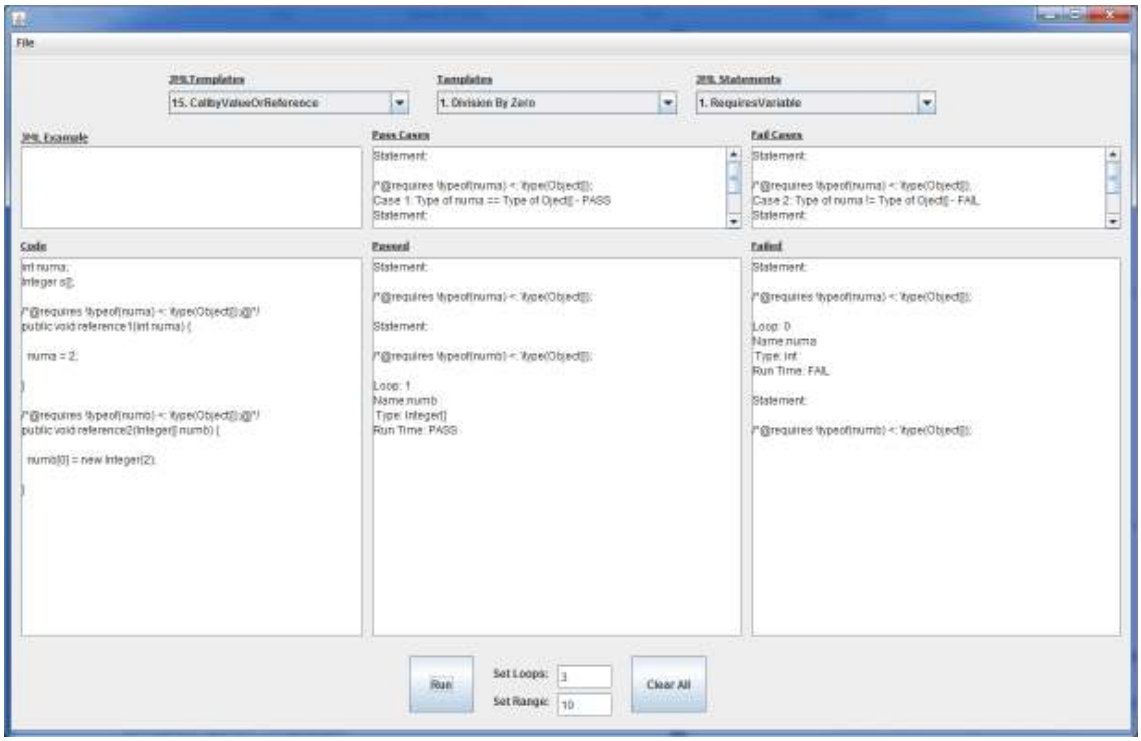
Εικόνα 5.19: Επιλογή 14ου Template και Ενεργοποίηση του Λογισμικού

Στο 14ο Template που ονομάζεται: CatchingException, χρησιμοποιούμε ένα JML statement για να προδιαγράψουμε ότι το πρόγραμμα θα πιάσει το exception.

Η έκφραση που χρησιμοποιούμε σε αυτήν την περίπτωση είναι**: //@ensures \result == ((d == 0) ? \old(r) : \old(r) / d) ;** η οποία με βάση την πιο κάτω μέθοδο στον κώδικα, ελέγχει εάν πιάστηκε το exception και το πρόγραμμα συνέχισε κανονικά. Οι κλάσεις ισοδυναμίας που παρατηρούνται είναι δυο. Στην περίπτωση όπου η μεταβλητή d πάρει την τιμή μηδέν τότε το αποτέλεσμα τις μεθόδου θα πρέπει να είναι η τιμή που εμπεριέχεται στο r αλλιώς εάν το d δεν είναι μηδέν τότε το αποτέλεσμα πρέπει να είναι η διαίρεση r/(d+1). Εάν συμβαίνει κάτι από αυτά τότε η έκφραση είναι ορθή και οι προδιαγραφές τηρήθηκαν στο πρόγραμμα.

Η άλλη περίπτωση που η έκφραση δεν θα επαληθευτεί είναι όταν το d έχει την τιμή μηδέν αλλά το αποτέλεσμα δεν είναι η τιμή του r, ή όταν το d δεν πάρει την τιμή μηδέν και το αποτέλεσμα δεν είναι r/(d+1)

Στο παράδειγμα μας πρέπει να αναφέρουμε ότι η έκφραση θα είναι πάντα αληθής αφού με βάση τον κώδικα το αποτέλεσμα θα είναι πάντα το ίδιο. Εάν αλλάξουμε τον κώδικα και αφαιρέσουμε το σημείο του catch και κάτω τότε θα διαπιστώσουμε ότι στην περίπτωση που το d είναι μηδέν οι προδιαγραφές δεν επαληθεύονται.



Εικόνα 5.20: Επιλογή 15ου Template και Ενεργοποίηση του Λογισμικού

Φθάνοντας στο τελευταίο Template υπ αριθμό 15 και ονομασία: **CallbyValueOrReference**, θα δώσουμε προδιαγραφές στο πρόγραμμα οι οποίες θα εγγυούνται ότι το κάλεσμα τις μεθόδου δεν γίνεται με value δηλαδή η τιμές που δίνονται στην μέθοδο δεν αλλάζουνε και έξω από την μέθοδο. Αλλά γίνεται με την τυπική περίπτωση καλέσματος με Reference, όπου οι τιμές αλλάζουν μετά την έξοδο τους από την μέθοδο.

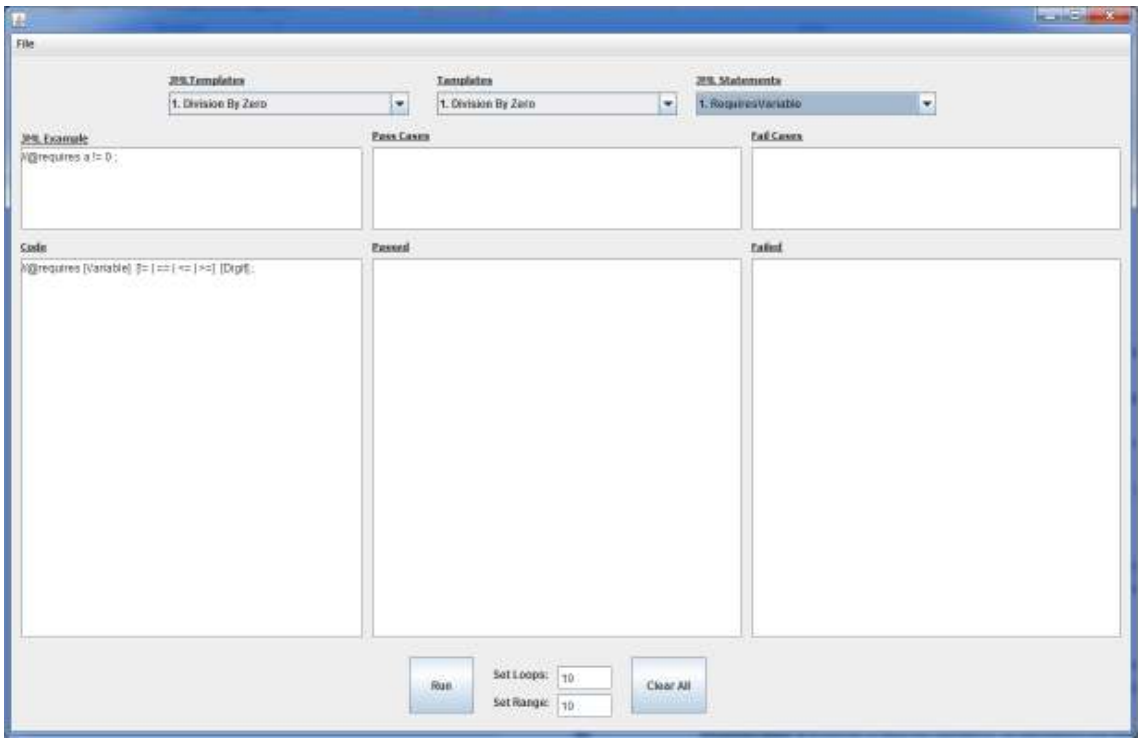
Στο παράδειγμα αυτό θα ασχοληθούμε με μια JML έκφραση η οποία παρουσιάζεται δυο φορές με διαφορετικές μεταβλητές. Η έκφραση είναι η **/*@requires \typeof(numa) <: \type(Object[]);@*/** και η **/*@requires \typeof(numb) <: \type(Object[]);@*/** Και στις δυο περιπτώσεις για να επαληθευτεί η έκφραση πρέπει η πρώτη μεταβλητή να είναι τύπου Object[]. Στην πρώτη περίπτωση βλέπουμε ότι το JML statement αποτυγχάνει αφού ο τύπος δεδομένων του numa είναι int, ενώ στην δεύτερη περίπτωση ο τύπος δεδομένων του numb είναι Integer[], ο οποίος είναι τύπος Object[]. Σε αυτή την περίπτωση οι προδιαγραφές επαληθεύονται και οι τιμή που δόθηκε μέσα στην μέθοδο παραμένει και έξω από αυτήν.

5.5.2 Περιγραφή των JML Statements

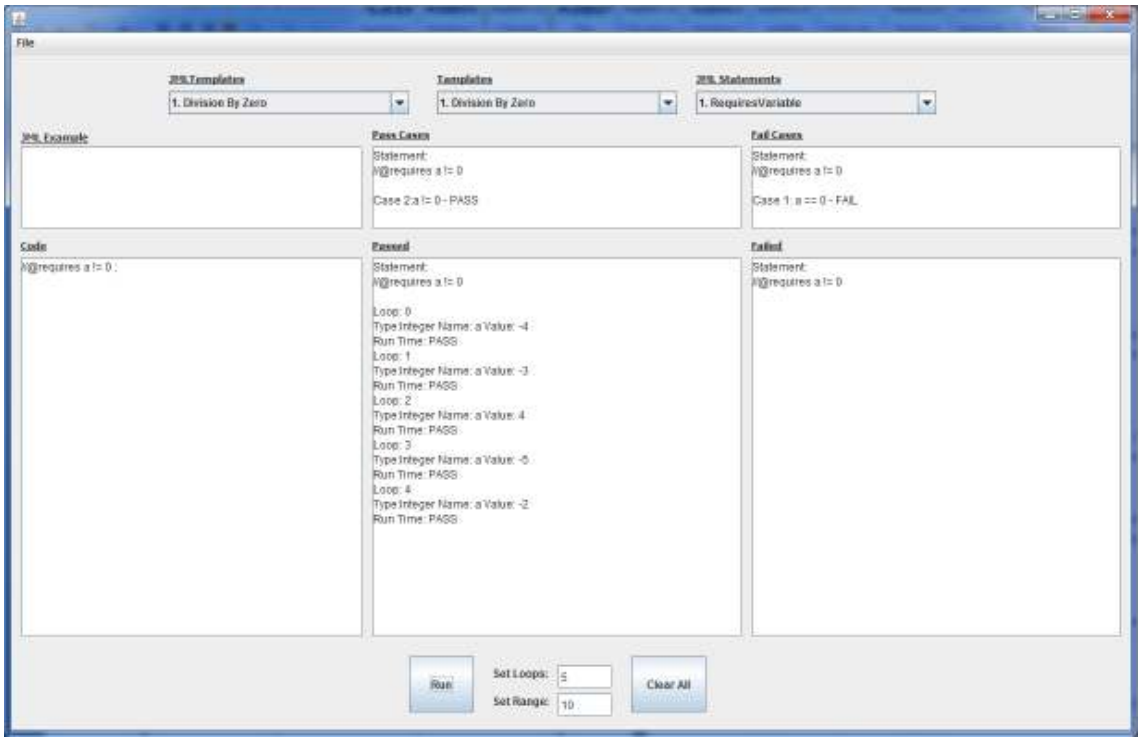
Η επόμενη λειτουργία του λογισμικού που θα παρουσιάσουμε θα είναι η λίστα με τα JML Statements, η οποία βρίσκεται τοποθετημένη στο πάνω δεξιά τμήμα του λογισμικού.

Όπως έχουμε αναφέρει η λίστα αυτή περιέχει εκφράσεις JML οι οποίες μπορούν να τοποθετηθούν σε οποιοδήποτε σημείο, στον κώδικα μας επιθυμούμε, φτάνει να βάλουμε τον δρομέα του ποντικιού στο σημείο που θέλουμε να μπει η έκφραση, και να την επιλέξουμε από την λίστα. Μόλις το κάνουμε αυτό τότε η έκφραση θα βγει στη γραμμή του δρομέα, και θα πρέπει ο χρήστης με την σειρά του να την συμπληρώσει. Για περισσότερη βοήθεια του χρήστη πάνω από το κείμενο του κώδικα θα εμφανιστεί ένα συμπληρωμένο παράδειγμα του συγκεκριμένου JML statement, και ο χρήστης θα μπορεί να συμπληρώσει και το δικό του αναλόγως, ή και ακόμα να το αντιγράψει όπως είναι στο παράδειγμα.

Πιο κάτω θα δείξουμε πως φαίνεται αυτή η διαδικασία στην οθόνη και έπειτα θα αναλύσουμε τις εκφράσεις αυτές εξηγώντας πως μπορούν να χρησιμοποιηθούν.



Εικόνα 5.21: Επιλογή 1ου JML Statement



Εικόνα 5.22: Επιλογή 1ου JML Statement με αποτέλεσμα από το λογισμικό

Οι πιο κύριες κατηγορίες των εκφράσεων που μπορεί να διαλέξει ο χρήστης είναι οι requires , assert , invariant και ensures. Με σειρά οι γενικές εκφράσεις δίνονται πιο κάτω:

1η επιλογή: RequiresVariable

Έκφραση τύπου requires η οποία, προδιαγράφει μία μεταβλητή με βάση την τιμή που θα πάρει από το πρόγραμμα. Μπορούμε να βάλουμε όνομα μεταβλητής την ανισότητα αλλά και τον αριθμό της επιθυμίας μας.

2η επιλογή: RequiresNullHandle,

Έκφραση τύπου requires η οποία, προδιαγράφει ένα χειριστήριο εάν έχει πάρει τιμή ή εάν έχει τιμή null. Μπορούμε να βάλουμε όνομα του χειριστηρίου και την ανισότητα.

3η επιλογή: RequiresDigit

Έκφραση τύπου requires η οποία, προδιαγράφει έναν αριθμό με βάση την τιμή μίας μεταβλητής που θα δοθεί στο πρόγραμμα. Μπορούμε να ορίσουμε τον αριθμό την ανισότητα αλλά και το όνομα της μεταβλητής.

4η επιλογή: RequiresDigit_toTableLength,

Έκφραση τύπου requires η οποία, προδιαγράφει μία μεταβλητή με βάση το μέγεθος κάποιου πίνακα που θα αρχικοποιηθεί.

5η επιλογή: RequiresTableLengthtoDigit,

Έκφραση τύπου requires η οποία, προδιαγράφει το μέγεθος κάποιου πίνακα με βάση την τιμή που θα πάρει από το πρόγραμμα.

6η επιλογή: RequiresSortedTable,

Έκφραση τύπου requires η οποία, προδιαγράφει ότι ο πίνακας που θα δοθεί θα είναι ταξινομημένος.

7η επιλογή: RequiresSameTableLength,

Έκφραση τύπου requires η οποία, προδιαγράφει ότι δύο πίνακες έχουν το ίδιο μήκος.

8η επιλογή: RequiresSameType

Έκφραση τύπου requires η οποία, προδιαγράφει τους τύπους δυο μεταβλητών να είναι οι ίδιοι. Σε αυτή την περίπτωση ο χρήστης χρειάζεται να δώσει τον ορισμό τον μεταβλητών πάνω από την έκφραση.

9η επιλογή: RequiresCallingbyObject_Reference,

Έκφραση τύπου requires η οποία, προδιαγράφει εάν η κλήση της μεθόδου έγινε με reference, δηλαδή εάν τα στοιχεία που θα μπούνε στην μέθοδο θα κρατήσουν την τιμή τους και στην έξοδο τους από αυτή. Στην περίπτωση αυτή θα πρέπει να δοθεί η μέθοδος με τον τρόπο που αναγράφεται, καθώς επίσης το όνομα της μεταβλητής και ο τύπος της.

10η επιλογή: AssertCounter_insideTableBoundaries

Έκφραση assert, η οποία τοποθετείται μέσα σε κάποια μέθοδο, πάνω από την εντολή for η οποία θα βγάζει μήνυμα στον χρήστη εάν ο μετρητής ξεπεράσει τα όρια του μεγέθους του πίνακα.

11η επιλογή: AssertWhile_Counters,

Έκφραση assert, η οποία τοποθετείται μέσα σε κάποια μέθοδο, πάνω από την εντολή while η οποία ελέγχει εάν ο μετρητής του βρόγχου αυξάνεται κατά ένα σε κάθε επανάληψη και εντοπίζει εάν ο βρόγχος έχει τέρμα και κατ' επέκταση δεν είναι ατέρμονας.

12η επιλογή: VariableTypes_insideTableTypes,

Έκφραση τύπου requires η οποία, προδιαγράφει ότι ο τύπος μιας μεταβλητής είναι κατανοητός από τον τύπο κάποιου πίνακα. Στην έκφραση αυτή θα πρέπει να δηλωθούν από πάνω τις οι δύο τύποι για να μπορέσει να τους επεξεργαστεί.

13η επιλογή: CheckingCounter_TobeInsideTableSize,

Έκφραση τύπου invariant η οποία, προδιαγράφει ότι μέσα στη μέθοδο η μεταβλητή που θα ορίσουμε θα κυμαίνεται από το μηδέν μέχρι τα όρια κάποιου πίνακα.

14η επιλογή: Invariant_CheckTableLength_CheckType,

Έκφραση τύπου invariant η οποία, προδιαγράφει ότι η μεταβλητή θα κυμαίνεται στα όρια του πίνακα και ότι ο τύπος της μεταβλητής θα είναι συμβατός με αυτόν του πίνακα έτσι ώστε να μην υπάρξει κάποιο λάθος.

15η επιλογή: Factorial

Έκφραση τύπου requires η οποία, προδιαγράφει μία μεταβλητή με βάση την τιμή που θα πάρει από το πρόγραμμα, ότι θα κυμαίνεται στα πλαίσια του παραγοντικού. Δηλαδή θα μπορεί να έχει τιμή από ένα μέχρι n, και το αποτέλεσμα δεν θα ξεπερνά μια προκαθορισμένη τιμή για να μην υπάρξει υπερχείλιση.

16η επιλογή: EnsuresTableHasValues,

Έκφραση τύπου ensures η οποία, προδιαγράφει ότι ο βρόγχος θα εκτελεστεί σωστά και το αποτέλεσμα της μεθόδου θα δώσει σε μια μεταβλητή την τιμή του πίνακα στο τελευταίο σημείο που ελέχθηκε τελευταία.

17η επιλογή: EnsuresFindMinValuesFromTable

Έκφραση τύπου ensures η οποία ορίζει ότι η θα ολοκληρωθεί βρόγχος στα πλαίσια του μήκους ενός πίνακα και για κάθε θέση στον πίνακα εάν η τιμή είναι μικρότερη ενός αριθμού, τότε ο μετρητής στη μέθοδο θα πρέπει να αυξάνεται κατά ένα. Σε αντίθετη περίπτωση η τιμή στη θέση του πίνακα θα πρέπει να πάρει την τιμή μηδέν.

Η έκφραση αυτή δίνεται μαζί με τον κώδικα της μεθόδου η οποία πρέπει να συμπληρωθεί για να δώσει απάντηση το πρόγραμμα.

18η επιλογή: EnsuresResultNotZero,

Έκφραση τύπου ensures η οποία, προδιαγράφει τη μέθοδο, στην οποία η απάντηση της μεθόδου δεν μπορεί να είναι μηδέν. Για να γίνει αυτό η έκφραση συνοδεύεται μαζί με την μέθοδο η οποία πρέπει να συμπληρωθεί κανονικά.

19η επιλογή: EnsuresInitializeTable

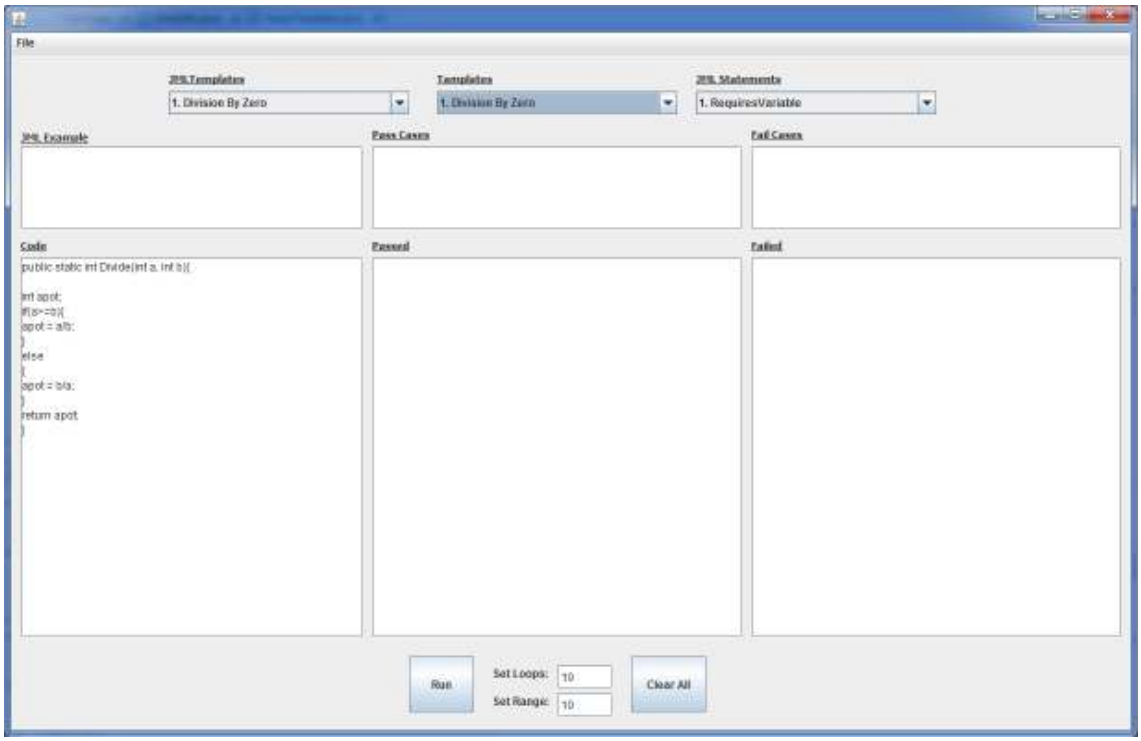
Έκφραση τύπου ensures η οποία, προδιαγράφει το αποτέλεσμα της μεθόδου να μην είναι null, χρησιμοποιώντας μια μέθοδο η οποία επιστρέφει το χειριστήριο του πίνακα που αρχικοποιήθηκε χρησιμοποιώντας μέγεθος που ορίστηκε από τον χρήστη.

20η επιλογή: EnsuresExceptions

Έκφραση τύπου ensures η οποία, προδιαγράφει το αποτέλεσμα της μεθόδου ότι έπιασε το exception. Η έκφραση αυτή, χρησιμοποιεί μια μέθοδο στην οποία προδιαγράφεται το αποτέλεσμα. Όπως η μεταβλητή που θα δοθεί έχει τιμή μηδέν τότε το αποτέλεσμα πρέπει να είναι η τιμή στην μεταβλητή r, ενώ σε αντίθετη περίπτωση το αποτέλεσμα αναμένεται να είναι η τιμή στην μεταβλητή r διαιρεμένη με το d.

5.5.3 Περιγραφή των Templates

Πιο κάτω φαίνεται το πώς εμφανίζονται τα Templates στην εφαρμογή. Τα Templates αυτά είναι οι ίδιες 15 περιπτώσεις που αναφέρθηκαν και παρουσιάστηκαν πριν, όμως χωρίς ενσωματωμένες τις JML εκφράσεις μέσα σε αυτά. Ο λόγος που δίνονται ανεξάρτητα είναι για να μπορεί ο χρήστης να εισάγει όποιες εκφράσεις θέλει χωρίς να έχει επιρροή από το σύστημα. Επίσης πρέπει να αναφέρουμε ότι όλες οι εκφράσεις που αναφέραμε στο προηγούμενο υποκεφάλαιο, μπορούν να εισαχθούν σε όποια γραμμή επιλέξει ο χρήστης άπλα τοποθετώντας τον δρομέα του σε κάποια γραμμή στον κώδικα και επιλογή κάποια από τις εκφράσεις που επιθυμεί.

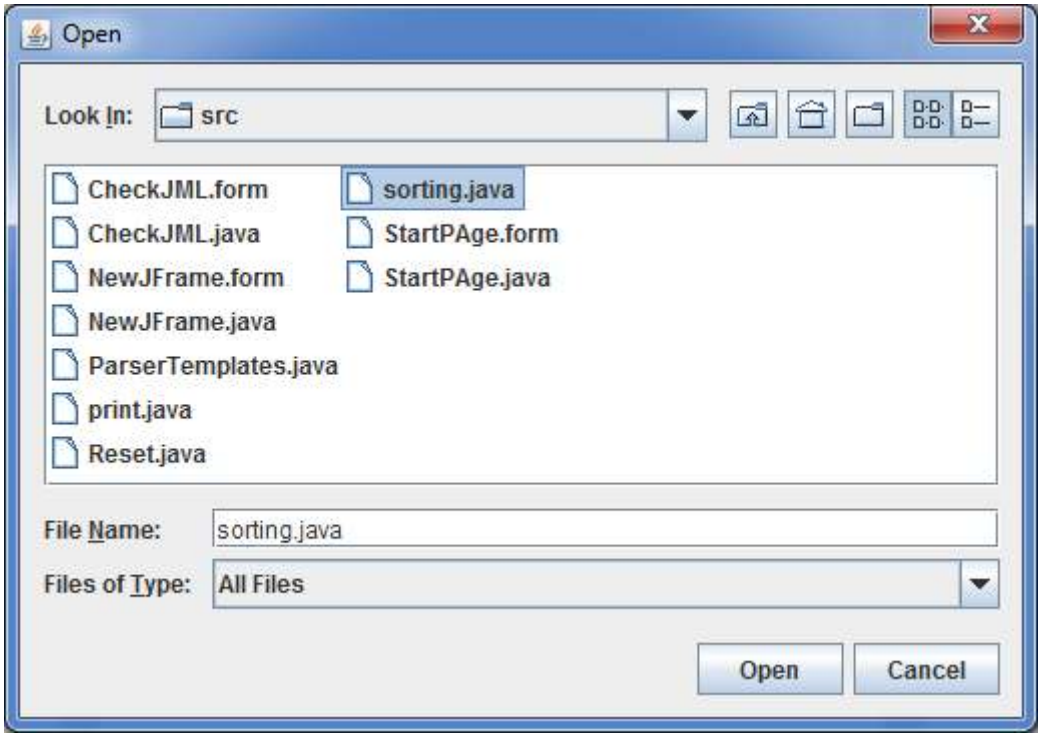


Εικόνα 5.23: Επιλογή 1ου JML Statement

Αν και υπάρχει η δυνατότητα εισαγωγής εκφράσεων ο χρήστης προτρέπεται να χρησιμοποιήσει εκφράσεις οι οποίες να ταιριάζουν με τα πρότυπα, για να υπάρχει μια λογική συνοχή κατά την εκτέλεση του λογισμικού.

5.5.4 Περιγραφή της εισαγωγής κώδικα

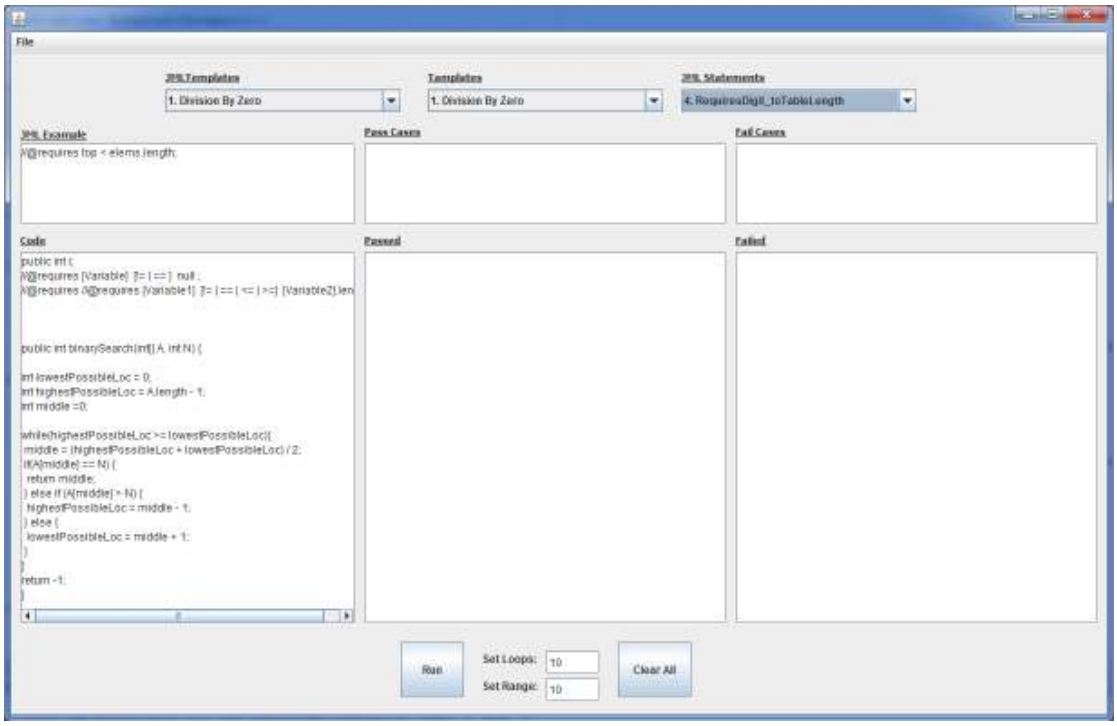
Το λογισμικό παρέχει την δυνατότητα εισαγωγής κώδικα στο πεδίο code, για να ευκολύνει τον χρήστη στην αλληλεπίδραση με το εξωτερικό περιβάλλον. Η εισαγωγή γίνεται από την επιλογή File και μετά Open. Το παράθυρο αναζήτησης θα εμφανιστεί και ο χρήστης θα μπορεί να κάνει την επιλογή του.



Εικόνα 5.24: Επιλογή 1ου JML Statement

Η εισαγωγή του κώδικα γίνεται κανονικά έτσι υπάρχει και η δυνατότητα εισαγωγής JML Statements από αυτά που παρέχονται στο λογισμικό. Και πάλι προτρέπεται ο χρήστης να ακολουθεί τους κανόνες και τη σύνταξη που υπάρχει στις εκφράσεις για να μπορεί το πρόγραμμα να παράγει ικανοποιητικά αποτελέσματα.

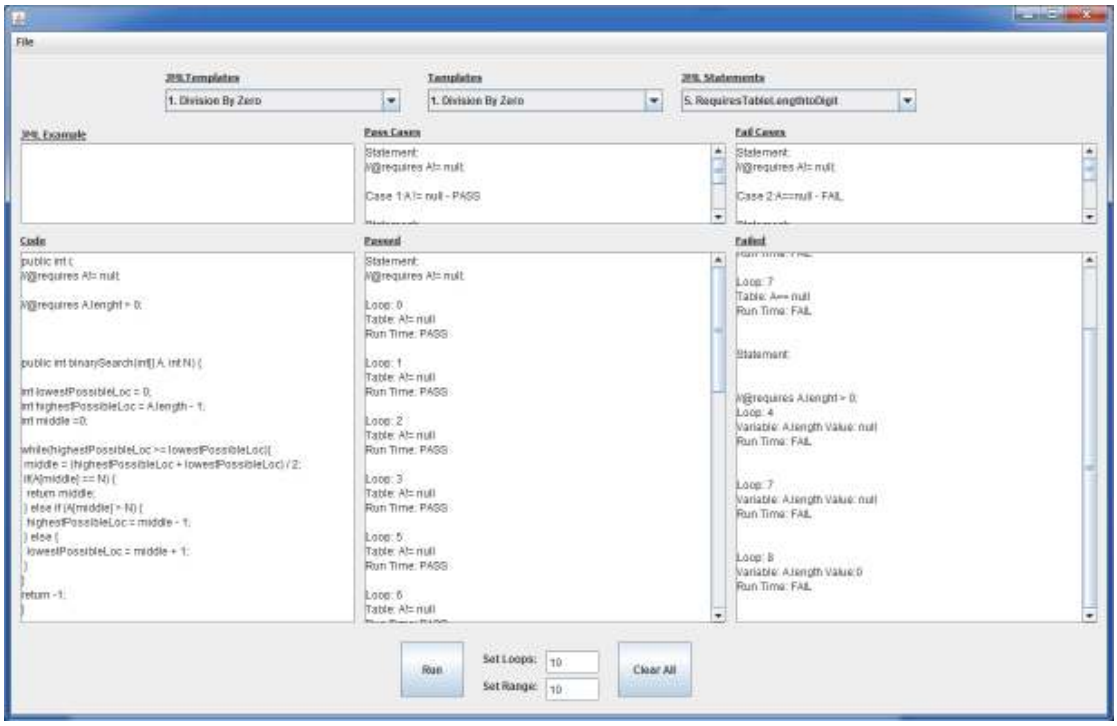
Πιο κάτω ακολουθεί ένα παράδειγμα εισαγωγής κώδικα στο λογισμικό, όπου στην συνέχεια το προδιαγράφουμε με δυο JML εκφράσεις και εκτελούμε το πρόγραμμα.



Εικόνα 5.25: Εισαγωγή δυο εκφράσεων JML στον κώδικα

Στην εικόνα 5.25 βλέπουμε ότι εισήχθηκε στο λογισμικό ένα αρχείο κώδικα, το οποίο υπήρχε στην μνήμη του υπολογιστή. Ο κώδικας χρησιμοποιεί ένα πίνακα ο οποίος αρχικοποιείται από τον προγραμματιστή. Ο πίνακας θα πρέπει να προδιαγραφεί σωστά έτσι ώστε το χειριστήριο του πίνακα να μην είναι null και το μέγεθος του να είναι μεγαλύτερο του μηδενός.

Σε αυτή την περίπτωση το προδιαγράψαμε με τις εκφράσεις: **RequiresNullHandle**, και **RequiresTableLengthtoDigit.** Οι δυο αυτές εκφράσεις περιμένουν τον προγραμματιστή να τις συμπληρώσει αναλόγως με βάση τις μεταβλητές του κώδικα. Η πρώτη έκφραση εισάγεται ως εξής: **//@requires [Variable] [!= | ==] null;** και διαμορφώνεται ως **//@requires A!= null; .** Στην περίπτωση αυτή προγράφουμε ότι το χειριστήριο που θα δοθεί δε θα έχει null τιμή. Το επόμενο JML statement που εισάγουμε είναι το: **//@requires [Variable].lenght [!= | == | <= | >=] [Digit] ;** το οποίο μετατρέπεται σε: **//@requires A.lenght > 0;** . Σε αυτή την περίπτωση η έκφραση αυτή προδιαγράφει ότι το μήκος του πίνακα θα είναι μεγαλύτερο του μηδενός και σε αντίθετη περίπτωση η έκφραση θα αποτύχει.



Εικόνα 5.26: Επεξεργασία των εκφράσεων JML, και εκτέλεση του προγράμματος

Αφού ολοκληρώσαμε την συγγραφεί των εκφράσεων JML με βάση την καθοδήγηση του λογισμικού, εκτελέσαμε το πρόγραμμα το οποίο δημιούργησε τις κλάσεις ισοδυναμίας για κάθε προδιαγραφή και παρήγαγε 10 Test Cases (TC) για κάθε μια από τις εκφράσεις που έχουν δοθεί.

Στην περίπτωση του JML statement //**@requires A!= null;** οι κλάσεις ισοδυναμίας που δημιουργήθηκαν ήταν δυο. Η περίπτωση η οποία το χειριστήριο του πίνακα A δεν είναι null και οι προδιαγραφές ικανοποιούνται και η άλλη κλάση που ορίζει ότι εάν το χειριστήριο έχει τιμή null τότε οι προδιαγραφές αποτυγχάνουν. Παρομοίως με την έκφραση //**@requires A.lenght > 0;** δημιουργούνται δυο κλάσεις ισοδυναμίας όπου η πρώτη που επαληθεύει τις προδιαγραφές ελέγχει εάν το μήκος του πίνακα είναι μεγαλύτερο του μηδενός, ενώ η δεύτερη που δεν επαληθεύει τις προδιαγραφές ελέγχει εάν το μήκος είναι μικρότερο ή ίσο του μηδενός.

Όπως βλέπουμε στην εικόνα 5.26, τα διάφορα Test Cases (TC), τα οποία επαλήθευσαν τις προδιαγραφές τυπώθηκαν στο αριστερό μέρος και όσα δεν τις επαλήθευσαν τυπώθηκαν στο δεξιό μέρος.

Κεφάλαιο 6

Συμπεράσματα

6.1 Εισαγωγή

Η διπλωματική αυτή εργασία είχε ως σκοπό την δημιουργία μιας εφαρμογής η οποία να έχει την δυνατότητα να ασκεί δυναμικό έλεγχο στις προδιαγραφές του πηγαίου κώδικα κάποιου λογισμικού με την εισαγωγή εκφράσεων JML. Δηλαδή να ελέγχει εάν το λογισμικό πληροί τις προδιαγραφές του, βρίσκοντας τις κλάσεις ισοδυναμίας στις εκφράσεις της JML και δίνοντας τιμές στο σύστημα, ελέγχοντας με αυτό τον τρόπο εάν τηρούνται οι προδιαγραφές που θέσαμε.

Όπως έχουμε δει η JML μπορεί να βοηθήσει στην γρήγορη αποσφαλμάτωση του κώδικα σε οποιοδήποτε στάδιο της ανάπτυξης μίας εφαρμογής. Η εφαρμογή που δημιουργήθηκε βασίστηκε στις προηγούμενες εργασίες που αφορούσαν το συγκεκριμένο θέμα και χρησιμοποιήθηκαν πρότυπα για ανίχνευση των λαθών ενσωματώνοντας JML εκφράσεις, για εύκολη και γρήγορη ανίχνευση των προγραμματιστικών λαθών που παρατηρούνται.

Τέλος, με βάση τις προδιαγραφές που εισάγονται στον κώδικα μπορούμε να παρατηρήσουμε τις περιπτώσεις που το λογισμικό πληροί τις προδιαγραφές που του έχουν δοθεί, αλλά και όλες τις περιπτώσεις που δεν πληρούνται. Στην δεύτερη περίπτωση που βρούμε ότι για κάποιο λόγο το λογισμικό μας αποτυγχάνει μπορούμε να κάνουμε αλλαγές στον κώδικα, και να ελέγξουμε ξανά τις προδιαγραφές του.

6.2 Συμπεράσματα

Το κύριο μέρος της διπλωματικής αυτής εργασίας, ήταν η υλοποίηση, του προγράμματος που έχει σκοπό τον δυναμικό έλεγχο ενός λογισμικού με τη χρήση εκφράσεων JML. Από την όλη δουλειά και έρευνα που έγινε, πάρθηκαν κάποια συμπεράσματα όσων αφορά την συμπεριφοριστική γλώσσα προδιαγραφών JML την ίδια την εφαρμογή αλλά και στα αποτελέσματα της.

Η συμπεριφοριστική γλώσσας προδιαγραφών JML, φάνηκε πολύ χρήσιμη στην όλη εργασία επειδή καθόριζε με ακρίβεια και σαφήνεια τις προδιαγραφές που χαρακτηρίζουν την αναμενόμενη συμπεριφορά σε κάποια ενότητα Java όπως και μία πιο αυστηρή τεκμηρίωση του ίδιου του κώδικα. Επίσης η γλώσσα λόγω της αλληλεπίδρασης της με την Java όπως φάνηκε έχει τη δυνατότητα να προσφέρει με υπάρχοντα εργαλεία την υποστήριξη της ίδιας της γλώσσας όσο και την προοπτική δημιουργίας εργαλείων για τις ανάγκες του κάθε προγραμματιστή.

Επίσης παρατηρήθηκε ότι τα διάφορα πρωτότυπα που χρησιμοποιήθηκαν στην εργασία ήταν πολύ βοηθητικά αλλά όπως ήταν αναμενόμενο δεν καλύπτουν όλες τις περιπτώσεις αναγκών και προβλημάτων που μπορεί να υπάρξουν. Παρόλα αυτά καλύπτουν τις πιο συνηθισμένες περιπτώσεις και ένας προγραμματιστής μπορεί να τα ενσωματώνει στον κώδικα του εύκολα και γρήγορα με μόνη αλλαγή τα ονόματα των μεταβλητών και ίσως κάποιες επιπλέον υποκειμενικές προδιαγραφές, που θα ορίσει ο ίδιος.

Γενικά κρίθηκε ότι είναι πολύ σημαντικό για ένα προγραμματιστή να ορίζει τις προδιαγραφές του συστήματος στον κώδικα, για να μπορεί να ελέγχει ανά πάσα στιγμή εάν η συμπεριφορά και τα αποτελέσματα που εξάγει το πρόγραμμα του είναι τα επιθυμητά. Παρατηρήθηκε επίσης ότι εάν οι προδιαγραφές του προγράμματος είναι γνωστές από την αρχή τότε το πρόγραμμα θα ακολουθήσει με σωστή πορεία μέχρι να υλοποιηθεί, ενώ σε αντίθετη περίπτωση ίσως να γίνουν λάθη και η διόρθωση αυτών των λαθών σε μεταγενέστερο στάδιο θα κοστίσει περισσότερο τόσο σε χρόνο όσο και σε χρήμα. Αυτό δεν είναι επιθυμητό από κανένα έτσι ο ορισμός των προδιαγραφών στο σύστημα από νωρίς μπορεί να βοηθήσει για να αποφευχθεί ένα τέτοιο γεγονός.

6.2.1 Πλεονεκτήματα και Μειονεκτήματα

Η δημιουργία αυτής της εφαρμογής έχει πλεονεκτήματα αλλά και μειονεκτήματα τόσο στον τρόπο που είναι υλοποιημένη αλλά και στον τρόπο που χρησιμοποιείται. Το λογισμικό αυτό έχει σχεδιαστεί με τέτοιο τρόπο έτσι ώστε ακόμα και κάποιος αρχάριος προγραμματιστής να μπορεί να μπορεί εύκολα να τεκμηριώσει αυστηρά την αναμενόμενη συμπεριφορά του προγράμματος του. Επίσης σε περίπτωση αναζήτησης των λαθών στον κώδικα να μπορεί εύκολα ένας προγραμματιστής να εισάγει αυτά τα πρότυπα μαζί έτσι ώστε να μπορεί να ελέγχει ταυτόχρονα όλες τις κύριες αιτίες λαθών στον κώδικα, παρά να ψάχνει στα τυφλά σε μεγάλα κομμάτια κώδικα για να βρει που μπορεί να είναι το λάθος.

Όπως έχουμε ήδη αναφέρει, για την κατασκευή του λογισμικού χρειάστηκε να δημιουργηθούν δυο είδη parser, το ένα που να μπορεί να ξεχωρίζει τις δηλωτικές εκφράσεις σε JML, και το άλλο να μπορεί να ξεχωρίζει τον κώδικα σε Java, όπου είναι απαραίτητο. Ο τρόπος λειτουργίας όμως των δυο αυτών parser, από μόνος του μας προξενεί κάποιους περιορισμούς αλλά ταυτόχρονα και ευελιξία. Συγκεκριμένα με τη δημιουργία parser, εξυπακούεται ότι το λογισμικό θα αναγνωρίζει εκφράσεις που συμπίπτουν με τις εκφράσεις που του έχουν δοθεί. Έτσι έχουμε το ρίσκο να δοθεί κάποιο κείμενο κώδικα από τον χρήστη και το λογισμικό να μην μπορέσει να το κατανοήσει. Από την άλλη μεριά υπάρχει και ένα μεγάλο πλεονέκτημα. Εφόσον το λογισμικό αναγνωρίσει τον κώδικα τότε έχουμε ελευθερία στην επεξεργασία του κώδικα αλλά και στην εξαγωγή αποτελεσμάτων.

Σε επέκταση του πιο πάνω, το λογισμικό έχει την δυνατότητα να αναγνωρίζει πλήρως τις μεταβλητές, τους αριθμούς τις δεσμευμένες λέξεις αλλά και τις ανισότητες που θα δοθούν προς ανάλυση. Παρόλα αυτά μπορεί να υπάρξουν περιπτώσεις όπου η αλλαγή κάποιας θέσης από αυτές τις λέξεις, ή και αλλαγή σε κάποια ανίσωση σε σημείο που δεν προβλέπεται πιθανόν να μην δώσει τα επιθυμητά αποτελέσματα. Ο λόγος που μπορεί να συμβαίνει αυτό είναι ότι το λογισμικό έχει βασιστεί στα πρότυπα που καλύπτουν γενικές περιπτώσεις και όχι σε όλο το φάσμα κώδικα που μπορεί να υπάρξει τόσο σε JML όσο και σε Java.

Τέλος ο τρόπος με τον οποίο διεξάχθηκε η υλοποίηση του λογισμικού από πλευράς κώδικα είναι αρκετά αφαιρετικός και επαναχρησιμοποιήσιμος, με αποτέλεσμα στο μέλλον να μπορεί να κατανοηθεί εύκολα, να συντηρηθεί αλλά και να επεκταθεί εάν κριθεί απαραίτητο.

6.3 Μελλοντική Εργασία

Για αυτή τη ατομική διπλωματική εργασία, μελετήθηκαν και υλοποιήθηκαν κάποια πρότυπα ανίχνευσης λαθών όπου το καθένα επιλύει ένα συγκεκριμένο πρόβλημα. Για μελλοντική εργασία θα μπορούσαν να βρεθούν και να εισαχθούν πιο πολλά και πιο πολύπλοκα πρότυπα, βάσει των οποίων να δημιουργηθεί η ευχέρεια προδιαγραφής πιο πολλών λειτουργιών.

Όπως ήδη γνωρίζουμε το λογισμικό δημιουργήθηκε κάνοντας χρήση κανονικών εκφράσεων για να μπορεί να καταλαβαίνει των κώδικα που του δίνεται. Θα μπορούσαν να προστεθούν περισσότερες κανονικές εκφράσεις ή και βελτίωση στις ήδη υπάρχουσες έτσι ώστε το πρόγραμμα να μπορεί να αναγνωρίζει πιο πολλά σημεία στον κώδικα και να επεξεργάζεται πιο πολλά δεδομένα.

Άλλο σημείο που θα μπορούσε να επεκταθεί είναι η ευελιξία του προγράμματος. Αυτή τη στιγμή υπάρχει περιορισμένη δυνατότητα για ποια σημεία μπορεί να κάνει αλλαγές ο χρήστης και να εξάγει αποτελέσματα. Έτσι θα ήταν καλό εάν το πρόγραμμα άφηνε τον χρήστη να επέμβει σε πιο πολλά σημεία μέσα στα πρότυπα που διατίθενται στο λογισμικό. Όπως κάποιες εκφράσεις που απαιτούν την εμφάνιση κάποιου συγκεκριμένου βρόγχου, ή κάποιες μεταβλητές και αριθμοί που μένουν σταθεροί λόγω της λειτουργικότητας του προτύπου.

Το λογισμικό όπως είδαμε κάνει χρήση μιας γραφικής διαπροσωπίας, η οποία δημιουργήθηκε αποκλειστικά για αυτό. Θα μπορούσε να τελειοποιηθεί η διαπροσωπία κάνοντας την πιο φιλική στο χρήστη και δημιουργώντας ένα παράρτημα επεξηγηματικών σχολίων για το πώς δουλεύουν οι λειτουργίες του λογισμικού.

Όπως έχει αναφερθεί, ο τρόπος με τον οποίο το λογισμικό βρίσκει και δίνει δεδομένα στο σύστημα είναι τυχαίος, μέσα στο φάσμα που καθορίζεται από τον χρήστη. Επίσης βρίσκονται και οι κλάσεις ισοδυναμίας για κάθε έκφραση που δίνει στον χρήστη μια πιο ακριβής εικόνα των αποτελεσμάτων που φαίνονται στην οθόνη. Όμως η αρχική ιδέα για την οποία δεν έγινε επιτυχής η εγκατάσταση στο εργαλείο, ήταν η χρήση γενετικών αλγορίθμων οι οποίοι θα ήταν υπεύθυνοι για να τροφοδοτούν το σύστημα με δεδομένα ελέγχου βάσει της λογικής που θα τους δοθεί. Οι δημιουργία αυτών των αλγορίθμων και η συνένωση τους με το λογισμικό αυτό θα έφερνε το λογισμικό σε μια ανώτερη βαθμίδα, αφού η κατανομή των τιμών στο πρόγραμμα θα γινόταν με πολύ πιο έξυπνο και αποδοτικό τρόπο.

Τέλος σαν μελλοντική εργασία θα ήταν ωφέλιμο να υλοποιηθούν έλεγχοι στο πρόγραμμα οι οποίοι να ενημερώνουν τον χρήστη για τυχών λάθος χρήση του προγράμματος, και να τον καθοδήγα πώς να διόρθωση το σφάλμα που βρέθηκε κατά τη χρήση.

Βιβλιογραφία

[1] Αργυρού Μαρίνος, ΕΛΕΓΧΟΣ ΠΡΟΔΙΑΓΡΑΦΩΝ ΛΟΓΙΣΜΙΚΟΥ ΣΕ ΕΠΙΠΕΔΟ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ JAVA MODELING LANGUAGE(JML), 2009

[2] Du, Wenliang and Mathur, Aditya P., "Categorization of Software Errors that led to Security Breaches",1998

[3] David Cok, Joe Kiniry & Erik Poll - ESC/Java2 & JML Tutorial

[4] Gary T. Leavens, Albert L. Baker, and Clyde Ruby, "JML: A Notation for Detailed Design",1999

[5] Gary T. Leavens, Yoonsik Cheon, "Design by Contract with JML", September 2006

[6] Gary T. Leavens, Erik Poll, Curtis Clifton, Yoonsik Cheon, Clyde Ruby, David Cok, Peter Müller, Joseph Kiniry, Patrice Chalin, and Daniel M. Zimmerman. JML Reference Manual (DRAFT), May 2008

[7] Iowa State University, Department of Computer Science, "Tools, examples, and documentation for the Java Modeling Language (JML)"

[8] Ιωάννου Κωνσταντίνος, ΕΛΕΓΧΟΣ ΠΡΟΔΙΑΓΡΑΦΩΝ ΛΟΓΙΣΜΙΚΟΥ ΣΕ ΕΠΙΠΕΔΟ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ JAVA MODELING LANGUAGE(JML), 2008

[9] K. Rustan M. Leino, Greg Nelson, and James B. Saxe, "ESC/Java User's Manual",2000

[10] Lilian Burdy, Yoonsik Cheon, David R. Cok, Michael D. Ernst, Joseph R. Kiniry, Gary T, "An overview of JML tools and applications", 2004

[11] Meyer B., “Design By Contract. The Eiffel Method”,1998

[12] Patrice Chalin, Joseph R. Kiniry, Gary T. Leavens, and Erik Poll, “Beyond Assertions: Advanced Specification and Verification with JML and ESC/Java2”, 2001

[13] Yoonsik Cheon, Myoung Yee Kim, and Ashaveena Perumandla, “A Complete Automation of Unit Testing for Java Programs”, 2005

[14] Ian Sommerville 2004 Software Engineering, 7th edition. Chapter 23

Παράρτημα Α

Κώδικες Υλοποίησης Λογισμικού

CheckJML.java

```
 * * * * * * * * * * * * * * * * * * * * * *
 * Author    : Panayiotis Christofi          *
 * Student Id: 879986                        *
 * Program   : Senior Project                *
 * * * * * * * * * * * * * * * * * * * * * *
```

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.regex.*;
import java.util.*;
import java.util.Iterator;
import java.util.Set;
import java.util.Enumeration;
import javax.swing.JFileChooser;
import javax.swing.text.Caret;

public class CheckJML extends javax.swing.JFrame {
public int range ;
public int glcounter;
   /** Creates new form CheckJML */
   public CheckJML() {
      initComponents();
   }
  public int intRandomInitialize() {
                  Random generator = new Random();
            int givenumber= generator.nextInt(range);
            return givenumber -5;
      }
  public float floatRandomInitialize() {
                  Random generator = new Random();
            float givenumber= generator.nextInt(range);
            return givenumber-5;
     }
  public double doubleRandomInitialize() {
                  Random generator = new Random();
```

```
            double givenumber= generator.nextInt(range);
            return givenumber-5;
    }
  public char charRandomInitialize() {
                Random generator = new Random();
            char givenumber= (char) generator.nextInt(range);
            return givenumber;
    }
public String StringRandomInitialize() {
                Random generator = new Random();
            int givenumber=  generator.nextInt(range);
            if(givenumber%10 == 1){
               return null;
            }
            return "notnull";
    }
  public int ArrayRandomInitializeSize() {
                Random generator = new Random();

            int givenumber=  generator.nextInt(range);
            givenumber= (int) (givenumber - 0.1 * range);
            if(givenumber == 0){
               return 0;
            }
            else if(givenumber <0){
               return -1;
            }
            return givenumber;
    }
  public void InitilizeTable(int size,ArrayList<Integer> table){
     Random generator = new Random();
     if(size <0){
        return ;
     }

     for(int i=0;i<size;i++){
        table.add(generator.nextInt(range));
     }
     return;
  }
 public String typeRandomInitialize() {
```

```
        Random generator = new Random();
        String randomtype = "int";
        int givenumber= generator.nextInt(8);
         switch (givenumber) {
           case 0:{randomtype= "int";}
           case 1:{randomtype=  "float";}
           case 2:{randomtype=  "double";}
           case 4:{randomtype=   "char";}
           case 5:{randomtype=  "Integer[]";}
           case 6:{randomtype=  "Double[]";}
           case 7:{randomtype=  "Float[]";}
           case 8:{randomtype=  "Object[]";}

         }
        String type= randomtype;
        return type;
}
 @SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jScrollPane3 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jComboBox1 = new javax.swing.JComboBox();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextPane2 = new javax.swing.JTextPane();
    jScrollPane4 = new javax.swing.JScrollPane();
    jTextPane3 = new javax.swing.JTextPane();
    jScrollPane6 = new javax.swing.JScrollPane();
    jTextArea3 = new javax.swing.JTextArea();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jScrollPane7 = new javax.swing.JScrollPane();
    jTextArea4 = new javax.swing.JTextArea();
    jLabel5 = new javax.swing.JLabel();
    jButton2 = new javax.swing.JButton();
    jComboBox2 = new javax.swing.JComboBox();
```

```
        jScrollPane8 = new javax.swing.JScrollPane();
        jTextArea5 = new javax.swing.JTextArea();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea6 = new javax.swing.JTextArea();
        jScrollPane9 = new javax.swing.JScrollPane();
        jTextArea7 = new javax.swing.JTextArea();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jComboBox3 = new javax.swing.JComboBox();
        fileChooser = new javax.swing.JFileChooser();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        Open = new javax.swing.JMenuItem();
        Exit = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jButton1.setText("Run");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(430, 650, 70, 60));

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane3.setViewportView(jTextArea1);

        getContentPane().add(jScrollPane3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(390, 220, 400, 410));

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"1. Division By Zero", "2. Array Index Out Of Range", "3. String Index Out Of Range",
```

```
"4. Array Store Exception", "5. Dereferencing Null", "6. Stack Overflow Area", "7. Unsorted Table", "8. Infinite Loop due to mathematical operator", "9. Error due to Logical operator", "10. Negative Array Size", "11. SizeOfParallelTables", "12. DiffTypeEquationException", "13. StaticInitialisation", "14. CatchingException", "15. CallbyValueOrReference", " " }));
        jComboBox1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox1ActionPerformed(evt);
            }
        });
        getContentPane().add(jComboBox1, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 40, 260, -1));

        jLabel1.setText("Set Loops:");
        getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(520, 660, 60, -1));

        jLabel2.setText("Set Range:");
        getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(520, 690, -1, -1));

        jTextPane2.setText("10");
        jScrollPane1.setViewportView(jTextPane2);

        getContentPane().add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(590, 660, 60, -1));

        jTextPane3.setText("10");
        jScrollPane4.setViewportView(jTextPane3);

        getContentPane().add(jScrollPane4, new org.netbeans.lib.awtextra.AbsoluteConstraints(590, 690, 60, -1));

        jTextArea3.setColumns(20);
        jTextArea3.setRows(5);
        jScrollPane6.setViewportView(jTextArea3);

        getContentPane().add(jScrollPane6, new org.netbeans.lib.awtextra.AbsoluteConstraints(800, 220, 400, 410));

        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel3.setText("<html><u>Passed</u></html>\"");
```

```
        getContentPane().add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(390, 200, -1, -1));

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel4.setText("<html><u>Failed</u></html>\"");
        getContentPane().add(jLabel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(800, 200, 40, -1));

        jTextArea4.setColumns(20);
        jTextArea4.setRows(5);
        jScrollPane7.setViewportView(jTextArea4);

        getContentPane().add(jScrollPane7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(390, 100, 400, 90));

        jLabel5.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel5.setText("<html><u>Fail Cases</u></html>\"");
        getContentPane().add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(800, 80, 120, -1));

        jButton2.setText("Clear All");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(670, 650, -1, 60));

        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"1. RequiresVariable", "2. RequiresNullHandle", "3. RequiresDigit", "4.
RequiresDigit_toTableLength", "5. RequiresTableLengthtoDigit", "6.
RequiresSortedTable", "7. RequiresSameTableLength", "8. RequiresSameType", "9.
RequiresCallingbyObject_Reference", "10. AssertCounter_insideTableBoundaries",
"11. AssertWhile_Counters", "12. VariableTypes_insideTableTypes", "13.
CheckingCounter_TobeInsideTableSize", "14.
Invariant_CheckTableLength_CheckType", "15. Factorial", "16.
EnsuresTableHasValues", "17. EnsuresFindMinValuesFromTable", "18.
EnsuresResultNotZero", "19. EnsuresInitializeTable", "20. EnsuresExceptions", " " }));
        jComboBox2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox2ActionPerformed(evt);
            }
```

```
        });
        getContentPane().add(jComboBox2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(740, 40, 260, -1));

        jTextArea5.setColumns(20);
        jTextArea5.setRows(5);
        jScrollPane8.setViewportView(jTextArea5);

        getContentPane().add(jScrollPane8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 220, 370, 410));

        jTextArea6.setColumns(20);
        jTextArea6.setRows(5);
        jScrollPane2.setViewportView(jTextArea6);

        getContentPane().add(jScrollPane2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 100, 370, 90));

        jTextArea7.setColumns(20);
        jTextArea7.setRows(5);
        jScrollPane9.setViewportView(jTextArea7);

        getContentPane().add(jScrollPane9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(800, 100, 400, 90));

        jLabel6.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel6.setText("<html><u>Pass Cases</u></html>\"");
        getContentPane().add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(390, 80, -1, -1));

        jLabel7.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel7.setText("<html><u>JML Example</u></html>\""); // NOI18N
        getContentPane().add(jLabel7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, -1, 20));

        jLabel8.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel8.setText("<html><u>Code</u></html>\"");
        getContentPane().add(jLabel8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 200, -1, -1));

        jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"1. Division By Zero", "2. Array Index Out Of Range", "3. String Index Out Of Range",
```

```
"4. Array Store Exception", "5. Dereferencing Null", "6. Stack Overflow Area", "7. Unsorted Table", "8. Infinite Loop due to mathematical operator", "9. Error due to Logical operator", "10. Negative Array Size", "11. SizeOfParallelTables", "12. DiffTypeEquationException", "13. StaticInitialisation", "14. CatchingException", "15. CallbyValueOrReference", " " }));
        jComboBox3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox3ActionPerformed(evt);
            }
        });
        getContentPane().add(jComboBox3, new org.netbeans.lib.awtextra.AbsoluteConstraints(460, 40, 260, -1));

        fileChooser.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                fileChooserActionPerformed(evt);
            }
        });
        getContentPane().add(fileChooser, new org.netbeans.lib.awtextra.AbsoluteConstraints(820, 40, 50, 0));

        jLabel9.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel9.setText("<html><u>JML Statements</u></html>\"");
        getContentPane().add(jLabel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(740, 20, -1, -1));

        jLabel10.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel10.setText("<html><u>Tamplates</u></html>\"");
        getContentPane().add(jLabel10, new org.netbeans.lib.awtextra.AbsoluteConstraints(460, 20, -1, -1));

        jLabel11.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel11.setText("<html><u>JMLTemplates</u></html>\"");
        getContentPane().add(jLabel11, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 20, -1, -1));
        getContentPane().add(jLabel12, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, -20, 1210, 750));

        jMenu1.setText("File");

        Open.setText("Open");
        Open.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                OpenActionPerformed(evt);
            }
        });
        jMenu1.add(Open);

        Exit.setText("Exit");
        Exit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ExitActionPerformed(evt);
            }
        });
        jMenu1.add(Exit);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String srange = jTextPane3.getText();
        range = Integer.parseInt(srange );
        jTextArea5.getText();
        String text2 = jTextPane2.getText();
        jTextArea1.setText("");                                    jTextArea3.setText("");
jTextArea4.setText("");jTextArea6.setText("");jTextArea7.setText("");
        String text = null;
        glcounter = Integer.parseInt(text2 );

        text =jTextArea5.getText();
        Hashtable hashint = new Hashtable();
        Hashtable hashfloat = new Hashtable();
        Hashtable hashdouble = new Hashtable();
        Hashtable hashchar = new Hashtable();
        Hashtable hashstring = new Hashtable();
        int[] patterns = new int[34];
        for(int cv2=0;cv2<patterns.length;cv2++)
            patterns[cv2]=0;
        int GLOBALCOUNTER = glcounter;
```

```
int loop1=0; int loop2=0;
ArrayList<Integer> passorfail28 = new ArrayList<Integer>(); //pass=1 fail =0 //#0
ArrayList<String> statement28 = new ArrayList<String>();//#0

ArrayList<Integer> passorfail = new ArrayList<Integer>(); //pass=1 fail =0 //#25
ArrayList<Integer> numbera = new ArrayList<Integer>();//#25
ArrayList<Integer> numberb = new ArrayList<Integer>();//#25
ArrayList<Integer> result = new ArrayList<Integer>();//#25
ArrayList<String> name25 = new ArrayList<String>();//#25
ArrayList<String> statement25 = new ArrayList<String>();//#25
ArrayList<String> printcase25 = new ArrayList<String>();//#25


ArrayList<String> statement01 = new ArrayList<String>();//#01
ArrayList<ArrayList<Integer>> numbers01 = new ArrayList<ArrayList<Integer>>();//#01
ArrayList<ArrayList<Integer>> passorfail01 = new ArrayList<ArrayList<Integer>>();//#01
ArrayList<String> name01 = new ArrayList<String>();//#01
ArrayList<String> digit01 = new ArrayList<String>();//#01
ArrayList<String> anisosi01 = new ArrayList<String>();//#01

ArrayList<ArrayList<Integer>> tables09 = new ArrayList<ArrayList<Integer>>();//#09
ArrayList<Integer> tablesize09 = new ArrayList<Integer>();//#09

ArrayList<String> statement02=new ArrayList<String>();//#2
ArrayList<String> name02=new ArrayList<String>();//#2
ArrayList<String> names02=new ArrayList<String>();//#2
ArrayList<ArrayList<Integer>> passorfail02 = new ArrayList<ArrayList<Integer>>();//#02
ArrayList<String> names03=new ArrayList<String>();//#2
ArrayList<String> statement03 = new ArrayList<String>();//#03
ArrayList<Integer> passorfail03 = new ArrayList<Integer>(); //pass=0 fail =1 //#26
ArrayList<String> statement26 = new ArrayList<String>();//#03
ArrayList<Integer> passorfail26 = new ArrayList<Integer>(); //pass=0 fail =1 //#26
ArrayList<String> name26 = new ArrayList<String>();//#26

ArrayList<String> statement05=new ArrayList<String>();//#5
```

```
        ArrayList<Integer> variable05 = new ArrayList<Integer>();//#5
        ArrayList<Integer> tablesize05 = new ArrayList<Integer>();//#5
        int compare05 =0;
        ArrayList<Integer> passorfail05 = new ArrayList<Integer>(); //pass=1 fail =0
//#05
        ArrayList<String> names05 = new ArrayList<String>(); //pass=1 fail =0  //#05
         ArrayList<String> names06 = new ArrayList<String>(); //pass=1 fail =0  //#05

        ArrayList<String> statement06=new ArrayList<String>();//#5
        ArrayList<Integer> passorfail06 = new ArrayList<Integer>(); //pass=1 fail =0
//#06

        ArrayList<String> names07 = new ArrayList<String>();//#07
        ArrayList<String> statement07 = new ArrayList<String>();//#07
        ArrayList<String> type07 = new ArrayList<String>();//#07
        int compare07 =0;

        ArrayList<String> names08 = new ArrayList<String>();//#08
        ArrayList<String> statement08 = new ArrayList<String>();//#08
        ArrayList<Integer> passorfail08 = new ArrayList<Integer>();//#08

        ArrayList<String> names15 = new ArrayList<String>();//#15
        ArrayList<String> statement15 = new ArrayList<String>();//#15
        ArrayList<Integer> passorfail15 = new ArrayList<Integer>();//#15
        ArrayList<Integer> number15 = new ArrayList<Integer>();//#15

        ArrayList<String> names16 = new ArrayList<String>();//#16
        ArrayList<String> statement16 = new ArrayList<String>();//#16
        ArrayList<Integer> passorfail16 = new ArrayList<Integer>();//#16
        ArrayList<Integer> number16 = new ArrayList<Integer>();//#16


        ArrayList<String> statement17 = new ArrayList<String>();//#17
        ArrayList<Integer> passorfail17 = new ArrayList<Integer>();//#17
        ArrayList<String> names17 = new ArrayList<String>();//#17

        ArrayList<String> names18 = new ArrayList<String>();//#18
        ArrayList<String> statement18 = new ArrayList<String>();//#18
        ArrayList<Integer> passorfail18 = new ArrayList<Integer>();//#18
        ArrayList<Integer> number18 = new ArrayList<Integer>();//#18
```

```
int numofstatements =0;

ArrayList<String> statement19 = new ArrayList<String>();//#19
ArrayList<Integer> passorfail19 = new ArrayList<Integer>();//#19
ArrayList<String> names19 = new ArrayList<String>();//#19

ArrayList<String> statement22 = new ArrayList<String>();//#22
ArrayList<Integer> passorfail22 = new ArrayList<Integer>();//#22
ArrayList<String> names22 = new ArrayList<String>();//#22
ArrayList<Integer> result22 = new ArrayList<Integer>();//#22

ArrayList<String> statement27 = new ArrayList<String>();//#27
ArrayList<Integer> passorfail27 = new ArrayList<Integer>();//#27
ArrayList<String> anisosis27 = new ArrayList<String>();//#27


ArrayList<String> names29 = new ArrayList<String>();//#29
ArrayList<ArrayList<Integer>> tables29 = new ArrayList<ArrayList<Integer>>();//#29
ArrayList<String> statement29 = new ArrayList<String>();//#29
ArrayList<Integer> passorfail29 = new ArrayList<Integer>();//#29
ArrayList<ArrayList<Integer>> tablesize29 = new ArrayList<ArrayList<Integer>>();//#29

ArrayList<String> names30 = new ArrayList<String>();//#30
ArrayList<String> statement30 = new ArrayList<String>();//#30
ArrayList<Integer> passorfail30 = new ArrayList<Integer>();//#30

String anisosi29 = null;

ArrayList<String> names31 = new ArrayList<String>();//#31
ArrayList<ArrayList<Integer>> variables31 = new ArrayList<ArrayList<Integer>>();//#31
ArrayList<String> statement31 = new ArrayList<String>();//#31
ArrayList<Integer> passorfail31 = new ArrayList<Integer>();//#31

ArrayList<ArrayList<String>> names32 = new ArrayList<ArrayList<String>>();//#32
ArrayList<ArrayList<String>> types32 = new ArrayList<ArrayList<String>>();//#32
ArrayList<String> statement32 = new ArrayList<String>();//#32
```

```
        ArrayList<ArrayList<Integer>> passorfail32 = new
ArrayList<ArrayList<Integer>>();//#32

        ArrayList<String> names33 = new ArrayList<String>();//#33
        ArrayList<String> result33 = new ArrayList<String>();//#33
        ArrayList<String> statement33 = new ArrayList<String>();//#33
        ArrayList<Integer> passorfail33 = new ArrayList<Integer>();//#33

        Reset rs = new Reset();
        ArrayList<String> mresultdigit_anisosi = new ArrayList<String>();
        ArrayList<Integer> mresultdigit_num = new ArrayList<Integer>();
        ArrayList<String> mresultnull_anisosi = new ArrayList<String>();
        ArrayList<String> mcheckvar_anisosi = new ArrayList<String>();

    //   text = "//@ensures \result!=null";
    // String[] tokens=  text.split(";");




        Pattern start = Pattern.compile("(//@ *|/\\*@ *)");
        Pattern start2 = Pattern.compile("/\\*(@ *.*[\\r\\n]*.*)+"); //.;.*^\\*/      .\\r\\n
        Pattern start3 = Pattern.compile("/\\*(@ *.*[\\r\\n]*.*)+"); //.;.*^\\*/      .\\r\\n
        Pattern end = Pattern.compile(";");
           Pattern white = Pattern.compile("[\\s]*");
        ////////////////////////////////
        Pattern whitenewline = Pattern.compile("[[^.][\\s]]*");

           Pattern anisotites = Pattern.compile("(!=|==|>=|<=|<|>)");
        Pattern andor = Pattern.compile("(&&|\\|\\|)");
        Pattern counting = Pattern.compile("(\\+\\+|--)");
        Pattern operator = Pattern.compile("(\\+|-|\\*|\\/)");
        Pattern backslash = Pattern.compile("\\\\");
        Pattern slash = Pattern.compile("[/]");
        Pattern variable = Pattern.compile("[a-zA-Z0-9]+");
        Pattern variable2 = Pattern.compile("([a-zA-Z0-9]+)");
        Pattern digits = Pattern.compile("([\\d]+)");
        Pattern vars = Pattern.compile("(int|float|double|String|byte|char|Object)");
        Pattern varsornothing =
Pattern.compile("(int|float|double|String|byte|char|Object|Integer|)");
        Pattern variableornothing = Pattern.compile("([a-zA-Z0-9]*)");
```

```
Pattern brackets = Pattern.compile("(\\[[\\s]*\\]|)");
Pattern publics = Pattern.compile("(public|private|)");
Pattern statics = Pattern.compile("(static|final|)");
Pattern comma = Pattern.compile("(,|)");
Pattern brackets2 = Pattern.compile("(\\[\\]| )");
Pattern voids = Pattern.compile("(int|float|double|String|byte|char|Object|void)");

Pattern function = Pattern.compile(white + "" + publics + white + statics + white
+voids + white+brackets+white + variable + white + "\\(" + white + varsornothing+
white+brackets + white + variableornothing + white+ brackets+ white+comma+white +
varsornothing+ white+brackets + white + variableornothing + white+ brackets+ white+
"\\)" + white + "\\{");

//@ensures \result != null;   #28
  Pattern resultnull = Pattern.compile(white+ ""+ start + "ensures" +white +
backslash+ "result" + white +anisotites+ white +"null" + white + end);
//@requires a!=0;   #1
Pattern checkvar = Pattern.compile(white + "" + start +"requires"+ white
+variable2 + white + anisotites + white + digits + white + end);
//@requires a!=null;          #2
Pattern checkvarnull = Pattern.compile(white+ "" + start +"requires"+ white
+variable2 + white + anisotites + white + "null" + white + end);
//@ensures (\forall int i; 0 <= i && i < a.length; c!==a[a.length - 1]);
//@ensures (\forall int i; 0 <= i && i < a.length; c==a[a.length - 1]);          #3
Pattern arrayforvalue = Pattern.compile(white + "" + start +"ensures"+ white +
"\\(" +white + backslash+ "forall" + white + "int" + white+variable2+ white +";"+ white
+ digits + white + anisotites + white + variable + white + andor +white + variable +
white + anisotites + white + variable + ".length" + white + ";" + white + variable2 +
white + anisotites + white + variable2+white+"\\["+white+variable+".length" + white +
"-" + white + "1"+"\\]" + white + "\\)" + white +end );
//@ensures (\forall int i; 0 <= i && i < a.length; c!=null);          #4
Pattern arrayforvaluenull = Pattern.compile(white + "" + start +"ensures"+ white +
"\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +";"+ white
+ digits + white + anisotites + white + variable + white + andor +white + variable +
white + anisotites + white + variable + ".length" + white + ";" + white + variable +
white + anisotites + white + "null" + white + "\\)" + white +end );
//@invariant 0 <= top && top <= elems.length && \typeof(elems) ==
\type(Object[]);     #5
Pattern checktype = Pattern.compile(white + "" + start +"invariant"+ white +
digits+ white  + anisotites+ white  + variable2 + white + andor+ white + variable +
white + anisotites + white + variable2 + ".length" + white + andor + white +backslash+
"typeof\\(elems\\)" + white + anisotites + white + backslash +
"type\\("+variable2+"\\[\\]\\)" + white + end );
//@requires top < elems.length;     #6
```

```
Pattern checklenght = Pattern.compile(white + "" + start +"requires"+ white +variable2 + white + anisotites + white + variable2 +".length" + white + end);

/*@requires \typeof(o) <: \elemtype(\typeof(elems));@*/     //#7

Pattern checkvartypewitharray = Pattern.compile(white + "" + start +"requires"+ white + backslash + "typeof\\(" + variable2 + "\\)" + white + "<:" + white + backslash + "elemtype\\(" + backslash +"typeof\\(" + variable2 + "\\)\\)" + white + ";" );


//@invariant 0<=size && size <= elements.length;                    #8
//@invariant 0<=size && size <= elements.length;

Pattern checksizeandarray = Pattern.compile(white + "" + start +"invariant"+white+digits+white+anisotites+white+variable2+white+andor+ white+ variable + white + anisotites + white + variable2 + ".length" + white + end );

/*@spec_public@*/ //int c;     #9

Pattern definevariable = Pattern.compile(white + "" + "/\\*@" + white + "spec_public" + white + "@\\*/" + white + vars + white + variable2+ white);

/*@spec_public@*/ //int a[];  #10

Pattern definetables = Pattern.compile(white + "" + "/\\*@" + white + "spec_public" + white + "@\\*/" + white + vars + white + variable+ "\\[\\]"+ white );

/*@non_null@*/ //int elements[]; #11

Pattern tablesnotnull = Pattern.compile(white + "" + "/\\*@" + white + "non_null" + white + "@\\*/" + white + vars + white + variable+ "\\[\\]"+ white + end);

//@assignable \nothing;    #12

Pattern assignablenothing = Pattern.compile(white + "" + start + white + "assignable" +white + backslash+ "nothing" + white + end );

//@assignable c;    #13

Pattern assignablevar = Pattern.compile(white + "" + start + white + "assignable" +white + variable+ white + end );

/*@spec_public non_null@*/ //Object[] elems;   #14

Pattern definetablesnonnull = Pattern.compile(white + "" + "/\\*@" + white + "spec_public"+white +"non_null" + white + "@\\*/" + white + variable2+"\\[\\]" + white + variable2 + white + end);

/*@requires 1 <= n;*/                  // #15

Pattern checknumvar = Pattern.compile(white + "" + start +"requires"+ white +digits + white + anisotites + white + variable2 + white + end);

/*@ensures \result== (\product int i ; 1<= i && i<=n ;i); &&\result<6227020800 #16 */

Pattern factorial = Pattern.compile(white + "" + start2 + "ensures"  +white + backslash+ "result" + white +anisotites+ white +  "\\(" +white + backslash+ "product" + white + "int" + white+variable2+ white +";"+ white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white + variable2 + white + end + white + variable + "\\)" + white + end + white + andor + backslash + "result" + white + anisotites + digits );

//@requires A.lenght > 0;        #17
```

```
Pattern checktable = Pattern.compile(white + "" + start +"requires"+ white +variable2+".lenght" + white + anisotites + white + digits + white + end);

//  @requires n<13;      #18

Pattern checkvar2 = Pattern.compile(white + "" + start2 +"requires"+ white +variable2 + white + anisotites + white + digits + white + end);

//@requires(\forall int i, j;0 <= i && i < j && j < A.length; A[i] <= A[j]);      #19

Pattern sorted = Pattern.compile(white + "" + start +"requires"+ white + "\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +"," + white + variable + white + ";" + white + digits + white + anisotites + white + variable + white + andor+ white + variable + white + anisotites +white + variable + white + andor + white + variable + white + anisotites + white + variable + ".length" + white + ";" + white + variable + "\\[" + white + variable2 + white + "\\]" + white + anisotites + white + variable2 + white + "\\[" + white + variable2 + white + "\\]" + white + "\\)" +white + end);

//@ensures (\forall int i ;0 <= i && i < A.length ==> \old(A[i]== N)) ? (\result==i) : (\result== -1);   #20

Pattern arrayresultornot = Pattern.compile(white + "" + start +"ensures"+ white + "\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +";"+ white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white + variable + ".length" + white + "==>" + white + backslash  + white + "old" + white + "\\(" + white + variable  + "\\[" + white + variable + white + "\\]"  + white + anisotites  + white + variable  + white + "\\)" + white +"\\)" + white + "\\?" + white + "\\(" + white +backslash + "result"+ white + anisotites + white + variable + white +"\\)" + white + ":" + white + "\\("  + white +backslash  + "result" + white + anisotites  + white +"-"+ digits + white +"\\)" + white +end );

//  @assignable ia[*];    #21

Pattern assigntable = Pattern.compile(white + "" + start2 + white + "assignable" +white + variable+"\\["+white+ "\\*" + "\\]"+ white + end );

// @ensures(\forall int i; 0<=i && i <ia.length ==> (\old(ia[i]) >0) && (\forall int j; 0<=j && j<i ==> (\old(ia[j])>5))) ? (count==count+1) : (ia[i]== \old(ia[i])); @*/  #22

Pattern resultarray = Pattern.compile(white + "" + start +"ensures"+ white + "\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +";"+ white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white + variable + ".length" + white + " ==>" + white  + "\\("+backslash+"forall" + white + "int" + white + variable + white + end + white + digits + white + anisotites + white + variable + white + andor + white + variable +white +
anisotites+white+variable+white+anisotites+white+anisotites+white+"\\("+backslash+" old\\("+variable+"\\["+variable+"\\]"+"\\)"+anisotites+
white+digits+"\\)\\)\\)"+white+"\\?"+white+"\\("+white+variable2+white+anisotites+white+variable+white+operator +white + digits+"\\)" + white + ":" + white+ "\\(" +variable2 +"\\["+variable+"\\]"+white+anisotites+white +digits + white+"\\)"+ white+end);

/*@ invariant k+m==0; */  //#23

Pattern invariantquation = Pattern.compile(white + "" + start +"invariant"+white+ variable + white + operator + white + variable + anisotites + white + digits + end );
```

```
//@ensures k==\old(k)-1 && m==\old(m)+1;  //24

Pattern oldandnewvalue = Pattern.compile(white + "" + start +"ensures"+ white + variable + white + anisotites + white + backslash + "old" + white + "\\(" + white + variable + white + "\\)" + white + operator + white + digits + white + andor + white + variable + anisotites + white + backslash + "old" + white + "\\(" + white + variable + white + "\\)" + white+ operator +white + digits + white + end) ;

//@ensures \result != 0;  //#25

Pattern resultdigit = Pattern.compile(white+ ""+start + "ensures"  +white + backslash+ "result" + white +anisotites+ white +digits+ white + end);

//@assert(\forall int i,j; 0<=i && i<=j && j<a.length; i<a.length);  //#26

Pattern assert2 =  Pattern.compile(""+ start +"assert"+ white + "\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +","+white+variable+white+end+white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white+ variable+ white + andor + white + variable + white + anisotites + white+ variable + ".length" + white + end +white + variable2 + white +  anisotites + white + variable2 + ".length" + white + "\\)" +white+ end);

/*@assert(\forall int i; 0<=i && i<(\old(number)-\old(num)); number==\old(number)-1);@*/ //#27

Pattern whilenumcheck = Pattern.compile(white + "" + start +"assert"+ white + "\\(" +white + backslash+ "forall" + white + "int" + white+variable+ white +";"+ white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white + "\\(" + white + backslash + "old\\(" + white +variable2+"\\)"+white+"\\-"+white+backslash+"old\\("+variable2+"\\)\\);"+ white + variable+white+anisotites + white + backslash + "old\\(" + white +variable2+"\\)"+white+"\\-"+digits+white+"\\)"+white+end );

//@requires a.length == b.length ;

Pattern checktablewithtablelength = Pattern.compile(white + "" + start +"requires"+ white +variable2+".lenght" + white + anisotites + white + variable2+".lenght" + white + end);

/*@requires \typeof(num) <: \typeof(c);@*/ //#30

Pattern checkdiftype = Pattern.compile(white + "" + start +"requires"+ white + backslash +"typeof" +white+"\\("+ white+variable2+white+"\\)" +white+ "<:"+ white + backslash+"typeof" +white + "\\(" + white + variable2 + white + "\\)" + white + end);

//@ensures \result == ((d == 0) ? \old(r) : \old(r) / d );            #31

Pattern resultexceptions = Pattern.compile(white+ ""+ start + "ensures" +white + backslash+ "result" + white +anisotites+ white + "\\(" + white + "\\(" + white + variable2 + white + anisotites + white + digits + white + "\\)" + white + "\\?" + white +backslash +"old"+white + "\\("+white+ variable2+ white + "\\)" + white + ":" + white + backslash + "old" + "\\(" + white + variable2 + white + "\\)" + white + operator + white + variable2 + white + "\\)"+white + end );

/*@requires \typeof(numa) <: \type(Object[]);@*/   //#32

Pattern checkdifobject = Pattern.compile(white + "" + start +"requires"+ white + backslash +"typeof" +white+"\\("+ white+variable2+white+"\\)" +white+ "<:"+ white + backslash+"type" +white + "\\(" + white + vars + white + "\\[" + white + "\\]" + white + "\\)" + white + end);
```

```
        //@ensures r1 && r2 && r3 && r4;  #33
         Pattern checkstatic = Pattern.compile(white + "" + start +"ensures"+ white + variable2 + white + andor + white + variable2 + white + andor + white + variable2 + white + andor + white + variable2 + white + end );
        int globalcounter=0;
      // ArrayList<Enumeration> enumarray = new  ArrayList<Enumeration>();

        Enumeration e = hashint.keys();
         //ResetableIterator itr = (ResetableIterator) hashint.keys();




        while(globalcounter<GLOBALCOUNTER){

     //e= null;e = hashint.keys();
       e = hashint.keys();
    loop2=0;

        jTextArea5.getText();
        text =jTextArea5.getText();

        Matcher mstart = start.matcher(text);
        Matcher mresultdigit = resultdigit.matcher(text);
        Matcher mresultnull = resultnull.matcher(text);
        Matcher mcheckvar = checkvar.matcher(text);
        Matcher marrayforvalue = arrayforvalue.matcher(text);
        Matcher mcheckvarnull = checkvarnull.matcher(text);
        Matcher marrayforvaluenull = arrayforvaluenull.matcher(text);
        Matcher mchecktype = checktype.matcher(text);
        Matcher mchecklenght = checklenght.matcher(text);
        Matcher mcheckvartypewitharray = checkvartypewitharray.matcher(text);
        Matcher mchecksizeandarray = checksizeandarray.matcher(text);
        Matcher mdefinevariable = definevariable.matcher(text);
        Matcher mdefinetables = definetables.matcher(text);
        Matcher mtablesnotnull = tablesnotnull.matcher(text);
        Matcher massignablenothing = assignablenothing.matcher(text);
        Matcher massignablevar = assignablevar.matcher(text);
        Matcher mdefinetablesnonnull = definetablesnonnull.matcher(text);
        Matcher mchecknumvar = checknumvar.matcher(text);
        Matcher mfactorial = factorial.matcher(text);
```

```
        Matcher mchecktable = checktable.matcher(text);
        Matcher mcheckvar2 = checkvar2.matcher(text);
        Matcher msorted = sorted.matcher(text);
        Matcher marrayresultornot = arrayresultornot.matcher(text);
        Matcher massigntable = assigntable.matcher(text);
        Matcher mresultarray = resultarray.matcher(text);
        Matcher minvariantquation = invariantquation.matcher(text);
        Matcher moldandnewvalue = oldandnewvalue.matcher(text);
        Matcher mfunction = function.matcher(text);
        Matcher mwhitenewline = whitenewline.matcher(text);
        Matcher massert2 = assert2.matcher(text);
        Matcher mwhilenumcheck = whilenumcheck.matcher(text);
        Matcher mchecktablewithtablelength = checktablewithtablelength.matcher(text);
        Matcher mcheckdiftype = checkdiftype.matcher(text);
        Matcher mresultexceptions = resultexceptions.matcher(text);
        Matcher mcheckdifobject = checkdifobject.matcher(text);
        Matcher mcheckstatic = checkstatic.matcher(text);
          String typeofvariable1=null;



 // if(globalcounter==0)
 while(mfunction.find()) {

   typeofvariable1= mfunction.group(5);
   String bracketsfortable11= mfunction.group(6);
   String nameofvariable1= mfunction.group(7);
   String bracketsfortable112= mfunction.group(8);
   String typeofvariable2= mfunction.group(10);
   String bracketsfortable21= mfunction.group(11);
   String nameofvariable2= mfunction.group(12);
   String bracketsfortable22= mfunction.group(13);
  //2 variables
 // if(typeofvariable2!=null){
     if((typeofvariable1.equals("int")||
typeofvariable1.equals("int")||typeofvariable1.equals("float")||typeofvariable1.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object")) &&
     (typeofvariable2.equals("int")||
typeofvariable2.equals("int")||typeofvariable2.equals("float")||typeofvariable2.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable2.equals("byte")||
typeofvariable2.equals("char")||typeofvariable2.equals("Object"))
```

```
   && bracketsfortable11.equals("") && bracketsfortable112.equals("") &&
bracketsfortable21.equals("") && bracketsfortable22.equals("") ){
      if( typeofvariable1.equals("int")){
         hashint.put(nameofvariable1, intRandomInitialize());
      }else if (typeofvariable1.equals("float")){
         hashfloat.put(nameofvariable1, floatRandomInitialize());
      }else if (typeofvariable1.equals("double")){
         hashdouble.put(nameofvariable1, doubleRandomInitialize());
      }else if (typeofvariable1.equals("String")){
         hashstring.put(nameofvariable1, StringRandomInitialize());
      }else if (typeofvariable1.equals("char")){
         hashchar.put(nameofvariable1, charRandomInitialize());
      }

      if( typeofvariable2.equals("int")){
          hashint.put(nameofvariable2, intRandomInitialize());
      }else if (typeofvariable2.equals("float")){
         hashfloat.put(nameofvariable2, floatRandomInitialize());
      }else if (typeofvariable2.equals("double")){
         hashdouble.put(nameofvariable2, doubleRandomInitialize());
      }else if (typeofvariable2.equals("String")){
         hashstring.put(nameofvariable2, StringRandomInitialize());
      }else if (typeofvariable2.equals("char")){
         hashchar.put(nameofvariable2, charRandomInitialize());
      }
   }else                                                    if((typeofvariable1.equals("int")||
typeofvariable1.equals("float")||typeofvariable1.equals("double")||typeofvariable1.equal
s("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object")) &&
    (typeofvariable2.equals("int")||
typeofvariable2.equals("int")||typeofvariable2.equals("float")||typeofvariable2.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable2.equals("byte")||
typeofvariable2.equals("char")||typeofvariable2.equals("Object"))
    && bracketsfortable11.equals("") && bracketsfortable112.equals("[]") &&
bracketsfortable21.equals("") && bracketsfortable22.equals("[]") ){
      if( typeofvariable1.equals("int")){
         hashint.put(nameofvariable1, intRandomInitialize());
      }else if (typeofvariable1.equals("float")){
         hashfloat.put(nameofvariable1, floatRandomInitialize());
      }else if (typeofvariable1.equals("double")){
         hashdouble.put(nameofvariable1, doubleRandomInitialize());
      }else if (typeofvariable1.equals("String")){
```

```
        hashstring.put(nameofvariable1, StringRandomInitialize());
      }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, charRandomInitialize());
      }

      if( typeofvariable2.equals("int")){
         hashint.put(nameofvariable2, intRandomInitialize());
      }else if (typeofvariable2.equals("float")){
        hashfloat.put(nameofvariable2, floatRandomInitialize());
      }else if (typeofvariable2.equals("double")){
        hashdouble.put(nameofvariable2, doubleRandomInitialize());
      }else if (typeofvariable2.equals("String")){
        hashstring.put(nameofvariable2, StringRandomInitialize());
      }else if (typeofvariable2.equals("char")){
        hashchar.put(nameofvariable2, charRandomInitialize());
      }
    }
    //int[] A, in N

     else                                                    if((typeofvariable1.equals("int")||
typeofvariable1.equals("int")||typeofvariable1.equals("float")||typeofvariable1.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object")) &&
     (typeofvariable2.equals("int")||
typeofvariable2.equals("int")||typeofvariable2.equals("float")||typeofvariable2.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable2.equals("byte")||
typeofvariable2.equals("char")||typeofvariable2.equals("Object"))
     && bracketsfortable11.equals("[]") && bracketsfortable112.equals("") &&
bracketsfortable21.equals("") && bracketsfortable22.equals("") ){
      /* if( typeofvariable1.equals("int")){
        hashint.put(nameofvariable1, intRandomInitialize(10));
      }else if (typeofvariable1.equals("float")){
        hashfloat.put(nameofvariable1, floatRandomInitialize(10));
      }else if (typeofvariable1.equals("double")){
        hashdouble.put(nameofvariable1, doubleRandomInitialize(10));
      }else if (typeofvariable1.equals("String")){
        hashstring.put(nameofvariable1, StringRandomInitialize(10));
      }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, charRandomInitialize(10));
      }*/

      if( typeofvariable2.equals("int")){
```

```
          hashint.put(nameofvariable2, intRandomInitialize());
      }else if (typeofvariable2.equals("float")){
          hashfloat.put(nameofvariable2, floatRandomInitialize());
      }else if (typeofvariable2.equals("double")){
          hashdouble.put(nameofvariable2, doubleRandomInitialize());
      }else if (typeofvariable2.equals("String")){
          hashstring.put(nameofvariable2, StringRandomInitialize());
      }else if (typeofvariable2.equals("char")){
          hashchar.put(nameofvariable2, charRandomInitialize());
      }
    }
    //int n
    else                    if((                    typeofvariable1.equals("int")||
typeofvariable1.equals("int")||typeofvariable1.equals("float")||typeofvariable1.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object") )
    && bracketsfortable11.equals("") && bracketsfortable112.equals("") ){
       if( typeofvariable1.equals("int")){
          hashint.put(nameofvariable1, intRandomInitialize());
      }else if (typeofvariable1.equals("float")){
          hashfloat.put(nameofvariable1, floatRandomInitialize());
      }else if (typeofvariable1.equals("double")){
          hashdouble.put(nameofvariable1, doubleRandomInitialize());
      }else if (typeofvariable1.equals("String")){
          hashstring.put(nameofvariable1, StringRandomInitialize());
      }else if (typeofvariable1.equals("char")){
          hashchar.put(nameofvariable1, charRandomInitialize());
      }
    }
 //}//int ai[]
    /*
      else                    if((                    typeofvariable1.equals("int")||
typeofvariable1.equals("int")||typeofvariable1.equals("float")||typeofvariable1.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object") )
    && bracketsfortable11.equals("") && bracketsfortable112.equals("[]") ){
       if( typeofvariable1.equals("int")){

          hashint.put(nameofvariable1, ArrayRandomInitializeSize());
      }else if (typeofvariable1.equals("float")){
          hashfloat.put(nameofvariable1, ArrayRandomInitializeSize());
      }else if (typeofvariable1.equals("double")){
```

```
        hashdouble.put(nameofvariable1, ArrayRandomInitializeSize());
      }else if (typeofvariable1.equals("String")){
        hashstring.put(nameofvariable1, ArrayRandomInitializeSize());
      }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, ArrayRandomInitializeSize());
      }
    }//int[] input
       */
        //Integer o
    else if(( typeofvariable1.equals("Integer") )
    && bracketsfortable11.equals("") && bracketsfortable112.equals("") ){
      if( typeofvariable1.equals("int")){
        hashint.put(nameofvariable1, intRandomInitialize());
      }else if (typeofvariable1.equals("float")){
        hashfloat.put(nameofvariable1, floatRandomInitialize());
      }else if (typeofvariable1.equals("double")){
        hashdouble.put(nameofvariable1, doubleRandomInitialize());
      }else if (typeofvariable1.equals("String")){
        hashstring.put(nameofvariable1, StringRandomInitialize());
      }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, charRandomInitialize());
      }
    }
   else if(( typeofvariable1.equals("int")||
typeofvariable1.equals("int")||typeofvariable1.equals("float")||typeofvariable1.equals("d
ouble")||typeofvariable1.equals("String")||typeofvariable1.equals("byte")||
typeofvariable1.equals("char")||typeofvariable1.equals("Object") )
    && bracketsfortable11.equals("[]") && bracketsfortable112.equals("") ){
      /* if( typeofvariable1.equals("int")){
        hashint.put(nameofvariable1, intRandomInitialize(10));
      }else if (typeofvariable1.equals("float")){
        hashfloat.put(nameofvariable1, floatRandomInitialize(10));
      }else if (typeofvariable1.equals("double")){
        hashdouble.put(nameofvariable1, doubleRandomInitialize(10));
      }else if (typeofvariable1.equals("String")){
        hashstring.put(nameofvariable1, StringRandomInitialize(10));
      }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, charRandomInitialize(10));
      }*/
    }
   // ()
```

```
    else if(( typeofvariable1.equals("") )
   && bracketsfortable11.equals("") && bracketsfortable112.equals("") ){
     /* if( typeofvariable1.equals("int")){
        hashint.put(nameofvariable1, intRandomInitialize(10));
     }else if (typeofvariable1.equals("float")){
        hashfloat.put(nameofvariable1, floatRandomInitialize(10));
     }else if (typeofvariable1.equals("double")){
        hashdouble.put(nameofvariable1, doubleRandomInitialize(10));
     }else if (typeofvariable1.equals("String")){
        hashstring.put(nameofvariable1, StringRandomInitialize(10));
     }else if (typeofvariable1.equals("char")){
        hashchar.put(nameofvariable1, charRandomInitialize(10));
     }*/
   }
 }
//enumarray.add(hashint.keys() );
 e = hashint.keys();
String s = null;

while(  e.hasMoreElements() ){
   s= e.nextElement().toString();



}

//@requires a!=0;
//#1 start
 boolean tf = false;
       while(mcheckvar.find()) {
            patterns[1]=1;
          int counter=0;
           String variab = null;String statement = null;

e = hashint.keys();
if(!e.hasMoreElements() || tf==true){
   while(counter<=loop2){
   tf=true;
   variab = mcheckvar.group(2);
   hashint.put(variab, intRandomInitialize());
    if(globalcounter == 0 && counter == loop2){
```

```
        statement = mcheckvar.group(1) + "requires " + variab + " "    +
mcheckvar.group(3) + " " + mcheckvar.group(4) ;
        statement01.add(statement);
        digit01.add(mcheckvar.group(4));
        anisosi01.add(mcheckvar.group(3));
   if(numofstatements <=loop2){
                    numbers01.add(new ArrayList<Integer>());
                    passorfail01.add(new ArrayList<Integer>());
                    numofstatements++;
                    name01.add(variab);

                }
    }
   counter++;
   }
}else{

              e = hashint.keys();
           while(counter<=loop2){
             variab = mcheckvar.group(2);
              hashint.put(variab, intRandomInitialize());
             if(globalcounter == 0 && counter == loop2){
                statement = mcheckvar.group(1) + "requires " + variab + " "    +
mcheckvar.group(3) + " " + mcheckvar.group(4) ;
                statement01.add(statement);
                digit01.add(mcheckvar.group(4));
                anisosi01.add(mcheckvar.group(3));
                if(numofstatements <=loop2){
                   numbers01.add(new ArrayList<Integer>());
                   passorfail01.add(new ArrayList<Integer>());
                   numofstatements++;
                   name01.add(variab);
                }
             }
             counter++;
           }
}


        //variab                =                                        Integer.parseInt(
hashint.get(enumarray.get(globalcounter).nextElement().toString()).toString() );
```

```
          numbers01.get(loop2).add((Integer) hashint.get(variab));//sto proto array exo b, sto deftero a (sto proto array exo last variable, sto last proto)
          String anisosi = mcheckvar.group(3);
          int arithmos =  Integer.parseInt(mcheckvar.group(4));


        if(anisosi.equals("!=")){
          if(arithmos == Integer.parseInt( hashint.get(variab).toString() )){
            passorfail01.get(loop2).add(0);
           }
          else{

          passorfail01.get(loop2).add(1);
           }
        }else  if(anisosi.equals("==")){
          if(arithmos != Integer.parseInt( hashint.get(variab).toString() )){
            passorfail01.get(loop2).add(0);

          }else{passorfail01.get(loop2).add(1);
           }
        }else if(anisosi.equals(">")){
          if(arithmos < Integer.parseInt( hashint.get(variab).toString() )){
              passorfail01.get(loop2).add(1);

          }else{
              passorfail01.get(loop2).add(0);
            }
        }else if(anisosi.equals("<")){
          if(arithmos > Integer.parseInt( hashint.get(variab).toString() )){
              passorfail01.get(loop2).add(1);

          }else{
              passorfail01.get(loop2).add(0);
             }
        }else if(anisosi.equals("<=")){
          if(arithmos <= Integer.parseInt( hashint.get(variab).toString() )){
              passorfail01.get(loop2).add(1);

          }else{
              passorfail01.get(loop2).add(0);
             }
```

```
        }else if(anisosi.equals(">=")){
           if(arithmos >= Integer.parseInt( hashint.get(variab).toString() )){
               passorfail01.get(loop2).add(1);

           }else{
               passorfail01.get(loop2).add(0);
              }
        }
         loop2++;



      }
   //@requires a!=null;
   //#02 start
   loop2=0;
 while(mcheckvarnull.find()) {

    patterns[2]=2;
   int counter=0;
   String variab = null;String statement = null;
   e          =          hashint.keys();hashint.put(          mcheckvarnull.group(2),
ArrayRandomInitializeSize());
  // if(hashint.get(Integer)){ }
   while(counter<=loop2){

      variab = mcheckvarnull.group(2);

      if(globalcounter == 0 && counter == loop2){
         statement  =mcheckvarnull.group();  //mcheckvarnull.group(1)  +  "requires  "  +
variab + " "  + mcheckvarnull.group(2) + " null";
         names02.add(mcheckvarnull.group(2));   names02.add(mcheckvarnull.group(3));
names02.add("null");
         statement02.add(statement);
         if(numofstatements <=loop2){
            //numbers01.add(new ArrayList<Integer>());
            passorfail02.add(new ArrayList<Integer>());
            numofstatements++;
            name02.add(variab);
         }
      }
      counter++;
```

```
    }
//     numbers01.get(loop2).add((Integer) hashint.get(variab));//sto proto array exo b, sto deftero a (sto proto array exo last variable, sto last proto)
  //  String anisosi = mcheckvar.group(2);


     e = hashint.keys();
   String tablename = null;
   //if(globalcounter ==0)


   while(  e.hasMoreElements() ){
 tablename= e.nextElement().toString();
if(tablename.equals(mcheckvarnull.group(2))){

 tables09.add(new ArrayList<Integer>());
       int tablesize = Integer.parseInt(hashint.get(tablename).toString());
       tablesize09.add(tablesize);
    if(tablesize <0){
   passorfail02.get(loop2).add(0);InitilizeTable(0, tables09.get(tables09.size() -1 ));
    }else{passorfail02.get(loop2).add(1);
       InitilizeTable(tablesize, tables09.get(tables09.size() -1 ));}
   }
   }

    loop2++;
 }
 //#03 starts //@ensures (\forall int i; 0 <= i && i < a.length; c==a[i]);
     //0 <= i && i < a.length
     //C == a[i]
 while(marrayforvalue.find()) {
    patterns[3]=3;
     e = hashint.keys();


    Pattern forstatement = Pattern.compile(white + "" + "for" + white + "\\(" + white + vars + white + variable + white + "=" + white + digits + white + end + white + variable + white + anisotites + white + variable + ".length" + white + end + white + variable2 + "\\+\\+" + white + "\\)" + whitenewline + "\\{"  + whitenewline + variable + white + "=" + white + variable + white + "\\[" + white + variable + white + "\\]" + white + end + whitenewline + "\\}");
    Matcher mforstatement = forstatement.matcher(text);
```

```
if(tables09.size() !=0)
    while(mforstatement.find()) {//2=i, 3=0, <==4, 7=c,  8= ==, 9=a, 6= <   //i=0, 0=1 ,
<==2, c=3, ===4, a=5,< =6


        if(globalcounter == 0 ){
           names03.add(marrayforvalue.group(2));
names03.add(marrayforvalue.group(3)); names03.add(marrayforvalue.group(4));
           names03.add(marrayforvalue.group(7));
names03.add(marrayforvalue.group(8));
names03.add(marrayforvalue.group(9));names03.add(marrayforvalue.group(6));
          String statement =marrayforvalue.group(); //mcheckvarnull.group(1) + "requires
" + variab + " "  + mcheckvarnull.group(2) + " null";
          statement03.add(statement);}
        if(mforstatement.group(3).equals("<") && tablesize09.get(globalcounter) > 0){
           passorfail03.add(1);
        }else{
           passorfail03.add(0);
        }
    }
else{


    if(globalcounter == glcounter-1)
    if(tables09.size() ==0){
          String statement =marrayforvalue.group(); //mcheckvarnull.group(1) + "requires
" + variab + " "  + mcheckvarnull.group(2) + " null";
          statement03.add(statement);
           names03.add(marrayforvalue.group(2));
names03.add(marrayforvalue.group(3)); names03.add(marrayforvalue.group(4));
           names03.add(marrayforvalue.group(7));
names03.add(marrayforvalue.group(8));
names03.add(marrayforvalue.group(9));names03.add(marrayforvalue.group(6));
        completenumbers(tables09,glcounter, tablesize09);
          while(mforstatement.find()) {


        for(int cv03 = 0; cv03 <tablesize09.size(); cv03++ ){
        if(mforstatement.group(3).equals("<")&& tablesize09.get(cv03) > 0 ){
           passorfail03.add(1);
        }else{
           passorfail03.add(0);
        }
        }
    }
```

```
    }
}
 }
 while(marrayforvaluenull.find()) {

 }
   //#05 starts
   //@invariant 0 <= top && top <= elems.length && \typeof(elems) == \type(Object[]);     #5
//Print4 : 0<= top top <= elems.length - pass
//Print2: Elems == type of object  - pass

 while(mchecktype.find()) {
     //jTextArea2.append("" + mchecktype.groupCount() + mchecktype.group(7)+ "  4 = " + mchecktype.group(4) + " 7 = " + mchecktype.group(7) + " 10= " + mchecktype.group(10) );
    patterns[5]=5;
     e = hashint.keys();
     if(globalcounter == 0 ){

     names05.add(mchecktype.group(2)); //0. 0 //1. top //2. <=! //3. elems! //4. <=! //5. ==! //6. object!
    names05.add(mchecktype.group(4));
    names05.add(mchecktype.group(3));
    names05.add(mchecktype.group(7));
    names05.add(mchecktype.group(6));
      names05.add(mchecktype.group(9));
    names05.add(mchecktype.group(10));

    String statement =mchecktype.group();
    statement05.add(statement);
   while(mdefinetablesnonnull.find()) {
       if(globalcounter == 0 ){
         // names05.add(mdefinetablesnonnull.group(1)); //2. elems type
       }
      if(mchecktype.group(10).equals(mdefinetablesnonnull.group(1)) ){
         compare05 =1;
      }else{
         compare05=0;
      }
    }
```

```
   }
    variable05.add(intRandomInitialize());
    tablesize05.add(ArrayRandomInitializeSize());
    if(compare05 == 1 && (variable05.get(globalcounter) < tablesize05.get(globalcounter) ) && variable05.get(globalcounter)>=0 && tablesize05.get(globalcounter)>=0 )
       passorfail05.add(1);
    else
       passorfail05.add(0);
    if(globalcounter == (glcounter-1))
    hashint.put(mchecktype.group(4), 1);//apla vazw to variable top mesa

 }
 //#06 starts
 //@requires top < elems.length;
 while(mchecklenght.find()) {
  patterns[6]=6;
   if(glcounter==1 ||globalcounter == (glcounter-1)){
      if(variable05.size()==0 && tablesize05.size() ==0){
          names06.add(mchecklenght.group(2)); //0. top!
          names06.add(mchecklenght.group(4)); //1. elems!
          names06.add(mchecklenght.group(3)); //2. <!
         for(int cv006=0;cv006 < glcounter;cv006++){

           variable05.add(intRandomInitialize());
            tablesize05.add(ArrayRandomInitializeSize());}
      }
   }
      e = hashint.keys();
     if(globalcounter == 0 || globalcounter == (glcounter-1) ){
        if(globalcounter == 0){
          String statement =mchecklenght.group();
          statement06.add(statement);
 names06.add(mchecklenght.group(2)); //0. top!
          names06.add(mchecklenght.group(4)); //1. elems!
          names06.add(mchecklenght.group(3)); //2. <!

        }
        if(glcounter==1 ||globalcounter == (glcounter-1)){
         // if(hashint.get(mchecklenght.group(2)) != null){
             for(int cv06 =0; cv06 < tablesize05.size(); cv06++){
```

```
                if(variable05.get(cv06) < tablesize05.get(cv06) &&
variable05.get(cv06)>=0 &&  tablesize05.get(cv06)>=0){
                    passorfail06.add(1);
                }else{
                    passorfail06.add(0);
                }
               }
              //}
          }
      }
 }
 //#07 starts
 /*@requires \typeof(o) <: \elemtype(\typeof(elems));@*/
 while(mcheckvartypewitharray.find()) {
 patterns[7]=7;
 e = hashint.keys();

 if(globalcounter == 0 ){
 statement07.add(mcheckvartypewitharray.group());}
mdefinetablesnonnull.reset();
while(mdefinetablesnonnull.find()) {
    if(globalcounter == 0 ){

        names07.add(mcheckvartypewitharray.group(2)); //name of variable "o" at slot 0
        names07.add(mcheckvartypewitharray.group(3)); //name of variable "elems" at
slot 1
        type07.add(typeofvariable1);//2. o type (slot 1)
        type07.add(mdefinetablesnonnull.group(1)); //2. elems type (slot 1)
    }

   if(mdefinetablesnonnull.group(1).equals(typeofvariable1) ||
mdefinetablesnonnull.group(1).equals("Object") && (typeofvariable1.equals("Integer")
||  typeofvariable1.equals("Char") ||typeofvariable1.equals("Double")
||typeofvariable1.equals("Float"))  ){
       compare07 =1;
   }else{
        compare07 =0;
   }

 }
 //if(typeofvariable1 == Integer){
```

```
// }
 }
//@invariant 0<=size && size <= elements.length;        #8
 while(mchecksizeandarray.find()) {


 patterns[8]=8;
 e = hashint.keys();

 if(globalcounter == 0 ){
 statement08.add(mchecksizeandarray.group());
 names08.add(mchecksizeandarray.group(4));//size se thesi:0
 names08.add(mchecksizeandarray.group(7));//elems se thesi:1

names08.add(mchecksizeandarray.group(2));names08.add(mchecksizeandarray.group(3
));names08.add(mchecksizeandarray.group(6));//size=0, elems=1 , 0=2 , <= = 3i , <= =
4i


 }

  //Pattern forstatement = Pattern.compile(white + "" + "for" + white + "\\(" + white +
vars + white + variable + white + "=" + white + digits + white + end + white + variable
+ white + anisotites + white + variable + ".length" + white + end + white + variable2 +
"\\+\\+" + white + "\\)" + whitenewline + "\\{" );
  //Matcher mforstatement = forstatement.matcher(text);
 // mforstatement.reset();
Pattern sizeinitialize = Pattern.compile(white + "" + white + variable2 + white + "=" +
white + variable2 + ".length" + white + end  +whitenewline + variable2 + white + "=" +
white + "new" + white + "int" + white+ "\\[" + white + variable2 + white + "\\]" + white
+ end);
Matcher msizeinitialize = sizeinitialize.matcher(text);
msizeinitialize.reset();

while(msizeinitialize.find() && tablesize09.size() !=0) {

    if(passorfail02.get(0).get(globalcounter) == 0){
        passorfail08.add(0);
   }else          if(passorfail02.get(0).get(globalcounter)          ==          1          &&
msizeinitialize.group(1).equals(msizeinitialize.group(4)) ){

         passorfail08.add(1);
```

```
    }else{
         passorfail08.add(0);
    }
  }

  if(tables09.size() ==0 && globalcounter == glcounter-1){
   completenumbers(tables09,glcounter, tablesize09);

   for(int cv08=0; cv08<glcounter-1;cv08++) {
    if(tablesize09.get(cv08) < 0){
        passorfail08.add(0);
    }else if(tablesize09.get(cv08) >= 1){
         passorfail08.add(1);
    }
   }
  }


   //  size = input.length;
 //elements = new int[size];
 //System.arraycopy(input, 0, elements, 0, size);

 }
//#09 start
///*@spec_public@*/int a[];

while(mdefinevariable.find()) {
   //  patterns[9]=9;


//Initialize a[] with its current size



}

 while(mdefinetables.find()) {

 }
 while(mtablesnotnull.find()) {
```

```
}
while(massignablenothing.find()) {

}
while(massignablevar.find()) {

}
mdefinetablesnonnull.reset();
while(mdefinetablesnonnull.find()) {

}
/*@requires 1 <= n;*/                    // #15
while(mchecknumvar.find()) {

patterns[15]=15;
int num=0;
 e = hashint.keys();
if(globalcounter == 0 ){
statement15.add(mchecknumvar.group());

names15.add(mchecknumvar.group(4));
names15.add(mchecknumvar.group(2));
names15.add(mchecknumvar.group(3));   //size se thesi:0
}

if( hashint.get(mchecknumvar.group(4)) != null){
  num=Integer.parseInt( hashint.get(mchecknumvar.group(4)).toString() );
  number15.add(num);
  if( Integer.parseInt(mchecknumvar.group(2).toString()) < num )
     passorfail15.add(1);
  else
     passorfail15.add(0);
}else if(globalcounter ==glcounter -1){

    for(int cv015=0;cv015<glcounter;cv015++){
       number15.add(intRandomInitialize());
        if( number15.get(cv015) < num )
          passorfail15.add(1);
       else
           passorfail15.add(0);
    }
```

```
}

}
 /*@ensures \result== (\product int i ; 1<= i && i<=n ;i); &&\result<6227020800 #16 */
long factornum =0;
while(mfactorial.find()) {



patterns[16]=16;
Pattern factorial2 = Pattern.compile("@ensures"  +white + backslash+ "result" + white +anisotites+ white +   "\\(" +white + backslash+ "product" + white + "int" + white+variable+ white +";"+ white + digits + white + anisotites + white + variable + white + andor +white + variable + white + anisotites + white + variable + white + end + white + variable + "\\)" + white + end + white + andor + backslash + "result" + white + anisotites + digits );
Matcher mfactorial2 = factorial2.matcher(mfactorial.group());
while(mfactorial2.find())
{factornum=Long.parseLong(mfactorial2.group(8).toString());
if(globalcounter == 0 ){




statement16.add(mfactorial2.group());
//names18.add(mcheckvar3.group(1));   //size se thesi:0

names16.add(mfactorial.group(3));names16.add(mfactorial.group(4));names16.add(mfactorial.group(5));names16.add(mfactorial.group(7));  //  3=i  4=1    5<=    7<=  9<  8=n 12=6227020800
names16.add(mfactorial.group(9));names16.add(mfactorial.group(8));names16.add(mfactorial.group(11));// 0=i 1=1  2<=  3<= 4< 5=n 6=6227020800


}

if(number15.size() <= globalcounter)
   number15.add(intRandomInitialize());

int ni = number15.get(globalcounter);

if(ni >12){
   passorfail16.add(0); number16.add(0);
}else if(ni < 1){
```

```
    passorfail16.add(-1); number16.add(0);
 }else{

    int i=1;
    int f=1;
  while (i<ni) {
             i = i + 1;
             f = f * i;
            }

    if(f >= factornum){
    passorfail16.add(0);number16.add(f); }
    else{
    passorfail16.add(1) ;
    number16.add(f);}
 }
 }
 //if(globalcounter ==0)
// jTextArea2.append(mfactorial.group());
 }
  //@requires A.lenght > 0;        #17
 while(mchecktable.find()) {

  patterns[17]=17;
 int num=0;
 e = hashint.keys();
 if(globalcounter == 0 ){
 statement17.add(mchecktable.group());
names17.add(mchecktable.group(2));
names17.add(mchecktable.group(3));
names17.add(mchecktable.group(4));
 }
if(tables09.size()!=0){
 if(tables09.get(globalcounter).size()                                                    >
Integer.parseInt(mchecktable.group(4).toString()) ){
    passorfail17.add(1);

 }else {
    passorfail17.add(0);
 }
```

```
}else if(tables09.size()==0 && globalcounter == glcounter-1){
   completenumbers(tables09,glcounter,tablesize09);
 for(int cv017=0;cv017<glcounter;cv017++){
 if(tables09.get(cv017).size() > Integer.parseInt(mchecktable.group(4).toString()) ){
   passorfail17.add(1);

}else {
   passorfail17.add(0);
}
}
}

}
//@requires n<13;       #18
while(mcheckvar2.find()) {

Pattern checkvar3 = Pattern.compile("@requires"+ white +variable2 + white +
anisotites + white + digits + white + end);
Matcher mcheckvar3 = checkvar3.matcher(mcheckvar2.group());
 while(mcheckvar3.find()) {



patterns[18]=18;
int num=0;
e = hashint.keys();
if(globalcounter == 0 ){
statement18.add(mcheckvar3.group());
names18.add(mcheckvar3.group(1));   //size se thesi:0  n!
names18.add(mcheckvar2.group(3));names18.add(mcheckvar2.group(4));
}

if(names18.get(0).equals(names15.get(0))){
  num= number15.get(globalcounter);
   if( num < Integer.parseInt(mcheckvar3.group(3)))
      passorfail18.add(1);
   else
      passorfail18.add(0);

}
}
```

```
 }
 //@requires(\forall int i, j;0 <= i && i < j && j < A.length; A[i] <= A[j]);       #19
 while(msorted.find()) {


 patterns[19]=19;
 int num=0;
 e = hashint.keys();
if(globalcounter == 0 ){
 statement19.add(msorted.group());
 names19.add(msorted.group(10));
 names19.add(msorted.group(8));
names19.add(msorted.group(11));names19.add(msorted.group(3));
names19.add(msorted.group(7));names19.add(msorted.group(9));
//j , <= , < , <=
 }
  if(tables09.size() !=0){
 if(tables09.get(globalcounter).size() == 0){
    passorfail19.add(0);
 }
 else  if(tables09.get(globalcounter).size()  ==  1  ||  tables09.get(globalcounter).size()  ==
2){
 passorfail19.add(1);
 }
 else if(tables09.get(globalcounter).size() > 2){
 boolean ascending = false;
 int first = tables09.get(globalcounter).get(0);
 int second = tables09.get(globalcounter).get(1);
 if(second > first) //ascending
  ascending = true;
 int check=0;
 for(int i=1; i<tables09.get(globalcounter).size()-1; i++){
  if((ascending                &&                tables09.get(globalcounter).get(i)                >
tables09.get(globalcounter).get(i+1))                    ||                (!ascending                &&
tables09.get(globalcounter).get(i) < tables09.get(globalcounter).get(i+1) )){
   passorfail19.add(0); check=1;break;}
 }
 if(check==0)
 passorfail19.add(1);
 }
  }
```

```
//when this statement is alone:
if(globalcounter == glcounter-1)
  if(tables09.size() ==0){
    completenumbers(tables09,glcounter, tablesize09);

    for(int cv019=0; cv019 < glcounter; cv019++){

       if(tables09.get(cv019).size() == 0){
         passorfail19.add(0);
       }
       else if(tables09.get(cv019).size() == 1 || tables09.get(cv019).size() == 2){
        passorfail19.add(1);
       }
       else if(tables09.get(cv019).size() > 2){
       boolean ascending = false;
       int first = tables09.get(cv019).get(0);
       int second = tables09.get(cv019).get(1);
       if(second > first) //ascending
         ascending = true;
       int check=0;
       for(int i=1; i<tables09.get(cv019).size()-1; i++){
        if((ascending && tables09.get(cv019).get(i) > tables09.get(cv019).get(i+1))   ||
(!ascending && tables09.get(cv019).get(i) < tables09.get(cv019).get(i+1) )){
         passorfail19.add(0); check=1;break;}
       }
       if(check==0)
         passorfail19.add(1);
      }

    }

}


}

while(marrayresultornot.find()) {

}
while(massigntable.find()) {
```

```
 }
/*@ensures(\forall int i; 0<=i && i <ia.length ==> (\forall int j; 0<=j && j<i ==> (\old(ia[j])>5))) ? (count==count+1) : (ia[i]== 0)); @*/ //#22
 while(mresultarray.find()) {


   patterns[22]=22;
   e = hashint.keys();
   String anisosi = mresultarray.group(12);
   int number = Integer.parseInt(mresultarray.group(13) );
   if(globalcounter == 0 ){
       String statement =mresultarray.group(); //mcheckvarnull.group(1) + "requires " + variab + " "  + mcheckvarnull.group(2) + " null";
       statement22.add(statement);
       if(name02.size()!=0)
        names22.add(name02.get(0));
       else
        names22.add( mresultarray.group(18));
        //jTextArea2.append(mresultarray.group()+"     "+mresultarray.groupCount()     + "\r\n\r\n num == " + mresultarray.group(1)+" "+ mresultarray.group(2) +" pinakas= "+ mresultarray.group(21)+" \r\n" );
    }
   Pattern forstatement = Pattern.compile(white + "" + "for" + white + "\\(" + white + vars + white + variable + white + "=" + white + digits + white + end + white + variable + white + anisotites + white + variable + ".length" + white + end + white + variable2 + "\\+\\+" + white + "\\)" + whitenewline + "\\{"  + whitenewline + "if" + white+"\\(" + white + variable + "\\[" + white + variable + white + "\\]"+white+anisotites+white+ digits + "\\)" + white +"\\{"+whitenewline + variable2+white+operator+operator+white+end+whitenewline+"\\}" +whitenewline+"else" + white + "\\{" + whitenewline + variable + "\\[" + variable + "\\]" + white + "\\=" + white + digits + white + end + white + "\\}" );//
   Matcher mforstatement = forstatement.matcher(text);
  if(tables09.size() !=0){
   while(mforstatement.find()) {


         //(5) == <  //(6) == 5
     int result022 =0;
     if(tables09.get(globalcounter).size()                    >0                    && Integer.parseInt(mforstatement.group(6).toString())==number                    && mforstatement.group(5).equals(anisosi)){
        passorfail22.add(1);
     }else{
        passorfail22.add(0);
```

```
      }
       for(int cv22=0;cv22 <tables09.get(globalcounter).size() ; cv22++){
          if(mforstatement.group(5).equals("<"))
          if(tables09.get(globalcounter).get(cv22) > number){
            result022++;
          }
          if(mforstatement.group(5).equals(">"))
          if(tables09.get(globalcounter).get(cv22) < number){
            result022++;
          }
       }
        result22.add(result022);
      }
 }

   if(globalcounter == glcounter-1)
      if(tables09.size() ==0){
        completenumbers(tables09,glcounter, tablesize09);

        for(int cv022=0; cv022 < glcounter; cv022++){

           mforstatement.reset();
          while(mforstatement.find()) {
                //(5) == <  //(6) == 5
           int result022 =0;
           if(tables09.get(cv022).size()                    >0                    &&
Integer.parseInt(mforstatement.group(6).toString())==number                    &&
mforstatement.group(5).equals(anisosi)){
              passorfail22.add(1);
           }else{
              passorfail22.add(0);
           }
            for(int cv22=0;cv22 <tables09.get(cv022).size() ; cv22++){
              if(mforstatement.group(5).equals("<"))
              if(tables09.get(cv022).get(cv22) > number){
                result022++;
              }
              if(mforstatement.group(5).equals(">"))
              if(tables09.get(cv022).get(cv22) < number){
                result022++;
              }
```

```
            }
             result22.add(result022);
            }
         }
        }

 }
 while(minvariantquation.find()) {

 }
 while(moldandnewvalue.find()) {

 }
 loop1=0;
//text.toCharArray()[mcheckvar.end()]
  //@ensures \result != 0;
  //#25 start
while(mresultdigit.find()) {
          patterns[25]=25;
          int numa=0;
            int numb =0;

          String s1 = null;
         //mwhitenewline
          int num = Integer.parseInt(mresultdigit.group(3));
          String anisosi = mresultdigit.group(2);

      if(globalcounter                                                                    ==
0){printcase25.add(mresultdigit.group(2));printcase25.add(mresultdigit.group(3));
          String statement = mresultdigit.group(1) +"ensures \\result " +
mresultdigit.group(2) + " " +  mresultdigit.group(3) + ";";
      statement25.add(statement);
      }
          mresultdigit_num.add(num);

    Pattern devide = Pattern.compile(whitenewline+""+"if" + whitenewline + "\\(" +
whitenewline+ variable + whitenewline + anisotites + whitenewline + variable + "\\)" +
whitenewline+"\\{"+whitenewline+variable + whitenewline + "=" + whitenewline +
variable +whitenewline+ operator +whitenewline+variable+whitenewline+end);
        Matcher mdevide = devide.matcher(text);
         if(mdevide.find()){
```

```
text =jTextArea5.getText();

    e = hashint.keys();

     name25.add( e.nextElement().toString() );
     name25.add(e.nextElement().toString());
     e = hashint.keys();
     numb =   Integer.parseInt(  hashint.get(e.nextElement().toString()).toString());
     numa =   Integer.parseInt( hashint.get(e.nextElement().toString()).toString() );
     numbera.add(numa);
     numberb.add(numb);
    }

 if(anisosi.equals("!=")){
    if  ( numa == 0 || numb == 0){  result.add(null);passorfail.add(0);}

    else if(numa >= numb){

    if( (numa / numb) == num){

       result.add(num);
       passorfail.add(0);
    }else{
        result.add(numa/numb);
        passorfail.add(1);
      }
    }

    else {
     if( (numb / numa) == num){
       result.add(num);
       passorfail.add(0);
    }else{
        result.add(numb / numa);
        passorfail.add(1);
    }
 }
 }

loop1++;
```

```
  }//end #25
   loop1=0;


      while(mresultnull.find()) {
       // int Integer.parseInt(mresultnull.group(1));
        String anisosi = mresultnull.group(2);

        mresultnull_anisosi.add(anisosi);
     loop1++;

      }
//#26 starts
//@assert(\forall int i,j; 0<=i && i<=j && j<a.length; i<a.length);   //#26
while(massert2.find()) {



   patterns[26]=26;
   e = hashint.keys();
   if(globalcounter == 0 ){
       String statement =massert2.group(); //mcheckvarnull.group(1) + "requires " +
variab + " "  + mcheckvarnull.group(2) + " null";
       statement26.add(statement);
       name26.add(massert2.group(10)+".length");
    }
   Pattern forstatement = Pattern.compile(white + "" + "for" + white + "\\(" + white +
vars + white + variable + white + "=" + white + digits + white + end + white + variable
+ white + anisotites + white + variable + ".length" + white + end + white + variable2 +
"\\+\\+" + white + "\\)" + whitenewline + "\\{"  + whitenewline + variable + white + "="
+ white + variable + white + "\\[" + white + variable + white + "\\]" + white + end +
whitenewline + "\\}");
   Matcher mforstatement = forstatement.matcher(text);


    while(mforstatement.find()) {

  if(mforstatement.group(3).equals("<=")      &&          mforstatement.group(4).equals(
massert2.group(8) )){
       passorfail26.add(1);
       if(globalcounter==0)
       name26.add(mforstatement.group(4));
    }
```

```
   else if(mforstatement.group(3).equals("<") && mforstatement.group(4).equals( massert2.group(8) )){
       passorfail26.add(0);
       if(globalcounter==0)
       name26.add(mforstatement.group(4));
    }

 }
   Pattern forstatementString = Pattern.compile(white + "" + "for" + white + "\\(" + white + vars + white + variable + white + "\\=" + white + digits + white + end + white + variable + white + anisotites + white + variable + ".length" + white + end + white + variable2 + "\\+\\+" + white + "\\)" + whitenewline + "\\{"  + whitenewline + variable + "\\[" + white + variable + white + "\\]" + white + "\\=" + white + "\"" + variable2 + "\"" + white+ end);
   Matcher mforstatementString = forstatementString.matcher(text);
    while(mforstatementString.find()) {

        if(mforstatementString.group(3).equals("<=") && mforstatementString.group(4).equals( massert2.group(8) )){
       passorfail26.add(1);
       if(globalcounter==0)
       name26.add(mforstatementString.group(4));
    }
   else if(mforstatementString.group(3).equals("<") && mforstatementString.group(4).equals( massert2.group(8) )){
       passorfail26.add(0);
       if(globalcounter==0)
       name26.add(mforstatementString.group(4));
    }
    }
  if(globalcounter == glcounter-1)
    if(tables09.size() ==0){
       completenumbers(tables09,glcounter, tablesize09);
    }

 }
  /*@assert(\forall int i; 0<=i && i<(\old(number)-\old(num)); number==\old(number)-1);@*/// #27
  while(mwhilenumcheck.find()) {
    //  jTextArea2.append(" size == " + mwhilenumcheck.groupCount() + " " + "1== " + mwhilenumcheck.group(5)+ "2== " + mwhilenumcheck.group(8));
       anisosis27.add("==");
```

```
        anisosis27.add(mwhilenumcheck.group(8).toString());


    if(globalcounter == 0 ){
        String statement =mwhilenumcheck.group(); //mcheckvarnull.group(1) + "requires
" + variab + " "  + mcheckvarnull.group(2) + " null";
        statement27.add(statement);
        if(name01.size() ==0){

name01.add(mwhilenumcheck.group(7));name01.add(mwhilenumcheck.group(6));


        }
         //0=num 1==number
        //names27.add(mwhilenumcheck.group(10)+".length");
     }
    Pattern whilecheck = Pattern.compile(white + "" + "while"+white +"\\("+white +
variable2 + white + anisotites + white + variable2 + white +"\\)"+ white + "\\{"
+whitenewline + variable2 + counting+white +end );//
    Matcher mwhilecheck = whilecheck.matcher(text);
    if(numbers01.size() !=0){


     while(mwhilecheck.find())  {       jTextArea6.append(     mwhilecheck.group()+"  "
+mwhilecheck.groupCount() + " " + mwhilecheck.group(2) + "");
       patterns[27]=27;
       e = hashint.keys();
       if(numbers01.get(0).get(globalcounter) > numbers01.get(1).get(globalcounter) &&
(numbers01.get(0).get(globalcounter) - numbers01.get(1).get(globalcounter)) >=0 &&
mwhilecheck.group(2).equals(">") && mwhilecheck.group(5).equals("--") ){
          passorfail27.add(1);
        }else {
          passorfail27.add(0);
       }
     }
    }else{
     if(globalcounter == glcounter-1)
      if(numbers01.size() ==0){
        completenumbersfixsize(numbers01 ,2, glcounter);
//jTextArea2.append(" size == " + numbers01.size() + " counter= " + glcounter);
       while(mwhilecheck.find()) {
        patterns[27]=27;
        for(int cv027=0; cv027 < glcounter; cv027++){
```

```
        if(numbers01.get(0).get(cv027) > numbers01.get(1).get(cv027) &&
(numbers01.get(0).get(cv027) - numbers01.get(1).get(cv027)) >=0 &&
mwhilecheck.group(2).equals(">") && mwhilecheck.group(5).equals("--")){
              passorfail27.add(1);
          }else{
               passorfail27.add(0);
             }
         }
        }
       }
   }
  }


mresultnull.reset();
  //@ensures \result!=null; #28
  while(mresultnull.find()) {

       // int Integer.parseInt(mresultnull.group(1));
   String anisosi = mresultnull.group(2);
   mresultnull_anisosi.add(anisosi);
   loop1++;

   Pattern tableini = Pattern.compile(white + "int" + white + variable2 + white + "\\[\\]"
+ white + "\\=" + white + "new" +white + variable + white+ "\\[" +white +
variable2+white +"\\]" + white + end);

   Matcher mtableini = tableini.matcher(text);

  if(numbers01.size() !=0){
   while(mtableini.find()) {
      patterns[28]=28;
       if(globalcounter == 0 ){
          String statement =mresultnull.group(); //mcheckvarnull.group(1) + "requires " +
variab + " "  + mcheckvarnull.group(2) + " null";
          statement28.add(statement);
       }
      if(numbers01.get(0).get(globalcounter) > 0){
         passorfail28.add(1);
      }else{
          passorfail28.add(0);
```

```
      }
    }

  }else{
  if(globalcounter == glcounter-1){

      if(numbers01.size() ==0){
        completenumbersfixsize(numbers01 ,1, glcounter);
//jTextArea2.append(" size == " + numbers01.size() + " counter= " + glcounter);
       while(mtableini.find()) {

             String statement =mresultnull.group(); //mcheckvarnull.group(1) + "requires
" + variab + " "  + mcheckvarnull.group(2) + " null";
             statement28.add(statement);
             patterns[28]=28;
              if(name01.size() ==0){
               name01.add(mtableini.group(2));
              }

           for(int cv028=0; cv028 < glcounter;cv028++){
              if(numbers01.get(0).get(globalcounter) > 0){
                passorfail28.add(1);
              }else{
                 passorfail28.add(0);
              }

           }
       }
      }
  }
  }
 }

  //@requires a.length == b.length ;  #//#29
 while(mchecktablewithtablelength.find()) {
     patterns[29]=29;
     e = hashint.keys();

     int counter=0;
    if(globalcounter == 0 ){
```

```
    String statement =mchecktablewithtablelength.group(); //mcheckvarnull.group(1) +
"requires " + variab + " "  + mcheckvarnull.group(2) + " null";
     statement29.add(statement);
      anisosi29=mchecktablewithtablelength.group(3);
      String a = mchecktablewithtablelength.group(2);
      String b = mchecktablewithtablelength.group(4);
      names29.add(a); names29.add(b);
     }
   if(tables09.size()!=0){
      tables29 = tables09;}
   else if(tables09.size()==0 && globalcounter == 0){ //nato kanw na perni times mono
tou..}

      String tablename = null;

      while(  counter < glcounter*2 ){

       tables09.add(new ArrayList<Integer>());
       int tablesize = intRandomInitialize();
       tablesize09.add(tablesize);
        if(tablesize <0){
          InitilizeTable(0, tables09.get(tables09.size() -1 ));
        }else{
          InitilizeTable(tablesize, tables09.get(tables09.size() -1 ));}
       counter++;
       }
   //initial table size
   tables29 = tables09;
   }

   if( globalcounter == (glcounter-1))
   for(int cv12 =0; cv12<2; cv12++){
      tablesize29.add(new ArrayList<Integer>());
      for(int cv22=0; cv22<glcounter; cv22++)
         tablesize29.get(cv12).add( tablesize09.get(cv22*(cv12+1)) );
    }

   if(globalcounter == (glcounter-1) &&
mchecktablewithtablelength.group(3).equals("==")){

   for(int cv22=0; cv22 <glcounter; cv22++){
```

```
   if(tablesize29.get(0).get(cv22) == tablesize29.get(1).get(cv22))
      passorfail29.add(1);
   else
      passorfail29.add(0);
    }
   }


 }
/*@requires \typeof(num) <: \typeof(str);@*/  //#30
while(mcheckdiftype.find()) {

 patterns[30]=30;
     e = hashint.keys();
String comparea = null;String compareb = null;String compareatype = null;String comparebtype = null;

     int counter=0;
    if(globalcounter == 0 ){
    String statement =mcheckdiftype.group(); //mcheckvarnull.group(1) + "requires " + variab + " "  + mcheckvarnull.group(2) + " null";
    statement30.add(statement);

     String a = mcheckdiftype.group(2);
     String b = mcheckdiftype.group(3);
     names30.add(a); names30.add(b);


     mdefinevariable.reset();
     while(mdefinevariable.find()) {
       if(a.equals(mdefinevariable.group(2))){
         comparea=a;                                compareatype=mdefinevariable.group(1) ;names30.add(compareatype);
       }else if(b.equals(mdefinevariable.group(2))){
         compareb=b;                                                    comparebtype= mdefinevariable.group(1);names30.add(comparebtype);
       }
     }
     if(comparea != null && compareb !=null){
```

```
        if(compareatype.equals(comparebtype))
           passorfail30.add(1);
        else
           passorfail30.add(0);
     }

    // if()
   }
}
//#31
//@ensures \result == ((d == num) ? \old(r) : \old(r) / d) ;
while(mresultexceptions.find()) {

   //names31  variables31 statement31  passorfail31
   String d1 = mresultexceptions.group(3);
   int  num1 = Integer.parseInt(mresultexceptions.group(5).toString());
   String r1 = mresultexceptions.group(6);
    patterns[31]=31;
    if(globalcounter == 0 ){
       String statement =mresultexceptions.group();// 8 spots
       statement31.add(statement);
       names31.add(d1);//0
     //  names31.add(num1);//1
       names31.add(r1);//1



      // jTextArea2.append(""+ mresultexceptions.group(3) + " " + mresultexceptions.group(5)+ " "+ mresultexceptions.group(6) );
       variables31.add(new ArrayList<Integer>());variables31.add(new ArrayList<Integer>());variables31.add(new ArrayList<Integer>());//variables31.add(new ArrayList<Integer>());
    //0= d  //1=num //2=r //3=result
    }
    for(int cv31 =0 ; cv31<variables31.size() ; cv31++){
      variables31.get(cv31).add(intRandomInitialize());
    }
   int varD=variables31.get(0).get(globalcounter); //int varNum=variables31.get(1).get(globalcounter);
   int varR=variables31.get(1).get(globalcounter);
   Pattern fullecxeptioncode = Pattern.compile(white + "try" + white + "\\{" + whitenewline + "return" + white + variable2 + white + operator +white+ variable2 + white + end + whitenewline + "\\}" + white + "catch" + white + "\\(" +white +
```

```
"Exception" + white + variable + white + "\\)" + white + "\\{" + whitenewline +"return" + white + variable2 + white + operator + white + "\\("+white + variable2 + white+ operator + white+ digits + white + "\\)" + white + end);
   Matcher mfullecxeptioncode = fullecxeptioncode.matcher(text);
   boolean find31 = false; boolean find32 = false;
   while(mfullecxeptioncode.find()){
      find31 = true;


      String d = mfullecxeptioncode.group(3); String r = mfullecxeptioncode.group(1); String anisosi1 = mfullecxeptioncode.group(2);
     // jTextArea2.append(d + "= " + d1 +"\r\n" + r + "= " + r1 +"\r\n" +anisosi1 + "= " + mresultexceptions.group(8) +"\r\n" +num1 + "= " + variables31.get(1).get(globalcounter) +"\r\n" );
      if(variables31.get(0).get(globalcounter) != 0 && d.equals(d1) && r.equals(r1) && anisosi1.equals(mresultexceptions.group(8))){
        int result31 = varR /varD; //jTextArea2.append("ELSE"+d + "= " + d1 +"\r\n" + r + "= " + r1 +"\r\n" +anisosi1 + "= " + mresultexceptions.group(8) +"\r\n" +num1 + "= " + variables31.get(1).get(globalcounter) +"\r\n" );


        variables31.get(2).add( result31); passorfail31.add(1);
      }else if(variables31.get(0).get(globalcounter) == 0 && d.equals(d1) && r.equals(r1) && anisosi1.equals(mresultexceptions.group(8)) /*&& varNum !=0*/){
         int result31 =  varR /(varD + 1) ;
         variables31.get(2).add( result31); passorfail31.add(1);
      }else{
         variables31.get(2).add(0); passorfail31.add(0);}
   }
   if(!find31){
   Pattern semiecxeptioncode = Pattern.compile(white + "try" + white + "\\{" + whitenewline + "return" + white + variable2 + white + operator +white+ variable2 + white + end + whitenewline + "\\}" + white  );
   Matcher msemiecxeptioncode = semiecxeptioncode.matcher(text);
    while(msemiecxeptioncode.find()){
        find32 = true;


      String d = msemiecxeptioncode.group(3); String r = msemiecxeptioncode.group(1); String anisosi1 = msemiecxeptioncode.group(2);
     // jTextArea2.append(d + "= " + d1 +"\r\n" + r + "= " + r1 +"\r\n" +anisosi1 + "= " + mresultexceptions.group(8) +"\r\n" +num1 + "= " + variables31.get(1).get(globalcounter) +"\r\n" );
      if(variables31.get(0).get(globalcounter) != 0 && d.equals(d1) && r.equals(r1) && anisosi1.equals(mresultexceptions.group(8))){
```

```
        int result31 = varR /varD; //jTextArea2.append("ELSE"+d + "= " + d1 +"\r\n" + r
+ "= " + r1 +"\r\n" +anisosi1 + "= " + mresultexceptions.group(8) +"\r\n" +num1 + "= "
+ variables31.get(1).get(globalcounter) +"\r\n" );

         variables31.get(2).add( result31); passorfail31.add(1);
      }else  if(variables31.get(0).get(globalcounter) == 0 && d.equals(d1) &&
r.equals(r1) && anisosi1.equals(mresultexceptions.group(8)) /*&& varNum !=0*/){
         // int result31 =  varR /(varD + 1) ;
          //variables31.get(2).add( result31);
          passorfail31.add(0);
      }else{
          variables31.get(2).add(0); passorfail31.add(0);}
    }

   }
   if(!find31 && !find32){

       variables31.get(2).add(null); passorfail31.add(0);
   }



}
//#32
 /*@requires \typeof(numa) <: \type(Object[]);@*/   //#32
int counter32=0;
while(mcheckdifobject.find()) {
 int counter32a=0;

   //names32 types32  statement32 passorfail32
    //public void reference2(Integer[] numb) {
    Pattern function2 = Pattern.compile(""+white + variable +white + variable +white +
variable + white + "\\(" + white + variable2 +white+ brackets + white + variable2 +
white + "\\)" + white + "\\{" );
    Matcher mfunction2 = function2.matcher(text);
    if( (globalcounter == 0 ) || counter32 >= statement32.size() ){

       patterns[32]=32;
      statement32.add(mcheckdifobject.group(0) );
       if(numofstatements <=counter32){

                numofstatements++;
```

```
                }

    }
   //periptosi pou iparxi method ston kodika
   if(globalcounter == 0 && counter32 ==0){
       // mfunction2.reset();
        while(mfunction2.find() ) {
           types32.add(new ArrayList<String>());
            names32.add(new ArrayList<String>());
            passorfail32.add(new ArrayList<Integer>());


           types32.get(counter32a).add(mfunction2.group(1)+mfunction2.group(2));

           names32.get(counter32a).add(mfunction2.group(3));
           if(mfunction2.group(2).equals("[]")){
              passorfail32.get(counter32a).add(1);
           }else{
              passorfail32.get(counter32a).add(0);
           }
           counter32a++;
        }
   }
     //periptosi pou DEN iparxi method ston kodika
    mfunction2.reset();
    if(!mfunction2.find() && (globalcounter ==0 || counter32 >= types32.size()) ){
       types32.get(counter32).add(typeRandomInitialize());
       names32.get(counter32).add(mcheckdifobject.group(2));
        if(types32.get(counter32).get(globalcounter).equals("Integer[]")                    ||
types32.get(counter32).get(globalcounter).equals("Double[]")||types32.get(counter32).g
et(globalcounter).equals("Float[]")||types32.get(counter32).get(globalcounter).equals("
Object[]")){
              passorfail32.get(counter32).add(1);
           }else{
              passorfail32.get(counter32).add(0);
           }
    }

   counter32++;
}
```

```
while(mcheckstatic.find()) {

// m(){
    Pattern anathesi = Pattern.compile(""+white+ variable2 + white + "=" + white + variable2 + "\\." + white + variable2 + white + end);
    Matcher manathesi = anathesi.matcher(text);
   if(globalcounter == 0 ){
      patterns[33]=33;
     statement33.add(mcheckstatic.group(0) );
     while(manathesi.find()) {
        names33.add(manathesi.group(1));
        //jTextArea2.append(" "+manathesi.group());
     }
   }
    manathesi.reset();
     if(!manathesi.find() && globalcounter == (glcounter-1)){
        for(int cv33=0; cv33<6; cv33++){
           int random = (intRandomInitialize() * intRandomInitialize()) % 4 ;
           if(random ==0 )
            names33.add("r1");
           else if(random ==1 )
            names33.add("r2");
           else if(random ==2 )
            names33.add("r3");
           else if(random ==3 )
            names33.add("r4");
           else  names33.add("r1");
        }
          if(names33.get(2).equals("r1") || names33.get(2).equals("r3")){
       passorfail33.add(0); result33.add("Values: r1= True, r2= False, r3= True, r4= True\r\nResult=False");
    }else {passorfail33.add(1); result33.add("Values: r1= True, r2= True, r3= True, r4= True\r\nResult=True");

    }
     }

     manathesi.reset();
     if(manathesi.find()){
    if(names33.get(2).equals("r1") || names33.get(2).equals("r3")){
```

```
        passorfail33.add(0); result33.add("Values: r1= True, r2= False, r3= True, r4= True\r\nResult=False");
    }else {passorfail33.add(1); result33.add("Values: r1= True, r2= True, r3= True, r4= True\r\nResult=True");

    }
     }
}
globalcounter++;
  rs.resethash(hashint, hashfloat, hashdouble, hashchar, hashstring);
rs.resetmethod( mstart, mresultdigit, mresultnull,mcheckvar,marrayforvalue ,mcheckvarnull , marrayforvaluenull,mchecktype ,
          mchecklenght,mcheckvartypewitharray,mchecksizeandarray, mdefinevariable, mdefinetables,mtablesnotnull ,massignablenothing,
          massignablevar, mdefinetablesnonnull, mchecknumvar , mfactorial, mchecktable , mcheckvar2 ,msorted, marrayresultornot,
          massigntable ,mresultarray,minvariantquation,moldandnewvalue, mfunction, mwhitenewline);


  }


int input=-2;
for(int cv1 =0;cv1 <patterns.length;cv1++){
   if(patterns[cv1] == cv1)
   input = cv1;

   else
   input =-1;
   switch (input) {
   case -1:{break;}

   case
1:{print01(statement01,numbers01,passorfail01,name01,digit01,anisosi01);break;}  //ok
   case      2:{print02(tables09,tablesize09,statement02,name02,passorfail02,names02); break;}  //ok
   case       3:{print03(tables09,tablesize09,statement03,passorfail03,names03);break;}
//ok
   case 5:{ print05(statement05, passorfail05, variable05, tablesize05,names05);break;}
//ok einai sto 4o: ( na mpi me tis 2 grammes apo pano specs..)
```

```
   case 6:{ print06(statement06,passorfail06, variable05, tablesize05,names06);break;}//ok
   case 7:{ print07(statement07,names07, compare07,type07);break;}  //ok ok einai sto 4o: ( na mpi me ta 2  specs apo pano ..)
   case 8:{ print08(statement08,names08,passorfail08,tables09);break;}//ok
   case 15:{ print15(statement15,names15,passorfail15,number15);break;}//ok
   case 16:{ print16(statement16,passorfail16,number16,number15,names16);break;}//ok alla prepi na mpi o kwdikas.. to 12..
   case 17:{ print17(statement17,passorfail17,tables09,names17,tablesize09);break;}//ok
   case 18:{ print18(statement18,names18,passorfail18,number15);break;}//ok
   case 19:{ print19(statement19,passorfail19,tables09,names19,tablesize09);break;}//ok
   case 22:{ print22(statement22,passorfail22,tables09,names22,tablesize09,result22);break;}//ok me to kimeno apo kato
   case 25:{ print25("Integer", "Integer",name25 ,  passorfail, numbera, numberb, result,statement25,printcase25);break;}//ok me kwdika.. (prwti periptosi)
   case 26:{print26(statement26,passorfail26,tables09,tablesize09,name26);break;}  //ok
   case 27:{print27(statement27,passorfail27,numbers01,name01,anisosis27);break;} //ok me ton kwdika tou
   case 28:{print28(statement28,numbers01,passorfail28,name01);break;}//ok me kwdika..
   case 29:{print29(statement29,passorfail29,tablesize29,names29,anisosi29);break;}//ok
   case 30:{print30(statement30,passorfail30,names30);break;}//ok
   case 31:{print31(statement31,passorfail31,names31,variables31);break;}//ok
   case 32:{print32(statement32,passorfail32,names32,types32);break;}//ok
   case 33:{print33(statement33,passorfail33,names33,result33);break;}//ok
   //func();
   }
   jTextArea1.setCaretPosition(0);
jTextArea3.setCaretPosition(0);jTextArea4.setCaretPosition(0);
jTextArea5.setCaretPosition(0); jTextArea6.setCaretPosition(0);
jTextArea7.setCaretPosition(0);
}
   }

   private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
//getItemAt(index):       // TODO add your handling code here:
ParserTemplates tm = new ParserTemplates();
String ss=null;
```

```
      int input= jComboBox1.getSelectedIndex();
switch (input) {
      case 0:{ss=tm.devidebyzero1(ss);  jTextArea5.setText(ss); break;}
      case 1:{ss=tm.ArrayIndexOutRange2(ss); jTextArea5.setText(ss); break;}
      case 2:{ss=tm.StringIndexOutRange(ss); jTextArea5.setText(ss); break;}
      case 3:{ss=tm.ArrayStoreEcxeption(ss); jTextArea5.setText(ss); break;}
      case 4:{ss=tm.DereferencingNull(ss); jTextArea5.setText(ss); break;}
      case 5:{ss=tm.StackOverflowArea(ss); jTextArea5.setText(ss); break;}
      case 6:{ss=tm.UnsortedTable(ss); jTextArea5.setText(ss); break;}
      case 7:{ss=tm.InfiniteLoop(ss); jTextArea5.setText(ss); break;}
      case 8:{ss=tm.InfiniteLoopException(ss); jTextArea5.setText(ss); break;}
     // case 9:{ss=tm.CallBack(ss); jTextArea5.setText(ss); break;}
      case 9:{ss=tm.NegativeArraySize(ss); jTextArea5.setText(ss); break;}
      case 10:{ss=tm.SizeOfParallelTables(ss);  jTextArea5.setText(ss); break;}
      case 11:{ss=tm.DiffTypeEquationException(ss); jTextArea5.setText(ss); break;}
      case 12:{ss=tm.StaticInitialisation(ss); jTextArea5.setText(ss); break;}
      case 13:{ss=tm.CatchingException(ss); jTextArea5.setText(ss); break;}
      case 14:{ss=tm.CallbyValueOrReference(ss); jTextArea5.setText(ss); break;}



   default: {ss=tm.devidebyzero1(ss);  jTextArea5.setText(ss); };
}



   }

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

jTextArea1.setText("");jTextArea3.setText("");jTextArea4.setText("");jTextArea5.setT
ext("");jTextArea6.setText("");jTextArea7.setText("");
   }

   private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {
      ParserTemplates tm = new ParserTemplates();
//String ss=null;
          String[] ss = new String[2];
      int input= jComboBox2.getSelectedIndex();
switch (input) {
```

```
    case 0:{tm.RequiresVariable(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]); break;}
    case 1:{tm.RequiresNullHandle(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 2:{tm.RequiresDigit(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 3:{tm.RequiresDigit_toTableLength(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 4:{tm.RequiresTableLengthtoDigit(ss);jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 5:{tm.RequiresSortedTable(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 6:{tm.RequiresSameTableLength(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 7:{tm.RequiresSameType(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 8:{tm.RequiresCallingbyObject_Reference(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 9:{tm.AssertCounter_insideTableBoundaries(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 10:{tm.AssertWhile_Counters(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 11:{tm.VariableTypes_insideTableTypes(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 12:{tm.CheckingCounter_TobeInsideTableSize(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 13:{tm.Invariant_CheckType(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 14:{tm.Factorial(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 15:{tm.EnsuresTableHasValues(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 16:{tm.EnsuresFindMinValuesFromTable(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 17:{tm.EnsuresResultNotZero(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
    case 18:{tm.EnsuresInitializeTable(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]); break;}
    case 19:{tm.EnsuresExceptions(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}

   default: {tm.RequiresVariable(ss); jTextArea5.insert(ss[1], jTextArea5.getCaretPosition()); jTextArea6.setText(ss[0]);break;}
}
```

```
    }

    private void jComboBox3ActionPerformed(java.awt.event.ActionEvent evt) {
      //getItemAt(index):       // TODO add your handling code here:
ParserTemplates tm = new ParserTemplates();
String ss=null;

      int input= jComboBox3.getSelectedIndex();
switch (input) {
      case 0:{ss=tm.devidebyzero12(ss);  jTextArea5.setText(ss); break;}
      case 1:{ss=tm.ArrayIndexOutRange22(ss); jTextArea5.setText(ss); break;}
      case 2:{ss=tm.StringIndexOutRange2(ss); jTextArea5.setText(ss); break;}
      case 3:{ss=tm.ArrayStoreEcxeption2(ss); jTextArea5.setText(ss); break;}
      case 4:{ss=tm.DereferencingNull2(ss); jTextArea5.setText(ss); break;}
      case 5:{ss=tm.StackOverflowArea2(ss); jTextArea5.setText(ss); break;}
      case 6:{ss=tm.UnsortedTable2(ss); jTextArea5.setText(ss); break;}
      case 7:{ss=tm.InfiniteLoop2(ss); jTextArea5.setText(ss); break;}
      case 8:{ss=tm.InfiniteLoopException2(ss); jTextArea5.setText(ss); break;}
     // case 9:{ss=tm.CallBack2(ss); jTextArea5.setText(ss); break;}
      case 9:{ss=tm.NegativeArraySize2(ss); jTextArea5.setText(ss); break;}
      case 10:{ss=tm.SizeOfParallelTables2(ss);  jTextArea5.setText(ss); break;}
      case 11:{ss=tm.DiffTypeEquationException2(ss); jTextArea5.setText(ss); break;}
      case 12:{ss=tm.StaticInitialisation2(ss); jTextArea5.setText(ss); break;}
      case 13:{ss=tm.CatchingException2(ss); jTextArea5.setText(ss); break;}
      case 14:{ss=tm.CallbyValueOrReference2(ss); jTextArea5.setText(ss); break;}



    default: {ss=tm.devidebyzero1(ss);  jTextArea5.setText(ss); };
}
    }

    private void OpenActionPerformed(java.awt.event.ActionEvent evt) {

       int returnVal = fileChooser.showOpenDialog(this);
       if (returnVal == JFileChooser.APPROVE_OPTION) {
          File file = fileChooser.getSelectedFile();
          try {
             // What to do with the file, e.g. display it in a TextArea
             jTextArea5.read( new FileReader( file.getAbsolutePath() ), null );
```

```
            } catch (IOException ex) {
                System.out.println("problem accessing file"+file.getAbsolutePath());
            }
        } else {
            System.out.println("File access cancelled by user.");
        }
    }

    private void ExitActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}

    private void fileChooserActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

   public void print01(ArrayList<String> statement01, ArrayList<ArrayList<Integer>>
numbers01,ArrayList<ArrayList<Integer>>                passorfail01,ArrayList<String>
name01,ArrayList<String> digit01,ArrayList<String> anisosi01){
  //  jTextPane2.setText("hi");jTextPane2.setText("ena");//
  //jTextArea1.append("hi");jTextArea1.append("hello");
    CheckJML xs = new CheckJML();

    for(int cv01 =0; cv01<statement01.size();cv01++) {
        jTextArea1.append("Statement: \r\n" + statement01.get(cv01)+"\r\n\r\n" );
        jTextArea7.append("Statement: \r\n" + statement01.get(cv01)+"\r\n\r\n" );
        jTextArea3.append("Statement: \r\n" + statement01.get(cv01)+"\r\n\r\n" );
        jTextArea4.append("Statement: \r\n" + statement01.get(cv01)+"\r\n\r\n" );
jTextArea4.append(printcases(name01.get(cv01),""+digit01.get(cv01),anisosi01.get(cv0
1),"PASS") );
jTextArea7.append(printcases(name01.get(cv01),""+digit01.get(cv01),anisosi01.get(cv0
1),"FAIL") );

        for(int cv02=0; cv02<numbers01.get(0).size();cv02++){
         // + " Compilation:" + "");
            if(passorfail01.get(cv01).get(cv02)== 1){
                jTextArea1.append("Loop: " + cv02 +"\r\nType:"+"Integer " + "Name: "
+name01.get(cv01)+" Value: "+numbers01.get(cv01).get(cv02) +"\r\n");
                jTextArea1.append("Run Time: " + "PASS"+"\r\n");}
            else if(passorfail01.get(cv01).get(cv02) == 0){
                jTextArea3.append("Loop: " + cv02 +"\r\nType:"+"Integer " + "Name: "
+name01.get(cv01)+" Value: "+numbers01.get(cv01).get(cv02) +"\r\n");
```

```
                jTextArea3.append("Run Time: " + "FAIL"+"\r\n");}

        }
         jTextArea1.append("\r\n");   jTextArea3.append("\r\n");
    }
}
  public void print02(ArrayList<ArrayList<Integer>> tables09,ArrayList<Integer>
tablesize09,ArrayList<String> statement02,ArrayList<String>
name02,ArrayList<ArrayList<Integer>> passorfail02,ArrayList<String> names02){
     //  jTextArea2.append(""+tables09.size());
    for(int cv01 =0; cv01<statement02.size();cv01++) {
    jTextArea1.append("Statement: " + statement02.get(cv01)+"\r\n\r\n" );
    jTextArea3.append("Statement: " + statement02.get(cv01)+"\r\n\r\n" );
    jTextArea4.append("Statement: " + statement02.get(cv01)+"\r\n\r\n" );
    jTextArea7.append("Statement: " + statement02.get(cv01)+"\r\n\r\n" );

jTextArea4.append(printcases2(names02.get(0),names02.get(2),names02.get(1),"PASS"
));

jTextArea7.append(printcases2(names02.get(0),names02.get(2),names02.get(1),"FAIL"
));

        for(int cv02=0; cv02<glcounter;cv02++){

            if(passorfail02.get(cv01).get(cv02)== 1){
               jTextArea1.append("Loop: " + cv02 +"\r\nTable: " +  name02.get(cv01) );
                jTextArea1.append("!= null\r\nRun Time: " + "PASS"+"\r\n");
            jTextArea1.append("\r\n");
            }
            else if(passorfail02.get(cv01).get(cv02) == 0){
               jTextArea3.append("Loop: " + cv02 +"\r\nTable: " +  name02.get(cv01) );
                jTextArea3.append("== null\r\nRun Time: " + "FAIL"+"\r\n");
                jTextArea3.append("\r\n");
            }

        }

    }

   }
```

```
  public void print03(ArrayList<ArrayList<Integer>> tables09,ArrayList<Integer> tablesize09,ArrayList<String> statement03,ArrayList<Integer> passorfail03,ArrayList<String> names03 ){
     int size1 = tables09.size();
  //     0 <= i && i < a.length
//C == a[i]
//i=0, 0=1 , <==2, c=3, ===4, a=5,< =6
      jTextArea1.append("\r\nStatement: "+statement03.get(0)+"\r\n\r\n" );
      jTextArea3.append("\r\nStatement: "+statement03.get(0)+"\r\n\r\n" );
      jTextArea4.append("\r\nStatement: "+statement03.get(0)+"\r\n\r\n" );
      jTextArea7.append("\r\nStatement: "+statement03.get(0)+"\r\n\r\n" );
      jTextArea4.append(printcases3(names03.get(1),names03.get(0)+"                &&
",names03.get(2),names03.get(0),names03.get(5)+".length",names03.get(6),"PASS"));

jTextArea7.append(printcases3(names03.get(1),names03.get(0)+"",names03.get(2),nam
es03.get(0),names03.get(5)+".length",names03.get(6),"FAIL"));

jTextArea4.append(printcases2(names03.get(3),names03.get(5)+"["+names03.get(5)+".
length-1"+"]",names03.get(4),"PASS"));

jTextArea7.append(printcases2(names03.get(3),names03.get(5)+"["+names03.get(5)+".
length-1"+"]",names03.get(4),"FAIL"));
     if(passorfail03.size() !=0)
      for(int cv01 =0; cv01 < size1; cv01++){

         if(passorfail03.get(cv01) ==1){
              jTextArea1.append("Loop: " + cv01 +"\r\nTable " + cv01 + " size= " +
tables09.get(cv01).size()+"\r\n"+"Values: ");
            for(int cv02=0; cv02<tables09.get(cv01).size();cv02++){
             jTextArea1.append(tables09.get(cv01).get(cv02) + " ");
            }
            jTextArea1.append("\r\nResults:");
           //  for(int cv02=0; cv02<tables09.get(cv01).size();cv02++){
             if(tables09.get(cv01).size() >0){
               jTextArea1.append(tables09.get(cv01).get(tables09.get(cv01).size()-1)   +   "
");
            }
            jTextArea1.append("\r\nRun Time: PASS");
            jTextArea1.append("\r\n"); jTextArea1.append("\r\n");
         }
         else  if(tablesize09.get(cv01)==-1){
            jTextArea3.append("Loop: " + cv01 +"\r\nTable " + cv01 + " size=
"+"null\r\n"+"Values: null\r\n"+"Results: null\r\n"+"Run Time: FAIL\r\n"+"\r\n");
```

```
            jTextArea3.append("\r\n");
         }else{
            jTextArea3.append("Loop: " + cv01 +"\r\nTable " + cv01 + " size= " +
tables09.get(cv01).size()+"\r\n"+"Values: ");
             for(int cv02=0; cv02<tables09.get(cv01).size();cv02++){
              jTextArea3.append(tables09.get(cv01).get(cv02) + " ");
             }
             jTextArea3.append("\r\nResults:");
        //    for(int cv02=0; cv02<tables09.get(cv01).size();cv02++){
            if(tables09.get(cv01).size() >0){
             jTextArea3.append(tables09.get(cv01).get(tables09.get(cv01).size() -1) + "
");
             }
             jTextArea3.append("\r\nRun Time: FAIL");
             jTextArea3.append("\r\n"); jTextArea3.append("\r\n");
         }



      }
      else{
       jTextArea1.append("Missing Input.\r\n" );
       jTextArea3.append("Missing Input.\r\n" );
    }


   }
   public void print05(ArrayList<String> statement05, ArrayList<Integer> passorfail05,
ArrayList<Integer> variable05, ArrayList<Integer> tablesize05,ArrayList<String>
names05){
     int size = passorfail05.size()-1;
   //Print4 : 0<= top top <= elems.length - pass
//Print2: Elems == type of object  - pass
//0. 0 //1. top //2. <=! //3. elems! //4. <=! //5. ==! //6. object!
    jTextArea1.append("Statement: " + statement05.get(0)+"\r\n\r\n" );
    jTextArea3.append("Statement: " + statement05.get(0)+"\r\n\r\n" );
    jTextArea4.append("Statement: " + statement05.get(0)+"\r\n\r\n" );
    jTextArea7.append("Statement: " + statement05.get(0)+"\r\n\r\n" );
    jTextArea4.append(printcases3(names05.get(0),names05.get(1)+"                    &&
",names05.get(2),names05.get(1),names05.get(3)+".length",names05.get(4),"PASS"));

jTextArea7.append(printcases3(names05.get(0),names05.get(1)+"",names05.get(2),nam
es05.get(1),names05.get(3)+".length",names05.get(4),"FAIL"));
```

```
   jTextArea4.append(printcases2(names05.get(3),"Type
of("+names05.get(6)+")",names05.get(5),"PASS"));
   jTextArea7.append(printcases2(names05.get(3),"Type
of("+names05.get(6)+")",names05.get(5),"FAIL"));

    for(int cv05 = 0; cv05 <= size;cv05++){
       if(passorfail05.get(cv05) == 1 ){
          jTextArea1.append("Loop: " + cv05 +"\r\nValues: "+"\r\n");
          jTextArea1.append(names05.get(1) + ": "+variable05.get(cv05)+"\r\n" );
          if(tablesize05.get(cv05) == -1 )
            jTextArea1.append(names05.get(3) + ".length: "+" null \r\n");
          else
            jTextArea1.append(names05.get(3) + ".length: "+tablesize05.get(cv05) +
"\r\n");
             jTextArea1.append(names05.get(3) + " type:"
+names05.get(6)+"\r\nStatement true\r\n\r\n");
       }
       else{
          jTextArea3.append("Loop: " + cv05 +"\r\nValues: "+"\r\n");
           jTextArea3.append(names05.get(1) + ": "+variable05.get(cv05)+"\r\n" );
          if(tablesize05.get(cv05) == -1 )
            jTextArea3.append(names05.get(3) + ".length: "+" null \r\n");
          else
            jTextArea3.append(names05.get(3) + ".length: "+tablesize05.get(cv05) +
"\r\n");
             jTextArea3.append(names05.get(3) + " type:
"+names05.get(6)+"\r\nStatement false\r\n\r\n");
       }

   }
   }
    public void print06(ArrayList<String> statement06, ArrayList<Integer>
passorfail06,ArrayList<Integer> variable05, ArrayList<Integer>
tablesize05,ArrayList<String> names06){
 int size = passorfail06.size()-1;
   jTextArea1.append("Statement: " + statement06.get(0)+"\r\n\r\n" );
   jTextArea3.append("Statement: " + statement06.get(0)+"\r\n\r\n" );
   jTextArea4.append("Statement: " + statement06.get(0)+"\r\n\r\n" );
   jTextArea7.append("Statement: " + statement06.get(0)+"\r\n\r\n" );

jTextArea4.append(printcases(names06.get(0),names06.get(1)+".length",names06.get(2
),"PASS"));
```

```
jTextArea7.append(printcases(names06.get(0),names06.get(1)+".length",names06.get(2
),"FAIL"));

 for(int cv05 = 0; cv05 <= size;cv05++){
    if(passorfail06.get(cv05) == 1 ){
     jTextArea1.append("Loop: " + cv05 +"\r\nValues: "+"\r\n");
     jTextArea1.append(names06.get(0) + ": "+variable05.get(cv05)+"\r\n" );
    if(tablesize05.get(cv05) == -1 )
      jTextArea1.append(names06.get(1) + ".length: "+" null \r\n");
    else
      jTextArea1.append(names06.get(1) + ".length: "+tablesize05.get(cv05) + "\r\n");

       jTextArea1.append("Statement true\r\n\r\n");}
    else{
     jTextArea3.append("Loop: " + cv05 +"\r\nValues: "+"\r\n");
     jTextArea3.append(names06.get(0) + ": "+variable05.get(cv05)+"\r\n" );
    if(tablesize05.get(cv05) == -1 )
      jTextArea3.append(names06.get(1) + ".length: "+" null \r\n");
    else
      jTextArea3.append(names06.get(1) + ".length: "+tablesize05.get(cv05) + "\r\n");

      jTextArea3.append("Statement false\r\n\r\n");}

   }

   }
   public void print07(ArrayList<String> statement07, ArrayList<String> names07,int
compare07,ArrayList<String> type07){

   jTextArea1.append("Statement: " + statement07.get(0)+"\r\n\r\n" );
   jTextArea3.append("Statement: " + statement07.get(0)+"\r\n\r\n" );
   jTextArea4.append("Statement: " + statement07.get(0)+"\r\n\r\n" );
   if(names07.size()!=0){
   jTextArea7.append("Statement: " + statement07.get(0)+"\r\n\r\n" );
   jTextArea4.append(printcases2("Type          of          ("+names07.get(0)+")","Type
of("+names07.get(1)+")","==","PASS"));
   jTextArea7.append(printcases2("Type          of          ("+names07.get(0)+")","Type
of("+names07.get(1)+")","==","FAIL"));

   if(compare07 == 1){
```

```
   jTextArea1.append("Loop: 0" +"\r\nVariable: " + names07.get(0) + " Type:" + type07.get(0) );
   jTextArea1.append("\r\nVariable: " + names07.get(1) + " Type:" + type07.get(1) );
   jTextArea1.append("\r\nType of " + names07.get(1) + " contains type of " + names07.get(0) );
   jTextArea1.append("\r\nRun Time: PASS");
   }else{
   jTextArea3.append("Loop: 0"  +"\r\nVariable: " + names07.get(0) + " Type:" + type07.get(0) );
   jTextArea3.append("\r\nVariable: " + names07.get(1) + " Type:" + type07.get(1) );
   jTextArea3.append("\r\nType of " + names07.get(1) + " does not contain type of " + names07.get(0) );
   jTextArea3.append("\r\nRun Time: FAIL");
   }
   }else{
      jTextArea1.append("Missing Input.\r\n" );
      jTextArea3.append("Missing Input.\r\n" );
   }
  }
   public void print08(ArrayList<String> statement08, ArrayList<String> names08,ArrayList<Integer> passorfail08, ArrayList<ArrayList<Integer>> tables09){
//size=0, elems=1 , 0=2 , <= = 3i , <= = 4i
      //0<=size && size <= elements.length;
    jTextArea1.append("\r\nStatement:\r\n" + statement08.get(0)+"\r\n" );
    jTextArea3.append("\r\nStatement:\r\n" + statement08.get(0)+"\r\n" );
    jTextArea4.append("\r\nStatement:\r\n" + statement08.get(0) +"\r\n");
    jTextArea7.append("\r\nStatement:\r\n" + statement08.get(0) +"\r\n");
     jTextArea4.append(printcases3(names08.get(2),names08.get(0)+"  &&
",names08.get(3),names08.get(0),names08.get(1)+".length",names08.get(4),"PASS"));

jTextArea7.append(printcases3(names08.get(2),names08.get(0),names08.get(3),names0
8.get(0),names08.get(1)+".length",names08.get(4),"FAIL"));
   int size = passorfail08.size();
   for(int cv08 =0; cv08<size;cv08++){


   if(passorfail08.get(cv08) == 1){
  // jTextArea1.append("\r\nType of " + names08.get(1) + " contains type of " + names07.get(0) );
   jTextArea1.append("Loop: " + cv08 +"\r\nVariable: " + names08.get(0) + ", Value:"
+tables09.get(cv08).size() );
   jTextArea1.append("\r\nVariable: " + names08.get(1) +".length"+ ", Value:" + tables09.get(cv08).size());
```

```
   jTextArea1.append("\r\nRun Time: PASS\r\n\r\n");
   }else{
   //jTextArea1.append("\r\nType of " + names08.get(1) + " does not contain type of " +
names07.get(0) );
   jTextArea3.append("Loop: " + cv08 +"\r\nVariable: " + names08.get(0) + ", Value:
null" );
   jTextArea3.append("\r\nVariable: " + names08.get(1) +".length"+ ", Value: null" );
   jTextArea3.append("\r\nRun Time: FAIL\r\n\r\n");
   }
   }
  }
   /*@requires 1 <= n;*/
   // case 15:{ print15(statement15,names15,passorfail15,number15);break;}
   public void print15(ArrayList<String> statement15, ArrayList<String>
names15,ArrayList<Integer> passorfail15,ArrayList<Integer> number15){
    jTextArea1.append("\r\nStatement:\r\n" + statement15.get(0)+"\r\n" );
    jTextArea3.append("\r\nStatement:\r\n" + statement15.get(0)+"\r\n" );
    jTextArea4.append("\r\nStatement:\r\n" + statement15.get(0) +"\r\n");
    jTextArea7.append("\r\nStatement:\r\n" + statement15.get(0) +"\r\n");

jTextArea4.append(printcases(names15.get(1),names15.get(0),names15.get(2),"PASS")
);

jTextArea7.append(printcases(names15.get(1),names15.get(0),names15.get(2),"FAIL"))
;
   //if(passorfail15.size() !=null)
   int size = passorfail15.size();
   for(int cv15 =0; cv15<size;cv15++){

   if(passorfail15.get(cv15) == 1){
      jTextArea1.append("Loop: " + cv15 +"\r\nVariable: " + names15.get(0) + ",
Value:" +number15.get(cv15)+"" );
      jTextArea1.append("\r\nRun Time: PASS\r\n");
      jTextArea1.append("\r\n");
   }else{
      jTextArea3.append("Loop: " + cv15 +"\r\nVariable: " + names15.get(0) + ",
Value:" +number15.get(cv15)+"" );
      jTextArea3.append("\r\nRun Time: FAIL\r\n");
      jTextArea3.append("\r\n");
   }

   }
```

```
    }
    public void print16(ArrayList<String> statement16,ArrayList<Integer>
passorfail16,ArrayList<Integer> number16,ArrayList<Integer>
number15,ArrayList<String> names16){

// 0=i 1=1  2<=  3<= 4< 5=n 6=6227020800
     jTextArea1.append("\r\nStatement:\r\n" + statement16.get(0)+"\r\n" );
     jTextArea3.append("\r\nStatement:\r\n" + statement16.get(0)+"\r\n" );
     jTextArea4.append("\r\nStatement:\r\n" + statement16.get(0) +"\r\n");
     jTextArea7.append("\r\nStatement:\r\n" + statement16.get(0) +"\r\n");
     jTextArea4.append(printcases("Result"+
".length",names16.get(6),names16.get(4),"PASS"));
     jTextArea7.append(printcases("Result"+
".length",names16.get(6),names16.get(4),"FAIL"));
     jTextArea4.append(printcases3(names16.get(1),names16.get(0)+"                &&
",names16.get(2),names16.get(0),names16.get(5),names16.get(3),"PASS"));
     jTextArea7.append(printcases3(names16.get(1),names16.get(0)+"                &&
",names16.get(2),names16.get(0),names16.get(5),names16.get(3),"FAIL"));

    int size = passorfail16.size();
    for(int cv16 =0; cv16<size;cv16++){
    //jTextArea1.append("Variable: " + names15.get(0) + ", Value:"
+number15.get(cv15)+"" );
    if(passorfail16.get(cv16) == 1){
       jTextArea1.append("\r\nLoop: " + cv16 +"\r\nFactorial:" +
number15.get(cv16)+"\r\nResult:"+number16.get(cv16)+"\r\nRun Time: PASS\r\n");
        jTextArea1.append("\r\n");
    }else{
       jTextArea3.append("\r\nLoop: " + cv16 +"\r\nFactorial:" +
number15.get(cv16)+"\r\nResult: null \r\nRun Time: FAIL\r\n");
        jTextArea3.append("\r\n");
    }

    }
    }

    public void print17(ArrayList<String> statement17, ArrayList<Integer>
passorfail17,ArrayList<ArrayList<Integer>> tables09, ArrayList<String>
names17,ArrayList<Integer> tablesize09){
       jTextArea1.append("\r\nStatement:\r\n" + statement17.get(0)+"\r\n" );
     jTextArea3.append("\r\nStatement:\r\n" + statement17.get(0)+"\r\n" );
     jTextArea4.append("\r\nStatement:\r\n" + statement17.get(0) +"\r\n");
```

```
    jTextArea7.append("\r\nStatement:\r\n" + statement17.get(0) +"\r\n");

jTextArea4.append(printcases(names17.get(0)+".length",names17.get(2),names17.get(1
),"PASS"));

jTextArea7.append(printcases(names17.get(0)+".length",names17.get(2),names17.get(1
),"FAIL"));
   if(passorfail17.size() != 0){
   int size = passorfail17.size();
   for(int cv17 =0; cv17<size;cv17++){



   if(passorfail17.get(cv17) == 1){
      jTextArea1.append("Loop: " + cv17 +"\r\nVariable: " +names17.get(0)+".length "+
"Value:" +tables09.get(cv17).size()+"" );
      jTextArea1.append("\r\nRun Time: PASS\r\n");
      jTextArea1.append("\r\n");
   }else  if(tablesize09.get(cv17)==-1){
     jTextArea3.append("Loop: " + cv17 +"\r\nVariable: " +names17.get(0)+".length "+
"Value: null\r\n"+"Run Time: FAIL\r\n"+"\r\n");
       jTextArea3.append("\r\n");
   }else{
      jTextArea3.append("Loop: " + cv17 +"\r\nVariable: " +names17.get(0)+".length "+
"Value:" +tables09.get(cv17).size()+"" );
      jTextArea3.append("\r\nRun Time: FAIL\r\n");
       jTextArea3.append("\r\n");
   }



   }
   }
   }

   public void print18(ArrayList<String> statement18, ArrayList<String>
names18,ArrayList<Integer> passorfail18,ArrayList<Integer> number15){
   jTextArea1.append("\r\nStatement:\r\n" + statement18.get(0)+"\r\n\r\n" );
   jTextArea3.append("\r\nStatement:\r\n" + statement18.get(0)+"\r\n\r\n" );
   jTextArea4.append("\r\nStatement:\r\n" + statement18.get(0) +"\r\n");
   jTextArea7.append("\r\nStatement:\r\n" + statement18.get(0) +"\r\n");

jTextArea4.append(printcases(names18.get(0),names18.get(2),names18.get(1),"PASS")
);
```

```
jTextArea7.append(printcases(names18.get(0),names18.get(2),names18.get(1),"FAIL"))
;
   if(number15.size() != 0){
   int size = passorfail18.size();
   for(int cv18 =0; cv18<size;cv18++){

   if(passorfail18.get(cv18) == 1){
       jTextArea1.append("Loop: " + cv18 +"\r\nVariable: " + names18.get(0) + ",
Value:" +number15.get(cv18)+"" );
      jTextArea1.append("\r\nRun Time: PASS\r\n");
       jTextArea1.append("\r\n");
   }else{
       jTextArea3.append("Loop: " + cv18 +"\r\nVariable: " + names18.get(0) + ",
Value:" +number15.get(cv18)+"" );
      jTextArea3.append("\r\nRun Time: FAIL\r\n");
       jTextArea3.append("\r\n");
   }

   }
   }
   }

   public void print19(ArrayList<String> statement19, ArrayList<Integer>
passorfail19,ArrayList<ArrayList<Integer>> tables09, ArrayList<String>
names19,ArrayList<Integer> tablesize09){


    jTextArea1.append("\r\nStatement:\r\n" + statement19.get(0)+"\r\n\r\n" );
    jTextArea3.append("\r\nStatement:\r\n" + statement19.get(0)+"\r\n\r\n" );
    jTextArea4.append("\r\nStatement:\r\n" + statement19.get(0) +"\r\n");
    jTextArea7.append("\r\nStatement:\r\n" + statement19.get(0) +"\r\n");
   jTextArea4.append(printcases3(names19.get(1),"0                                    &&
",names19.get(3),names19.get(1),names19.get(0)+".length",names19.get(4),"PASS"));
   jTextArea7.append(printcases3(names19.get(1),"0                                    &&
",names19.get(3),names19.get(1),names19.get(0)+".length",names19.get(4),"FAIL"));
         //a i j


  jTextArea4.append(printcases(names19.get(0)+"["+names19.get(1)+"]
",names19.get(0)+"["+names19.get(2)+"] ",names19.get(5),"PASS"));
     jTextArea7.append(printcases(names19.get(0)+"["+names19.get(1)+"]
",names19.get(0)+"["+names19.get(2)+"] ",names19.get(5),"FAIL"));
```

```
if(passorfail19.size() != 0){
int size = passorfail19.size();
for(int cv17 =0; cv17<size;cv17++){



if(passorfail19.get(cv17) == 1){
   jTextArea1.append("Loop: " + cv17 +"\r\nVariable: "+names19.get(0)+ " Value:"
+tables09.get(cv17) );
   jTextArea1.append("\r\nRun Time: PASS\r\n");
   jTextArea1.append("\r\n");
}else  if(tablesize09.get(cv17)==-1){
 jTextArea3.append("Loop: " + cv17 +"\r\nVariable: " +names19.get(0)+ " Value:
null\r\n"+"Run Time: FAIL\r\n"+"\r\n");
}else{
   jTextArea3.append("Loop: " + cv17 +"\r\nVariable: "+names19.get(0)+ " Value:"
+tables09.get(cv17) );
   jTextArea3.append("\r\nRun Time: FAIL\r\n");
   jTextArea3.append("\r\n");
}



}
}
}
public void print22(ArrayList<String> statement22, ArrayList<Integer>
passorfail22,ArrayList<ArrayList<Integer>> tables09, ArrayList<String>
names22,ArrayList<Integer> tablesize09,ArrayList<Integer> result22){
   jTextArea1.append("\r\nStatement:\r\n" + statement22.get(0)+"\r\n\r\n" );
   jTextArea3.append("\r\nStatement:\r\n" + statement22.get(0)+"\r\n\r\n" );
   jTextArea4.append("\r\nStatement:\r\n" + statement22.get(0) );
   jTextArea7.append("\r\nStatement:\r\n" + statement22.get(0) );
   jTextArea4.append("\r\nCase 1: IF 0<=i && i <"+names22.get(0)+".length -
PASS");
   jTextArea7.append("\r\nCase 2: IF 0>i || i >="+names22.get(0)+".length - FAIL");
   jTextArea4.append("\r\nCase 3: IF "+names22.get(0)+"[j]> 5 then
counter==counter +1 && IF "+names22.get(0)+"[i]<5 THEN "+names22.get(0)+"[i]
== 0 - PASS");
   jTextArea7.append("\r\nCase 4: IF "+names22.get(0)+"[i]>5 && counter
!=counter+1 - FAIL");
   jTextArea7.append("\r\nCase 5: IF "+names22.get(0)+"[i]<5 &&
"+names22.get(0)+"[i] !=0 - FAIL");
```

```
   if(passorfail22.size() != 0){
   int size = passorfail22.size();
   for(int cv22 =0; cv22<size;cv22++){
      if(passorfail22.get(cv22) == 1){
         jTextArea1.append("Loop: " + cv22 +"\r\nVariable: " +names22.get(0)+".length
Value:" +tables09.get(cv22) );
         jTextArea1.append("\r\nResult: " + result22.get(cv22));
         jTextArea1.append("\r\nRun Time: PASS\r\n");
          jTextArea1.append("\r\n");
      }else if(tablesize09.get(cv22)==-1){
       jTextArea3.append("Loop: " + cv22 +"\r\nVariable: "+names22.get(0)+".length
Value: null\r\n"+"Run Time: FAIL\r\n"+"\r\n");
         jTextArea3.append("\r\n");
      }else{
         jTextArea3.append("Loop: " + cv22 +"\r\nVariable: " +names22.get(0)+".length
Value:" +tables09.get(cv22) );
         jTextArea3.append("\r\nResult: " + result22.get(cv22));
         jTextArea3.append("\r\nRun Time: FAIL\r\n");
         jTextArea3.append("\r\n");
      }

   }
   }
   }

   public void print25(String typea,String typeb,ArrayList<String> name25,
ArrayList<Integer> passorfail, ArrayList<Integer> numbera , ArrayList<Integer>
numberb, ArrayList<Integer> result, ArrayList<String> statement25,ArrayList<String>
printcase25 ){

   int size = result.size();
   int counter=0;

      jTextArea1.append("\r\nStatement:\r\n" + statement25.get(0)+"\r\n\r\n" );
      jTextArea3.append("\r\nStatement:\r\n" + statement25.get(0)+"\r\n\r\n" );
      jTextArea4.append("\r\nStatement:\r\n" + statement25.get(0)+"\r\n\r\n" );
      jTextArea7.append("\r\nStatement:\r\n" + statement25.get(0)+"\r\n\r\n" );

jTextArea4.append(printcases("Result",printcase25.get(1),printcase25.get(0),"PASS"));

jTextArea7.append(printcases("Result",printcase25.get(1),printcase25.get(0),"FAIL"));
```

```
   while(counter <size){
      if(passorfail.get(counter) == 1){
         jTextArea1.append("Variables - Loop No:" + counter+"\r\n");
         jTextArea1.append("Type:" + typeb + " Name: " + name25.get(1) + " Value: " +
numbera.get(counter)+"\r\n");
         jTextArea1.append("Type:" + typea + " Name: " + name25.get(0) + " Value: " +
numberb.get(counter)+"\r\n");
         jTextArea1.append("Result: " + result.get(counter)+"\r\n");
         jTextArea1.append("Run Time: " + "PASS"+"\r\n");
       jTextArea1.append("\r\n");}
      else if(passorfail.get(counter) == 0){
         jTextArea3.append("Variables - Loop No:" + counter+"\r\n");
         jTextArea3.append("Type:" + typeb + " Name: " + name25.get(1) + " Value: " +
numbera.get(counter)+"\r\n");
         jTextArea3.append("Type:" + typea + " Name: " + name25.get(0) + " Value: " +
numberb.get(counter)+"\r\n");
         jTextArea3.append("Result: " + result.get(counter)+"\r\n");
         jTextArea3.append("Run Time: " + "FAIL"+"\r\n");
         jTextArea3.append("\r\n");
      }
    counter++;

   }



}
   public void print26(ArrayList<String> statement26,ArrayList<Integer>
passorfail26,ArrayList<ArrayList<Integer>> tables09,ArrayList<Integer>
tablesize09,ArrayList<String> name26){
       jTextArea1.append("Statement:\r\n" + statement26.get(0)+"\r\n\r\n" );
       jTextArea3.append("Statement:\r\n" + statement26.get(0)+"\r\n\r\n" );
       jTextArea4.append("Statement:\r\n" + statement26.get(0)+"\r\n\r\n" );
       jTextArea7.append("Statement:\r\n" + statement26.get(0)+"\r\n\r\n" );
      jTextArea4.append(printcases3(name26.get(1),"0",">="," &&
"+name26.get(1),name26.get(0),"<","PASS"));

jTextArea7.append(printcases3(name26.get(1),"0",">=",""+name26.get(1),name26.get(
0),"<","FAIL"));

          int size1 = tables09.size();
        for(int cv01 =0; cv01 < size1; cv01++){
```

```
        // jTextArea1.append("Values" + "\r\n"+name26.get(1) +": ");//
jTextArea3.append("Values" + "\r\n"+name26.get(1) +": ");
       if(passorfail26.get(cv01)==1){
          jTextArea3.append("Loop: " + cv01 +"\r\nValues" + "\r\n"+name26.get(1) +":
");
                  for(int                                                    cv02=0;
cv02<tables09.get(cv01).size()+passorfail26.get(cv01);cv02++){
             jTextArea3.append(cv02 + ", ");
         } jTextArea3.append("\r\n"+name26.get(0)+": " + tables09.get(cv01).size());
          jTextArea3.append("\r\nRun Time: FAIL"); jTextArea3.append("\r\n\r\n");
       }
       else if(tablesize09.get(cv01)==-1){
           jTextArea3.append("Loop: " + cv01 +"\r\nValues: null\r\n"+"Run Time:
FAIL\r\n"+"\r\n");
        }
        else{
           jTextArea1.append("Loop: " + cv01 +"\r\nValues" + "\r\n"+name26.get(1) +":
");
             for(int                                                    cv02=0;
cv02<tables09.get(cv01).size()+passorfail26.get(cv01);cv02++){
             jTextArea1.append(cv02 + ", ");
         } jTextArea1.append("\r\n"+name26.get(0)+": " + tables09.get(cv01).size());
          jTextArea1.append("\r\nRun Time: PASS");
           jTextArea1.append("\r\n\r\n");
         }




     }


   }
   // /*@assert(\forall int i; 0<=i && i<(\old(number)-\old(num));
number==\old(number)-1);@*/ //#27
   public void print27(ArrayList<String> statement27,ArrayList<Integer>
passorfail27,ArrayList<ArrayList<Integer>>                numbers01,ArrayList<String>
name01,ArrayList<String> anisosis27){
       jTextArea1.append("Statement:" + statement27.get(0)+"\r\n\r\n" );
       jTextArea3.append("Statement:" + statement27.get(0)+"\r\n\r\n" );
       jTextArea4.append("Statement:" + statement27.get(0)+"\r\n\r\n" );
       jTextArea7.append("Statement:" + statement27.get(0)+"\r\n\r\n" );
```

```
  jTextArea7.append(      printcases4("Loop Times","( " + name01.get(0) + "-"+
name01.get(1) + ") && ","==","      Inside loop: "+ name01.get(0), "old("
+name01.get(0)+ ")-1", anisosis27.get(0),"PASS" ));
    jTextArea4.append(      printcases4("Loop Times","( " + name01.get(0) + "-"+
name01.get(1) + ") && ","==","      Inside loop: "+ name01.get(0), "old("
+name01.get(0)+ ")-1", anisosis27.get(0),"FAIL" ));
      // printcases4("Loop Times",  + name01.get(0) + " - ", name01.get(1) + ")  && " ,
"Inside Loop");
      // jTextArea4.append(printcases2("Result","null",anisosi28,"PASS"));
     //   jTextArea7.append(printcases2("Result","null",anisosi28,"FAIL"));
       int size1 = glcounter;
       for(int cv01 =0; cv01 < size1; cv01++){


          if(passorfail27.get(cv01) == 1){
                jTextArea1.append("Loop: " + cv01 +"\r\nVariable: " +name01.get(0)+ "
Value: " + numbers01.get(0).get(cv01));
                jTextArea1.append("\r\nVariable:  "  +name01.get(1)+  "  Value:  "  +
numbers01.get(1).get(cv01));
                jTextArea1.append("\r\nRun Time: PASS\r\n");}
            else{
                jTextArea3.append("Loop: " + cv01 +"\r\nVariable: " +name01.get(0)+ "
Value: " + numbers01.get(0).get(cv01));
                jTextArea3.append("\r\nVariable:  "  +name01.get(1)+  "  Value:  "  +
numbers01.get(1).get(cv01));
                jTextArea3.append("\r\nRun Time: FAIL\r\n");}
       }
     }
      public void print28(ArrayList<String> statement28, ArrayList<ArrayList<Integer>>
numbers01,ArrayList<Integer> passorfail28,ArrayList<String> name01){
        //@ensures \result != null;   #28
         int  size1  =  passorfail28.size();String  anisosi28=null;  if(
statement28.get(0).indexOf("!=") > 0){
           anisosi28 ="!=";
          }else{
           anisosi28="==";
          }
        jTextArea1.append("Statement:" + statement28.get(0)+"\r\n\r\n" );
       jTextArea3.append("Statement:" + statement28.get(0)+"\r\n\r\n" );
       jTextArea4.append("Statement:" + statement28.get(0)+"\r\n\r\n" );
        jTextArea7.append("Statement:" + statement28.get(0)+"\r\n\r\n" );
        jTextArea4.append(printcases2("Result","null",anisosi28,"PASS"));
        jTextArea7.append(printcases2("Result","null",anisosi28,"FAIL"));
```

```
      for(int cv01 =0; cv01<size1;cv01++) {

     //              jTextArea1.append("Type:"+"Integer " + "Name: Value:
"+numbers01.get(0).get(cv01) +"\r\n");// + " Compilation:" + "");
           if(passorfail28.get(cv01)==  1){jTextArea1.append("Loop: " + cv01
+"\r\nType:"+"Integer " + "Name:  Value: "+numbers01.get(0).get(cv01) +"\r\n");
              jTextArea1.append("Run Time: " + "PASS"+"\r\n");
           }else if(passorfail28.get(cv01) == 0){jTextArea3.append("Loop: " + cv01
+"\r\nType:"+"Integer " + "Name:  Value: "+numbers01.get(0).get(cv01) +"\r\n");
              jTextArea3.append("Run Time: " + "FAIL"+"\r\n");
      }

       //jTextArea1.append("\r\n"); jTextArea3.append("\r\n");
   }
}

    public void print29(ArrayList<String> statement29,ArrayList<Integer>
passorfail29,ArrayList<ArrayList<Integer>> tablesize29,ArrayList<String>
names29,String anisosi29){

       int size1 = passorfail29.size();
      jTextArea1.append("Statement:" + statement29.get(0)+"\r\n\r\n" );
      jTextArea3.append("Statement:" + statement29.get(0)+"\r\n\r\n" );
      jTextArea4.append("Statement:" + statement29.get(0)+"\r\n\r\n" );
       jTextArea7.append("Statement:" + statement29.get(0)+"\r\n\r\n" );

jTextArea4.append(printcases2(names29.get(0)+".length",names29.get(1)+".length",ani
sosi29,"PASS"));

jTextArea7.append(printcases2(names29.get(0)+".length",names29.get(1)+".length",ani
sosi29,"FAIL"));
       for(int cv29 =0; cv29<glcounter;cv29++) {
        if(passorfail29.get(cv29) == 1){
              jTextArea1.append("Loop: " + cv29 +"\r\nName:" +names29.get(0)+
".length Value: "+tablesize29.get(0).get(cv29) +"\r\n");// + " Compilation:" + "");
              jTextArea1.append("Name:" +names29.get(1)+ ".length Value:
"+tablesize29.get(1).get(cv29) +"\r\n");
              jTextArea1.append("Run Time: " + "PASS"+"\r\n\r\n");}
          else if(passorfail29.get(cv29) == 0){
              jTextArea3.append("Loop: " + cv29 +"\r\nName:" +names29.get(0)+
".length Value: "+tablesize29.get(0).get(cv29) +"\r\n");// + " Compilation:" + "");
```

```
            jTextArea3.append("Name:" +names29.get(1)+ ".length Value: "+tablesize29.get(1).get(cv29) +"\r\n");
            jTextArea3.append("Run Time: " + "FAIL"+"\r\n\r\n");}
        }
}
    public void print30(ArrayList<String> statement30,ArrayList<Integer> passorfail30,ArrayList<String> names30){

      int size1 = statement30.size();
     jTextArea1.append("Statement:" + statement30.get(0)+"\r\n\r\n" );
     jTextArea3.append("Statement:" + statement30.get(0)+"\r\n\r\n" );
     jTextArea4.append("Statement:" + statement30.get(0)+"\r\n" );
     jTextArea7.append("Statement:" + statement30.get(0)+"\r\n" );
     jTextArea4.append("Case 1: Type of " +names30.get(0)+" == Type of " +names30.get(1)+"\r\n");
     jTextArea7.append("Case 2: Type of " +names30.get(0)+" != Type of " +names30.get(1)+"\r\n");

       if(passorfail30.get(0) == 1){
            jTextArea1.append("Loop: 0"+"\r\nName:" +names30.get(0)+ ".length Type:" +names30.get(2)+"\r\n");// + " Compilation:" + "");
            jTextArea1.append("Name:" +names30.get(1)+ ".length Type:"+names30.get(3)+ "\r\n");
            jTextArea1.append("Run Time: " + "PASS"+"\r\n\r\n");}
         else if(passorfail30.get(0) == 0){
            jTextArea3.append("Loop: 0" +"\r\nName:" +names30.get(0)+ ".length Type:"+names30.get(2)+ " \r\n");// + " Compilation:" + "");
            jTextArea3.append("Name:" +names30.get(1)+ ".length Type:"+names30.get(3)+ "  \r\n");
            jTextArea3.append("Run Time: " + "FAIL"+"\r\n\r\n");}

}
    public void print31(ArrayList<String> statement31, ArrayList<Integer> passorfail31, ArrayList<String> names31, ArrayList<ArrayList<Integer>> variables31){
//names31 variables31 statement31 passorfail31  //@ensures \result == ((d == 0) ? \old(r) : \old(r) / d) ;

          //0= d  //1=r //2=result
      int size1 = statement31.size();
     jTextArea1.append("Statement:" + statement31.get(0)+"\r\n\r\n" );
     jTextArea3.append("Statement:" + statement31.get(0)+"\r\n\r\n" );
```

```
        jTextArea4.append("Statement:" + statement31.get(0)+"\r\n" );
        jTextArea7.append("Statement:" + statement31.get(0)+"\r\n" );
        jTextArea4.append("Case 1: (IF "+names31.get(0)+"==0 then Result== "
+names31.get(1)+") OR (IF ("+names31.get(0)+"!=0 then Result== "+names31.get(1)+
"/(" +names31.get(0)+"+1) ) -PASS\r\n");
        jTextArea7.append("Case 2: (IF "+names31.get(0)+"==0 then Result!= "
+names31.get(1)+") OR (IF ("+names31.get(0)+"!=0 then Result!= "+names31.get(1)+
"/(" +names31.get(0)+"+1) ) - FAIL\r\n");

    for(int cv31 = 0 ; cv31 < glcounter; cv31 ++){
        if(passorfail31.get(cv31) == 1){
            jTextArea1.append("Loop: " + cv31 +"\r\nName:" +names31.get(0)+ "
Value: " +variables31.get(0).get(cv31)+"\r\n");
            jTextArea1.append("Name:" +names31.get(1)+ " Value: "
+variables31.get(1).get(cv31)+"\r\n");
            //jTextArea1.append("Name:" +names31.get(2)+ " Value: "
+variables31.get(2).get(cv31)+"\r\n");
            jTextArea1.append("Result: Value:"+variables31.get(2).get(cv31)+ "\r\n");
            jTextArea1.append("Run Time: " + "PASS"+"\r\n\r\n");}
        else if(passorfail31.get(cv31) == 0){
            jTextArea3.append("Loop: " + cv31 +"\r\nName:" +names31.get(0)+ "
Value: " +variables31.get(0).get(cv31)+"\r\n");
            jTextArea3.append("Name:" +names31.get(1)+ " Value: "
+variables31.get(1).get(cv31)+"\r\n");
            //jTextArea3.append("Name:" +names31.get(2)+ " Value: "
+variables31.get(2).get(cv31)+"\r\n");
            jTextArea3.append("Result: Value:"+variables31.get(2).get(cv31)+ "\r\n");
            jTextArea3.append("Run Time: " + "FAIL"+"\r\n\r\n");}
      }
}
   public void print32(ArrayList<String> statement32,ArrayList<ArrayList<Integer>>
passorfail32, ArrayList<ArrayList<String>> names32,  ArrayList<ArrayList<String>>
type32){
   //names31  variables31 statement31  passorfail31
              //0= d  //1=num //2=r //3=result
         int size1 = statement32.size();

      for(int cv32 = 0 ; cv32 < size1; cv32 ++){
        jTextArea1.append("Statement:" + statement32.get(cv32)+"\r\n\r\n" );
        jTextArea3.append("Statement:" + statement32.get(cv32)+"\r\n\r\n" );
        jTextArea4.append("Statement:" + statement32.get(cv32)+"\r\n" );
        jTextArea7.append("Statement:" + statement32.get(cv32)+"\r\n" );
```

```
        jTextArea4.append("Case 1: Type of "+names32.get(cv32).get(0)+" == Type of
Oject[] - PASS\r\n");
        jTextArea7.append("Case 2: Type of "+names32.get(cv32).get(0)+" != Type of
Oject[] - FAIL\r\n");
       for(int cv32b = 0 ; cv32b < 1; cv32b ++){
          if(passorfail32.get(cv32).get(0) == 1){
              jTextArea1.append("Loop:  "  +  cv32  +"\r\nName:"
+names32.get(cv32).get(0)+ "\r\n Type: " +type32.get(cv32).get(0)+"\r\n");
              jTextArea1.append("Run Time: " + "PASS"+"\r\n\r\n");}
          else if(passorfail32.get(cv32).get(0) == 0){
              jTextArea3.append("Loop:  "  +  cv32  +"\r\nName:"
+names32.get(cv32).get(0)+ "\r\n Type: " +type32.get(cv32).get(0)+"\r\n");
              jTextArea3.append("Run Time: " + "FAIL"+"\r\n\r\n");}
         }
      }
   }
   public void print33(ArrayList<String> statement33, ArrayList<Integer> passorfail33,
ArrayList<String> names33,ArrayList<String> result33){
   //names31  variables31 statement31  passorfail31
              //0= d  //1=num //2=r //3=result
         int size1 = statement33.size();
        jTextArea1.append("Statement:" + statement33.get(0)+"\r\n\r\n" );
        jTextArea3.append("Statement:" + statement33.get(0)+"\r\n\r\n" );
        jTextArea4.append("Statement:" + statement33.get(0)+"\r\n" );
        jTextArea7.append("Statement:" + statement33.get(0)+"\r\n" );
        jTextArea4.append("Case 1:   "+names33.get(2)+ " && "+ names33.get(3)+ "
&& "+ names33.get(4)+ " && "+ names33.get(5)+ "  == TRUE - PASS\r\n");
        jTextArea7.append("Case 2:   "+names33.get(2)+ " && "+ names33.get(3)+ "
&& "+ names33.get(4)+ " && "+ names33.get(5)+ "  == FALSE - FAIL\r\n");

      for(int cv32 = 0 ; cv32 < size1; cv32 ++){
          if(passorfail33.get(cv32) == 1){
              jTextArea1.append("Loop: " + cv32 +"\r\n"+ result33.get(cv32) + "\r\n");
              jTextArea1.append("Run Time: " + "PASS"+"\r\n\r\n");}
          else if(passorfail33.get(cv32) == 0){
              jTextArea3.append("Loop: " + cv32 +"\r\n" + result33.get(cv32) + "\r\n");
              jTextArea3.append("Run Time: " + "FAIL"+"\r\n\r\n");}
        }
   }
   public String printcases(String a,String b,String anisosi,String passorfail){
      String s=null;
      if(passorfail.equals("PASS")){
```

```
    if(anisosi.equals("==")){
       s= "Case 1: " + a + " == " + b+ " - PASS\r\n";
    }else if(anisosi.equals(">=")){
       s= "Case 1: " + a + " >= " + b+ " - PASS\r\n";
    }else if(anisosi.equals("<=")){
       s= "Case 1: " + a + " <= " + b+ " - PASS\r\n";
    }else if(anisosi.equals(">")){
       s="Case 2:"+ a + " > " + b +" - PASS \r\n";
    }else if(anisosi.equals("<")){
       s="Case 2:"+ a + " < " + b +" - PASS \r\n";
    }else if(anisosi.equals("!=")){
       s="Case 2:"+ a + " != " + b +" - PASS \r\n";
    }
}else if(passorfail.equals("FAIL")){
    if(anisosi.equals("==")){
       s="Case 2:"+ a + " > " + b +" - FAIL \r\n";
       s=s+"Case 3:"+ a + " < " + b +" - FAIL \r\n";
    }else if(anisosi.equals(">=")){
      s="Case 2:"+ a + " < " + b +" - FAIL \r\n";
    }else if(anisosi.equals("<=")){
       s="Case 2:"+ a + " > " + b +" - FAIL \r\n";
    }else if(anisosi.equals(">")){
       s= "Case 1: " + a + " == " + b+ " - FAIL\r\n";
       s=s+"Case 3:"+ a + " < " + b +" - FAIL \r\n";
    }else if(anisosi.equals("<")){
       s= "Case 1: " + a + " == " + b+ " - FAIL\r\n";
       s=s+"Case 3:"+ a + " > " + b +" - FAIL \r\n";
    }else if(anisosi.equals("!=")){
       s= "Case 1: " + a + " == " + b+ " - FAIL\r\n";

    }
}
    return s;
}

public String printcases2(String a,String b,String anisosi,String passorfail){
    String s=null;
    if(passorfail.equals("PASS")){
    if(anisosi.equals("==")){
       s= "Case 1: " + a + " == " + b+ " - PASS\r\n";
    }else if(anisosi.equals("!=")){
```

```
        s="Case 1:"+ a + " != " + b +" - PASS \r\n";
      }
   }else if(passorfail.equals("FAIL")){
      if(anisosi.equals("==")){
        s="Case 2:"+ a + " != " + b +" - FAIL \r\n";
      }else if(anisosi.equals("!=")){
        s="Case 2:"+ a + "==" + b +" - FAIL \r\n";
      }
   }
      return s;
   }
      public String printcases3(String a,String b,String anisosi1,String c,String d,String
anisosi2,String passorfail){
  String s=null;
      if(passorfail.equals("PASS")){
      if(anisosi1.equals(">=") && anisosi2.equals("<")){
        s= "Case 1: " + a + " >= " + b+ c + " < " + d+ " - PASS\r\n";
      }else if(anisosi1.equals("<=") && anisosi2.equals("<")){
        s= "Case 1: " + a + " <= " + b+ c + " < " + d+ " - PASS\r\n";
      }else if(anisosi1.equals("<=") && anisosi2.equals("<=")){
        s= "Case 1: " + a + " <= " + b+ c + " <= " + d+ " - PASS\r\n";
      }
   }else if(passorfail.equals("FAIL")){
     if(anisosi1.equals(">=") && anisosi2.equals("<")){
        s= "Case 2: " + a + " < " + b+ " - FAIL\r\n";
        s=s+ "Case 3: " + c + " >= " + d+" - FAIL\r\n";
     }else if(anisosi1.equals("<=") && anisosi2.equals("<")){
        s= "Case 2: " + a + " > " + b+ " - FAIL\r\n";
        s=s+ "Case 3: " + c + " >= " + d+" - FAIL\r\n";
      }else if(anisosi1.equals("<=") && anisosi2.equals("<=")){
        s= "Case 2: " + a + " > " + b+ " - FAIL\r\n";
        s=s+ "Case 3: " + c + " > " + d+" - FAIL\r\n";
      }
   }
      return s;
   }
      public String printcases4(String a,String b,String anisosi1,String c,String d,String
anisosi2,String passorfail){
      String s=null;
      if(passorfail.equals("PASS")){
      if(anisosi1.equals("==") && anisosi2.equals("==")){
```

```
        s= "Case 1: " + a + " == " + b+ c + " == " + d+ " - PASS\r\n";
      }else if(anisosi1.equals("!=") && anisosi2.equals("==")){
        s= "Case 1: " + a + " != " + b+ c + " == " + d+ " - PASS\r\n";
      }else if(anisosi1.equals("==") && anisosi2.equals("!=")){
        s= "Case 1: " + a + " == " + b+ c + " != " + d+ " - PASS\r\n";
      }else if(anisosi1.equals("!=") && anisosi2.equals("!=")){
        s= "Case 1: " + a +   " == " + b+ c + " == " + d+ " - PASS\r\n";
      }
    }else if(passorfail.equals("FAIL")){
       if(anisosi1.equals("==") && anisosi2.equals("==")){
        s= "Case 2: " + a + " != " + b+ c + " == " + d+ " - FAIL\r\n";
        s=s+ "Case 3: " + a + " == " + b+ c + " != " + d+ " - FAIL\r\n";
        s=s+ "Case 4: " + a + " != " + b+ c + " != " + d+ " - FAIL\r\n";
      }else if(anisosi1.equals("!=") && anisosi2.equals("==")){
        s= "Case 2: " + a + "== " + b+ c + " == " + d+ " - FAIL\r\n";
        s=s+ "Case 3: " + a + " == " + b+ c + " != " + d+ " - FAIL\r\n";
        s=s+ "Case 4: " + a + " != " + b+ c + " != " + d+ " - FAIL\r\n";
      }else if(anisosi1.equals("==") && anisosi2.equals("!=")){
        s= "Case 2: " + a + " != " + b+ c + " == " + d+ " - FAIL\r\n";
        s=s+ "Case 3: " + a + " == " + b+ c + " == " + d+ " - FAIL\r\n";
        s=s+ "Case 4: " + a + " != " + b+ c + " != " + d+ " - FAIL\r\n";
      }else if(anisosi1.equals("!=") && anisosi2.equals("!=")){
        s= "Case 2: " + a + " != " + b+ c + " == " + d+ " - FAIL\r\n";
        s=s+ "Case 3: " + a + " == " + b+ c + " != " + d+ " - FAIL\r\n";
        s=s+ "Case 4: " + a + " == " + b+ c + " == " + d+ " - FAIL\r\n";
      }
    }
       return s;
    }


    public String StringContainsThis(String strthis){
      return strthis;

    }



   public void completenumbers( ArrayList<ArrayList<Integer>> tables111, int
number111 , ArrayList<Integer> tablesize111){
       for(int cv111 =0; cv111 < number111; cv111++){
```

```
            tables111.add(new ArrayList<Integer>());
                int tablezise= intRandomInitialize();
               tablesize111.add(tablezise);
             if(tablezise <0){
              InitilizeTable(0, tables111.get(tables111.size() -1 ));
             }else{
              InitilizeTable(tablezise, tables111.get(tables111.size() -1 ));}
        }
    }
        public void completenumbersfixsize( ArrayList<ArrayList<Integer>> tables111,
int number111 , int size){
        for(int cv111 =0; cv111 < number111; cv111++){
             tables111.add(new ArrayList<Integer>());
              int tablezise= size;
              InitilizeTable(tablezise, tables111.get(tables111.size() -1 ));
        }
    }
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CheckJML().setVisible(true);

            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JMenuItem Exit;
    private javax.swing.JMenuItem Open;
    private javax.swing.JFileChooser fileChooser;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JComboBox jComboBox2;
    private javax.swing.JComboBox jComboBox3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
```

```
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JScrollPane jScrollPane7;
    private javax.swing.JScrollPane jScrollPane8;
    private javax.swing.JScrollPane jScrollPane9;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextArea jTextArea3;
    private javax.swing.JTextArea jTextArea4;
    private javax.swing.JTextArea jTextArea5;
    private javax.swing.JTextArea jTextArea6;
    private javax.swing.JTextArea jTextArea7;
    private javax.swing.JTextPane jTextPane2;
    private javax.swing.JTextPane jTextPane3;
    // End of variables declaration

}
```

Παράρτημα Β

Κώδικες Υλοποίησης JML Templates

ParserTemplates.java

```
public class ParserTemplates {

   public String devidebyzero1(String str){



      str="//@requires a!=0;\r\n//@ensures \\result != 0;\r\n//@requires b!=0;\r\n\r\npublic static int Divide(int a, int b){\r\n\r\nint apot;\r\nif(a>=b){\r\napot = a/b;\r\n}\r\nelse\r\n{\r\napot = b/a;\r\n}\r\nreturn apot;\r\n}";



      return str;

   }

   public String ArrayIndexOutRange2(String str){

      str="/*@spec_public@*/int c;\r\n/*@spec_public@*/int a[];\r\n//@requires a!=null;\r\n//@assignable c;\r\n//@ensures (\\forall int i; 0<= i && i < a.length; c==a[a.length - 1]);\r\n\r\npublic int checkIndex(int a[]) {\r\n\r\n//@assert(\\forall int i,j; 0<=i && i<=j && j<a.length; i<a.length);\r\nfor (int i = 0; i <= a.length; i++) {\r\nc = a[i];\r\n}\r\n\r\nreturn c;\r\n}\r\n";



      return str;

   }

   public String StringIndexOutRange(String str){

      str="/*@spec_public@*/String str[];\r\n//@requires str!=null;\r\n//@assignable \\nothing;\r\n\r\npublic void checkStrIndex(){\r\n\r\n//@assert(\\forall int i,j; 0<=i && i<=j && j<str.length; i<str.length);\r\nfor(int i=0;i<=str.length;i++){\r\nstr[i]=\"Finally\";\r\n}\r\n\r\n}";



      return str;

   }

    public String ArrayStoreEcxeption(String str){
```

```
str="/*@spec_public non_null@*/ Object[] elems;\r\n/*@spec_public@*/ int top;\r\n//@invariant 0 <= top && top < elems.length && \\typeof(elems) == \\type(Object[]);\r\n//@requires top < elems.length;\r\n/*@requires \\typeof(o) <: \\elemtype(\\typeof(elems));@*/\r\n\r\npublic void add(Integer o){\r\n\r\nelems[top++] = o;\r\n\r\n}";

    return str;

}

public String DereferencingNull(String str){

str="/*@spec_public@*/ int size;\r\n/*@non_null@*/ int[] elements;\r\n//@invariant 0<=size && size <= elements.length;\r\n//@requires input!=null; \r\n\r\npublic void adding(int[] input) {\r\n size = input.length;\r\n elements = new int[size];\r\n System.arraycopy(input, 0, elements, 0, size);\r\n for(int i =0; i<input.length;i++){\r\n System.out.print(\" \" + input[i]);\r\n }\r\n}\r\n\r\npublic int extractMin(){\r\n int min =Integer.MAX_VALUE;\r\n int minIndex = 0;\r\n for (int i=0; i<size;i++) {\r\n if(elements[i] < min){\r\n min = elements[i];\r\n //@assert min>0;\r\n minIndex = i;\r\n }\r\n }\r\n size--;\r\n elements[minIndex] = elements[size];\r\n return min;\r\n}";

return str;

}

public String StackOverflowArea(String str){

str="/*@requires 1 <= n;\r\n@ensures \\result== (\\product int i ; 1<= i && i<=n ;i); &&\\result<6227020800;\r\n@requires n<13;\r\n*/\r\n\r\npublic static int factorial(int n){\r\n if(n>12) throw new ArithmeticException();\r\n else {\r\n int f = 1;\r\n int i = 1; }\r\n\r\n \r\n while (i<n) {\r\n i = i + 1;\r\n f = f * i;\r\n }\r\n return f;\r\n }\r\n}";

    return str;

}

public String UnsortedTable(String str){

str="public int i;\r\n//@requires A!=null;\r\n//@requires A.lenght > 0;\r\n//@ requires(\\forall int i, j;0 <= i && i < j && j < A.length; A[i] <= A[j]);\r\n\r\npublic int binarySearch(int[] A, int N) {\r\n\r\nint lowestPossibleLoc = 0;\r\nint highestPossibleLoc = A.length - 1;\r\nint middle =0; \r\n\r\n\r\nwhile(highestPossibleLoc >= lowestPossibleLoc){\r\n middle = (highestPossibleLoc + lowestPossibleLoc) / 2;\r\n if(A[middle] == N) {\r\n return middle;\r\n } else if (A[middle] > N) {\r\n highestPossibleLoc = middle - 1;\r\n } else {\r\n lowestPossibleLoc = middle + 1;\r\n }\r\n}\r\nreturn -1;\r\n}";

//\r\n//@ensures (\\forall int i ;0 <= i && i < A.length ==> \\old(A[i]== N)) ? (\\result==i) : (\\result== -1);
```

```
 return str;

}

 public String InfiniteLoop(String str){

 str="//@requires number > 0;\r\n//@requires num > 0;\r\n//@assignable number;\r\n\r\npublic int Compute(int number, int num){\r\n/*@assert(\\forall int i; 0<=i && i<(\\old(number)-\\old(num)); number==\\old(number)-1);@*/\r\nwhile(number > num){\r\n number++;\r\n System.out.println(\"Done\");\r\n}\r\nreturn number;\r\n}\r\n\r\n" ;

        return str;

 }

 public String InfiniteLoopException(String str){

 str="public int count;\r\nprivate /*@spec_public@*/ int i;\r\n/*@requires ia != null;\r\n@assignable ia[*];\r\n@assignable count;*/\r\n/*@ensures(\\forall int i; 0<=i && i <ia.length ==> (\\forall int j; 0<=j && j<i ==> (\\old(ia[j])>5))) ? (count==count+1) : (ia[i]== 0); @*/\r\n\r\npublic int negatefirst(int ia[]){ \r\nfor(int i=0; i< ia.length; i++) {\r\n if(ia[i] < 5){\r\n count++;\r\n }\r\n else{\r\n ia[i] = 0; }\r\n}\r\nreturn count; \r\n}";

       return str;

 }

 public String CallBack(String str){

 str="private/*@spec_public@*/int k;\r\nprivate/*@spec_public@*/int m;\r\n/*@ invariant\r\n @k + m ==0;\r\n @*/ \r\n/*@ normal_behavior\r\n @ requires true;\r\n @assignable k,m;\r\n @ensures k== \\old(k) - 1 &&\r\n @ m == \\old(m) +1;\r\n @*/\r\npublic void decrementk(){\r\n k--;\r\n m++;\r\n}\r\nB b;\r\n/*@ normal_behavior\r\n @ requires b !=null;\r\n @assignable k,m;\r\n @ensures true;\r\n @*/ \r\npublic void increment(){\r\n k++;\r\n b.go(this); m--;\r\n}\r\nclass B{\r\n/*@ normal behavior\r\n @ requires b !=null;\r\n @assignable k,m;\r\n @ensures true;\r\n@*/ \r\n}\r\nvoid go(Callback arg){\r\n arg.decrementk();\r\n}";

        return str;

 }

 public String NegativeArraySize(String str){

  str="//@ requires size > 0;\r\n//@ensures \\result!=null;\r\n\r\npublic int[] NegArSize(int size){\r\n int pinakas[]=new int[size];\r\n\r\n for (int i = 0; i < size; i++){\r\n pinakas[i]=i;\r\n }\r\n return pinakas;\r\n}";

       return str;
```

```
}

public String SizeOfParallelTables(String str){

str="//@requires a!=null;\r\n//@requires b!=null;\r\n//@assignable \\nothing;\r\n//@requires a.lenght == b.lenght ;\r\n\r\npublic void checksize(int a[], int b[]) { \r\nint i=0; \r\nwhile(i<a.length){ \r\nSystem.out.println(\"Value of 1st table:\"+ a[i] + \"Value of 2nd table:\" + b[i]); \r\ni++; \r\n}\r\n}";

return str;

}

public String DiffTypeEquationException(String str){

str="\r\n/*@spec_public@*/int num;\r\n/*@spec_public@*/String str;\r\n/*@requires \\typeof(num) <: \\typeof(str);@*/\r\n\r\npublic void add() {\r\n\r\n str = str + num;\r\n System.out.println(\"Result = \" + str);\r\n\r\n}";

return str;

}


public String CatchingException(String str){

str="int r;\r\nint num;\r\n//@assignable r;\r\n//@ensures \\result == ((d == 0) ? \\old(r) : \\old(r) / d) ;\r\n\r\nint m(int d) {\r\n try {\r\n return r / d;\r\n } catch (Exception e) {\r\n return r / (d + 1);\r\n}\r\n}";

return str;

}

public String CallbyValueOrReference(String str){

str="int numa;\r\nInteger s[];\r\n\r\n/*@requires \\typeof(numa) <: \\type(Object[]);@*/\r\npublic void reference1(int numa) {\r\n\r\n numa = 2;\r\n\r\n}\r\n\r\n/*@requires \\typeof(numb) <: \\type(Object[]);@*/\r\npublic void reference2(Integer[] numb) {\r\n\r\n numb[0] = new Integer(2);\r\n\r\n}";

return str;

}

public String StaticInitialisation(String str){

str="\r\nclass C1 {\r\nstatic boolean b1 = C2.d2;\r\nstatic boolean d1 = true;\r\n}\r\n\r\nclass C2 {\r\nstatic boolean d2 = true;\r\nstatic boolean b2 = C1.d1;\r\n}\r\n\r\npublic class C {\r\n static boolean r1, r2, r3, r4;\r\n\r\n/*@ normal_behavior\r\n@ assignable \\everything;\r\n@*/\r\n//@ensures r1 && r2 && r3
```

```
&& r4;\r\n\r\nstatic void m(){\r\n r1 = C1.b1;\r\n r2 = C2.b2;\r\n r3 = C1.d1;\r\n r4 = C2.d2;\r\n}";

        return str;

    }

//////////////////////////////////////SINGLE STATEMENTS////////////////////////////////


    public void RequiresVariable(String[] str){

        str[0] ="//@requires a != 0 ;\r\n";

        str[1] ="//@requires [Variable]  [!= | == | <= | >=]  [Digit] ; \r\n";



    }

    public void RequiresNullHandle(String[] str){

         str[0] ="//@requires a!=null; ;\r\n";

        str[1]=  "//@requires [Variable]  [!= | == ]  null ; \r\n";



    }



    public void EnsuresTableHasValues(String[] str){

          str[0] ="//@ensures (\\forall int i; 0 <= i && i < a.length; c==a[a.length-1]);\r\n\r\npublic void checkIndex(int a[]) {\r\nfor (int i = 0; i <= a.length; i++) {\r\n c = a[i];\r\n}\r\n}";

          str[1] ="//@ensures (\\forall int [Variable1]; 0 <=[Variable1] && i < [Variable2].length; [Variable3]==[Variable2][Variable2.length - 1]);\r\n\r\npublic void [MethodName](int []) {\r\n for (int [Variable1] = 0; [Variable1] <= [Variable2].length; [Variable1]++) {\r\n  [Variable3] = [Variable2] [[Variable1]];\r\n }\r\n} ";

    }

    public void Invariant_CheckType(String[] str){

          str[0] ="/*@spec_public non_null@*/ Object[] elems;\r\n/*@spec_public@*/ int top;\r\n/*@invariant 0 <= top && top < elems.length && \\typeof(elems) == \\type(Object[]);\r\n\r\n";
```

```
        str[1] ="/*@spec_public non_null@*/ [Type1][] [Variable1];\r\n/*@spec_public@*/ int [Variable2];\r\n/*@invariant 0 <= [Variable2] && [Variable2] < [Variable1].length && \\typeof([Variable1]) == \\type([Type1][]);}";



    }

    public void RequiresDigit_toTableLength(String[] str){

        str[0] ="//@requires top < elems.length; \r\n";

        str[1]= "//@requires //@requires [Variable1] [!= | == | <= | >=] [Variable2].length;\r\n";



    }

    public void VariableTypes_insideTableTypes(String[] str){

        str[0] ="/*@spec_public non_null@*/ Object[] elems;\r\n/*@spec_public@*/ int top;\r\n/*@requires \\typeof(o) <: \\elemtype(\\typeof(elems));@*/\r\n\r\npublic void add(Integer o){";

        str[1] ="/*@spec_public non_null@*/ [Type1][] [Variable1];\r\n/*@spec_public@*/ int [Variable2];\r\n/*@requires \\typeof([Variable1]) <: \\elemtype(\\typeof([Variable2]));@*/\r\n\r\npublic void [Name]([Type] [Variable1]){";



    }

    public void CheckingCounter_TobeInsideTableSize(String[] str){ //8

        str[0] ="//@invariant 0<=size && size <= elements.length;\r\n";

        str[1]= "//@invariant 0<=[Variable1] && [Variable1] <= [Variable2].length;\r\n";



    }

    public void RequiresDigit(String[] str){  //15

        str[0] ="/*@requires 1 <= n;@*/;\r\n";

        str[1] ="/*@requires  [Digit] [!= | == | <= | >=]  [Variable] ;@*/ \r\n";
```

```
}

    public void Factorial(String[] str){ //16

    str[0] ="/*@ensures \\result== (\\product int i ; 1<= i && i<=n ;i); &&\\result<6227020800;\r\n";

    str[1] ="/*@ensures \\result== (\\product int [variable1] ; 1<= [variable1] && [variable1]<=[variable2] ;[variable1]); &&\\result<6227020800 \r\n";


}

public void RequiresTableLengthtoDigit(String[] str){ //17

    str[0] ="//@requires A.lenght > 0; \r\n";

    str[1] ="//@requires [Variable].lenght  [!= | == | <= | >=]  [Digit] ; \r\n";


}

public void RequiresSortedTable(String[] str){ //19

    str[0] ="//@requires(\\forall int i, j;0 <= i && i < j && j < A.length; A[i] <= A[j]); \r\n";

    str[1] ="//@requires(\\forall int [Variable1], j;0 <= [Variable1] && [Variable1] < j && j < [Variable2].length; [Variable2][[ Variable1]] [<= | >= ] [Variable2][j]); \r\n";


}

public void EnsuresFindMinValuesFromTable(String[] str){ //22

    str[0] ="/*@ensures(\\forall int i; 0<=i && i <ia.length ==>(\\forall int j; 0<=j && j<i ==> (\\old(ia[j])>5))) ? (count==count+1) : (ia[i]== 0); @*/\r\n\r\npublic int negatefirst(int ia[]){\r\nfor(int i=0; i< ia.length; i++) {\r\nif(ia[i] < 5){\r\n count++;\r\n}\r\nelse{\r\n ia[i] = 0; }\r\n}\r\nreturn count;\r\n}";

    str[1] ="/*@ensures(\\forall int [Variable1]; 0<=[ Variable1] && [Variable1] <[ Variable2].length ==> (\\forall int j; 0<=j && j<[ Variable1] ==> (\\old(ia[j])>[Digit]))) ? ([Variable3]== [Variable3] +1) : ([Variable2][[Variable1]]== 0)); @*/\r\n\r\npublic int [name](int [Variable2][]){\r\nfor(int [Variable1]=0; [Variable1]< [Variable2].length; [Variable1]++) {\r\nif([Variable2][[Variable1]] > [Digit]){\r\n [Variable3]++;\r\n}\r\nelse{\r\n [Variable2][[Variable1]]= 0 }\r\n}\r\nreturn [Variable3];\r\n}";
```

```
}

  public void EnsuresResultNotZero(String[] str){ //25

    str[0] ="//@ensures \\result != 0;\r\n\r\npublic static int Divide(int a, int b){\r\n\r\nint apot;\r\nif(a>=b){\r\napot = a/b;\r\n}\r\nelse\r\n{\r\napot = b/a;\r\n}\r\nreturn apot;\r\n}";

    str[1] ="//@ensures \\result [!= ] [Digit];\r\n\r\npublic static int Divide(int [Variable1], int [Variable2]){\r\n\r\nint [Variable3];\r\nif([Variable1]>= [Variable2]){\r\n[Variable3] = [Variable1]/ [Variable2];\r\n}\r\nelse\r\n{\r\n[Variable3] = [Variable2]/ [Variable1];\r\n}\r\nreturn [Variable3];\r\n}";



}

public void AssertCounter_insideTableBoundaries(String[] str){  //26

    str[0] ="//@assert(\\forall int i,j; 0<=i && i<=j && j<a.length; i<a.length);\r\nfor (int i = 0; i <= a.length; i++) {\r\nc = a[i];\r\n}";

    str[1] ="//@assert(\\forall int [Variable1],j; 0<=[Variable1] && i<=j && j<[ Variable2].length; [Variable1]<[ Variable2].length);\r\nfor (int [Variable1] = 0; [Variable1] [<= | <] [Variable2].length; [Variable1]++) {\r\n[Variable3] = [Variable2] [[Variable1]];\r\n}";



}

public void AssertWhile_Counters(String[] str){ //27

    str[0] ="/*@assert(\\forall int i; 0<=i && i<(\\old(number)-\\old(num)); number==\\old(number)-1);@*/\r\nwhile(number > num){\r\nnumber++;\r\n";

    str[1] ="/*@assert(\\forall int i; 0<=i && i<(\\old([Variable1])-\\old([Variable2])); [Variable1]==\\old(Variable1)-1);@*/\r\nwhile([Variable1] > [Variable2]{\r\n[Variable1]++;";



}

public void EnsuresInitializeTable(String[] str){ //28

    str[0] ="//@ensures \\result != null;\r\n\r\npublic int[] NegArSize(int size){\r\nint pinakas[]=new int[size];\r\nreturn pinakas;\r\n}";
```

```
        str[1] ="//@ensures \\result != null;\r\n\r\npublic int[] NegArSize(int [Variable1]){\r\nint [Variable2][]=new int[[Variable1]];\r\nreturn [Variable2];\r\n}";




    }

    public void RequiresSameTableLength(String[] str){

        str[0] ="//@requires a.length == b.length ;\r\n";

        str[1] ="//@requires [Variable1].length == [Variable2].length ;\r\n";



    }

    public void RequiresSameType(String[] str){

        str[0] ="/*@spec_public@*/int num;\r\n/*@spec_public@*/String str;\r\n/*@requires \\typeof(num) <: \\typeof(str);@*/";

        str[1] ="/*@spec_public@*/[Type1] [Variable1];\r\n/*@spec_public@*/[Type2] [Variable2];\r\n/*@requires \\typeof([Variable1];) <: \\typeof([Variable2];);@*/";



    }

    public void EnsuresExceptions(String[] str){

        str[0] ="//@ensures \\result == ((d == 0) ? \\old(r) : \\old(r) / d) ;\r\nint m(int d) { \r\ntry {  \r\nreturn r / d; \r\n} catch (Exception e) { \r\n return r / (d + 1);\r\n}\r\n}";

        str[1] ="//@ensures \\result == (([Variable1] == 0) ? \\old([Variable3]) : \\old([Variable3]) /[Variable1] ) ;\r\nint m(int [Variable1]) { \r\ntry { \r\n return [Variable3] /[Variable1]; \r\n} catch (Exception [Variable4]) {  \r\nreturn [Variable3] / ([Variable1] + 1);\r\n}\r\n}";



    }

    public void RequiresCallingbyObject_Reference(String[] str){

        str[0] ="/*@requires \\typeof(numb) <: \\type(Object[]);@*/\r\npublic void reference2(Integer[] numb) {  \r\nnumb[0] = new Integer(2);\r\n}";
```

```
        str[1] ="/*@requires \\typeof([Variable1]) <: \\type(Object[]);@*/\r\npublic
void reference2([Type1] [] [Variable1] {\r\n}";



    }



  public String devidebyzero12(String str){



      str="public static int Divide(int a, int b){\r\n\r\nint apot;\r\nif(a>=b){\r\napot =
a/b;\r\n}\r\nelse\r\n{\r\napot = b/a;\r\n}\r\nreturn apot;\r\n}";



      return str;

  }

  public String ArrayIndexOutRange22(String str){

      str="int c;\r\nint a[];\r\n\n\r\npublic int checkIndex(int a[]) {\r\nfor (int i = 0; i <=
a.length; i++) {\r\nc = a[i];\r\n}\r\n\r\nreturn c;\r\n}\r\n";



      return str;

  }

  public String StringIndexOutRange2(String str){

      str="String str[];\r\n\r\npublic void checkStrIndex(){\r\n\r\nfor(int
i=0;i<=str.length;i++){\r\nstr[i]=\"Finally\";\r\n}\r\n\r\n}";



      return str;

  }

   public String ArrayStoreEcxeption2(String str){

    str="Object[] elems;\r\n int top;\r\npublic void add(Integer o){\r\n\r\nelems[top++]
= o;\r\n\r\n}";

      return str;

  }
```

```
public String DereferencingNull2(String str){

 str="int size;\r\nint[] elements;\r\npublic void adding(int[] input) {\r\n size = input.length;\r\n elements = new int[size];\r\n System.arraycopy(input, 0, elements, 0, size);\r\n for(int i =0; i<input.length;i++){\r\n  System.out.print(\" \" + input[i]);\r\n }\r\n}\r\n\r\npublic int extractMin(){\r\n int min =Integer.MAX_VALUE;\r\n int minIndex = 0;\r\n for (int i=0; i<size;i++) {\r\n  if(elements[i] < min){\r\n  min = elements[i];\r\n  minIndex = i;\r\n  }\r\n }\r\n size--;\r\n elements[minIndex] = elements[size];\r\n return min;\r\n}";

 return str;

}

 public String StackOverflowArea2(String str){

 str="\r\n\r\npublic static int factorial(int n){\r\n if(n>12) throw new ArithmeticException();\r\n else {\r\n  int f = 1;\r\n  int i = 1;\r\n\r\nwhile (i<n) {\r\n  i = i + 1;\r\n  f = f * i;\r\n  }\r\n  return f;\r\n }\r\n}";

      return str;

}

 public String UnsortedTable2(String str){

 str="public int i;\r\n\r\npublic int binarySearch(int[] A, int N) {\r\n\r\nint lowestPossibleLoc = 0;\r\nint highestPossibleLoc = A.length - 1;\r\nint middle =0; \r\n\r\nwhile(highestPossibleLoc >= lowestPossibleLoc){\r\n middle = (highestPossibleLoc + lowestPossibleLoc) / 2;\r\n if(A[middle] == N) {\r\n return middle;\r\n } else if (A[middle] > N) {\r\n  highestPossibleLoc = middle - 1;\r\n } else {\r\n  lowestPossibleLoc = middle + 1;\r\n }\r\n}\r\nreturn -1;\r\n}";

//\r\n//@ensures (\\forall int i ;0 <= i && i < A.length ==> \\old(A[i]== N)) ? (\\result==i) : (\\result== -1);

 return str;

}

 public String InfiniteLoop2(String str){

 str="public int Compute(int number, int num){\r\nwhile(number > num){\r\n number++;\r\n System.out.println(\"Done\");\r\n}\r\nreturn number;\r\n}\r\n\r\n" ;

      return str;

 }

public String InfiniteLoopException2(String str){
```

```
    str="public int count;\r\nprivate int i;\r\n\r\npublic int negatefirst(int ia[]){
\r\nfor(int i=0; i< ia.length; i++) {\r\n if(ia[i] < 5){\r\n  count++;\r\n }\r\n else{\r\n  ia[i]
= 0; }\r\n}\r\nreturn count; \r\n}";

        return str;

    }

   public String CallBack2(String str){

    str="private int k;\r\nprivate int m;\r\npublic void decrementk(){\r\n  k--;\r\n
m++;\r\n}\r\nB b;\r\n/*@ normal_behavior\r\n @ requires b !=null;\r\n @assignable
k,m;\r\n @ensures true;\r\n @*/  \r\npublic void increment(){\r\n k++;\r\n b.go(this); m-
-;\r\n}\r\nclass B{\r\n/*@ normal behavior\r\n @ requires b !=null;\r\n @assignable
k,m;\r\n  @ensures  true;\r\n@*/                \r\n}\r\nvoid  go(Callback  arg){\r\n
arg.decrementk();\r\n}";

          return str;

    }

   public String NegativeArraySize2(String str){

      str="public int[] NegArSize(int size){\r\n int pinakas[]=new int[size];\r\n for (int i
= 0; i < size; i++){\r\n  pinakas[i]=i;\r\n }\r\n return pinakas;\r\n}";

        return str;

    }

     public String SizeOfParallelTables2(String str){

     str="public void checksize(int a[], int b[]) { \r\nint i=0;\r\nwhile(i<a.length){
\r\nSystem.out.println(\"Value of 1st table:\"+ a[i] + \"Value of 2nd table:\" + b[i]);
\r\ni++; \r\n}\r\n}";

      return str;

     }

     public String DiffTypeEquationException2(String str){

     str="int num;\r\nString str;\r\n\r\npublic void add() {\r\n\r\n  str = str + num;\r\n
System.out.println(\"Result = \" + str);\r\n\r\n}";

      return str;

     }



     public String CatchingException2(String str){
```

```
        str="int r;\r\nint num;\r\n\r\nint m(int d) {\r\n try {\r\n  return r / d;\r\n } catch (Exception e) {\r\n  return r / (d + 1);\r\n}\r\n}";

        return str;

       }

       public String CallbyValueOrReference2(String str){

        str="int numa;\r\nInteger s[];\r\n\r\npublic void reference1(int numa) {\r\n\r\n numa = 2;\r\n\r\n}\r\n\r\npublic void reference2(Integer[] numb) {\r\n\r\n  numb[0] = new Integer(2);\r\n\r\n}";

       return str;

       }

       public String StaticInitialisation2(String str){

        str="\r\nclass C1 {\r\nstatic boolean b1 = C2.d2;\r\nstatic boolean d1 = true;\r\n}\r\n\r\nclass C2 {\r\nstatic boolean d2 = true;\r\nstatic boolean b2 = C1.d1;\r\n}\r\n\r\npublic class C {\r\n static boolean r1, r2, r3, r4;\r\n\r\nstatic void m(){ \r\nr1 = C1.b1;\r\n r2 = C2.b2;\r\n r3 = C1.d1;\r\n r4 = C2.d2;\r\n}";

        return str;

       }
}
```