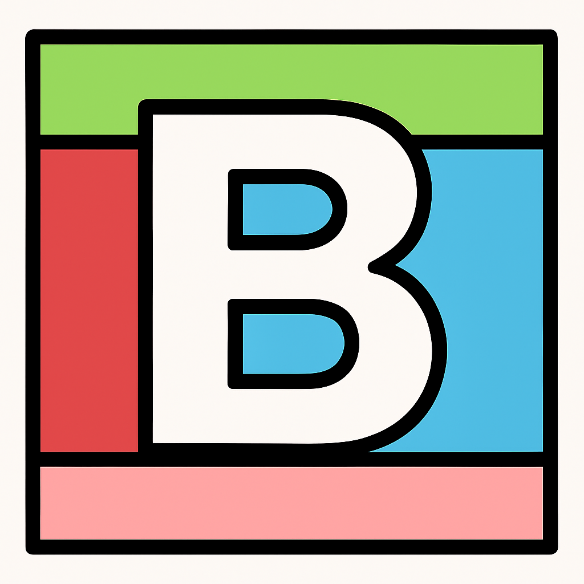
**BLOCKO-v1.0**



**Μέλη Ομάδας:**

**Όνομα**: Κωνσταντίνος

**Επώνυμο**: Καρακίτσος

**ΑΜ**: 1093381

**Όνομα**: Βασίλειος

**Επώνυμο**: Μιχανετζής

**ΑΜ:** 1093433

**Όνομα**: Νυμφοδώρα

**Επώνυμο:** Μπουζιάνη

**ΑΜ**: 1097429

**Όνομα**: Μιχαήλ

**Επώνυμο:** Σοκολάκης

**ΑΜ**: 1072589

**Όνομα**: Σωτήριος

**Επώνυμο**: Χασιώτης

**ΑΜ**: 1093510

**Περιεχόμενα**

[**Project-description-v1.0** 5](#_Toc198628681)

[**Use-cases-v1.0** 16](#_Toc198628675)

[**Robustness-diagram-v1.0** 30](#_Toc198628676)

[**Class-diagram-v1.0** 46](#_Toc198628677)

[**Sequence-diagram-v1.0** 49](#_Toc198628678)

[**Project-code-v1.0** 62](#_Toc198628679)

[**Test-cases-v1.0** 63](#_Toc198628680)

**Ρόλοι ομάδας:**

**Editor:** Νυμφοδώρα Μπουζιάνη, Σωτήριος Χασιώτης

**Contributors:** Κωσταντίνος Καρακίτσος, Βασίλειος Μιχανετζής, Μιχαήλ Σοκολάκης, Νυμφοδώρα Μπουζιάνη, Σωτήριος Χασιώτης

**Software Developer:** Σωτήριος Χασιώτης, Βασίλειος Μιχανετζής

**Εργαλεία :** Τα εργαλεία που χρησιμοποιήσαμε για την σύνταξη των τεχνικών κειμένων είναι το Microsoft Word και για την δημιουργία των διαγραμμάτων το draw.io.

**Σημαντική σημείωση: Στα use-cases έχει προστεθεί και ένα ακόμα use case το redo (το οποίο είναι παρόμοιας λογικής με το use case του undo) αλλά δεν το συμπεριλαμβάνουμε στα δέκα βασικά use cases του πρότζεκτ που θα υλοποιήσουμε.**

# **Project-description-v1.0**

**Αναλυτική περιγραφή του έργου σε φυσική γλώσσα:**

Ο σκοπός της εφαρμογής είναι η εξοικείωση και η οπτικοποίηση βασικών εννοιών προγραμματισμού. Ο χρήστης θα έχει την δυνατότητα να δηλώσει μεταβλητές και να τις χρησιμοποιεί μέσω μπλοκ εντολών συμπεριλαμβανομένων μαθηματικών πράξεων , δομών ελέγχου if καθώς και βρόχων while. Η επεξεργασία των εντολών θα γίνεται εύκολα από τον χρήστη μέσω drag and drop χρησιμοποιώντας το ποντίκι του. Η εφαρμογή θα παρέχει λειτουργία εκτέλεσης του ‘προγράμματος’ και την ενημέρωση των μεταβλητών η οποία θα γίνεται σε πραγματικό χρόνο. Η εφαρμογή εκτός από την κανονική εκτέλεση θα παρέχει και την λειτουργία debug όπου ο χρήστης θα έχει την δυνατότητα να επιλέγει το σημείο του προγράμματος μέσω breakpoints στο όποιο επιθυμεί να δει πιο συγκεκριμένα τις τιμές των μεταβλητών . Επίσης o χρήστης μπορεί να φτιάξει έως μία δική του υπορουτίνα και να την καλέσει μέσα στο πρόγραμμά του. Τέλος ο χρήστης θα έχει την ευκαιρία εξαγωγής, εισαγωγής και αποθήκευσης του project που θα έχει δημιουργήσει.

**Η υλοποίηση του έργου μας:**

Με το άνοιγμα της εφαρμογής ο χρήστης βρίσκεται στην παρακάτω οθόνη, η οποία αποτελεί και την βασική οθόνη καθ’ όλη τη διάρκεια χρήσης της εφαρμογής .

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, σχεδίαση

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Δήλωση και αρχικοποίηση μεταβλητών:**

Ο χρήστης έχει την δυνατότητα να δημιουργήσει νέες μεταβλητές ορίζοντας το όνομα τους και την τιμή τους, πατώντας το κουμπί New Var .

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, ορθογώνιο παραλληλόγραμμο, πολυχρωμία

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Όταν τελειώσει με την δήλωση των μεταβλητών, ο χρήστης πατάει το κουμπί check ώστε να πραγματοποιηθεί ο έλεγχος ορθότητας των μεταβλητών (δηλαδή το όνομα της μεταβλητής να αποτελείται από έναν χαρακτήρα και η τιμή να είναι ακέραια ). Εάν ο έλεγχος είναι επιτυχής, τότε οι μεταβλητές εκτυπώνονται όπως φαίνεται στην παρακάτω εικόνα ενώ αν ο έλεγχος είναι ανεπιτυχής απορρίπτονται.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λειτουργικό σύστημα

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Δημιουργία μπλοκ εντολών:**

Ο χρήστης επιλέγει από την αριστερή μεριά του GUI τα διαθέσιμα μπλοκ εντολών συμπληρώνοντας τα πεδία με τις τιμές που επιθυμεί.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, λογισμικό

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Στην τρέχουσα έκδοση ο συντακτικός έλεγχος γίνεται στα μπλοκ εντολών κατά την εισαγωγή τιμών δυναμικά δηλαδή τα μπλοκ θα αλλάζουν χρώμα καθώς ο χρήστης συμπληρώνει τις επιθυμητές τιμές και στα σφάλματα θα χρωματίζονται κόκκινα τα αντίστοιχα μπλοκ.

**Μετακίνηση και ένωση μπλοκ εντολών:**

Ο χρήστης τώρα έχει την δυνατότητα να μετακινεί και να ενώνει με το ποντίκι του τα μπλοκ εντολών ώστε να σχηματίζει τον κώδικά του(με πρώτο πάντα το start καθώς από αυτό ξεκινάει η εκτέλεση).

Επιπλέον μόλις δύο μπλοκ έρθουν αρκετά κοντά, θα εμφανιστεί preview της ένωσης.

Εικόνα που περιέχει ορθογώνιο παραλληλόγραμμο, στιγμιότυπο οθόνης, σχεδίαση

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Παρακάτω βλέπουμε ένα παράδειγμα κώδικα που μετράει τους άρτιους από το αριθμό 0 έως το 9.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, πολυχρωμία, διάγραμμα

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Διαγραφή μπλοκ εντολών:**

Ο χρήστης έχει την δυνατότητα να διαγράψει οποιοδήποτε μπλοκ επιθυμεί, με τα υπόλοιπα αριστερά μπλοκ εντολών που απομένουν να μετακινούνται ώστε να καλυφθεί το κενό που δημιουργήθηκε από την διαγραφή.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, ορθογώνιο παραλληλόγραμμο, σχεδίαση

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.Εικόνα που περιέχει στιγμιότυπο οθόνης, ορθογώνιο παραλληλόγραμμο, διάγραμμα, σχεδίαση

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Compile:**

Με τον κώδικα έτοιμο για compilation, ο χρήστης πατάει το κουμπί compile.Αν βρεθούν συντακτικά λάθη ή γενικά κάποιο άλλο σφάλμα σε κάποιο μπλοκ (πχ. ο κώδικας περιέχει if αλλά όχι endif) αυτό θα χρωματιστεί κόκκινο και θα απενεργοποιηθεί το κουμπί run.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, πολυχρωμία, διάγραμμα

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Εκτέλεση:**

Στην τρέχουσα έκδοση το σύστημα ελέγχει αν έχει προηγηθεί επιτυχή compilation μετά από κάποια αλλαγή και άμα δεν έχει γίνει να το ζητάει από τον χρήστη.

Ο χρήστης έχει την δυνατότητα να εκτελέσει το πρόγραμμα του με το πάτημα του κουμπιού run. Εάν επιθυμεί να σταματήσει την εκτέλεση του προγράμματος, πατάει το κουμπί stop.

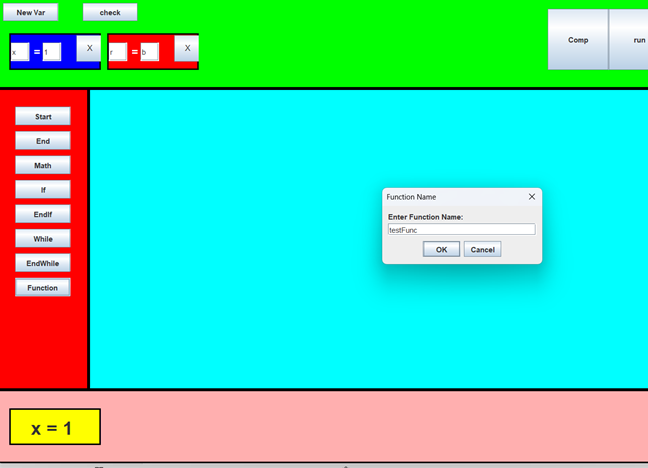
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, πολυχρωμία, διάγραμμα

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Παρουσιάζουμε παράδειγμα όπου το πρόγραμμα μετρά τους άρτιους αριθμούς από τον αριθμό 0 έως το 9 και μετά την εκτέλεση το y ισούται με 5 το οποίο είναι το σύνολο των άρτιων αριθμών και είναι σωστό.

**Υπορουτίνα:**

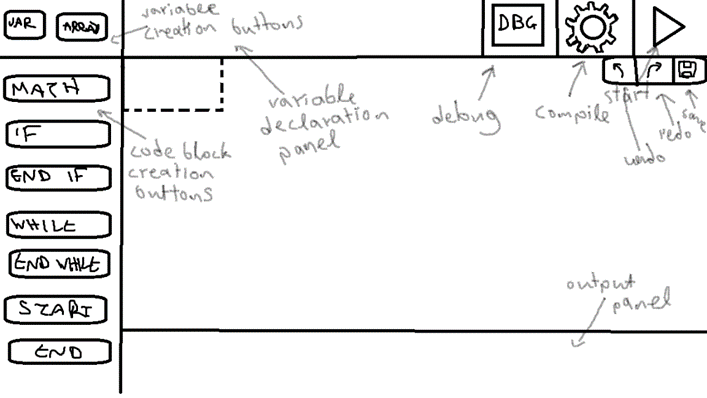
Ο χρήστης έχει την δυνατότητα να δημιουργήσει μια υπορουτίνα με το πάτημα του κουμπιού Function και εφόσον εισαγάγει το επιθυμητό όνομα δημιουργούνται και εμφανίζονται τα εξής τρία μπλοκ εντολών το CallFunction, StartFunction και EndFunction.



Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, πολυχρωμία

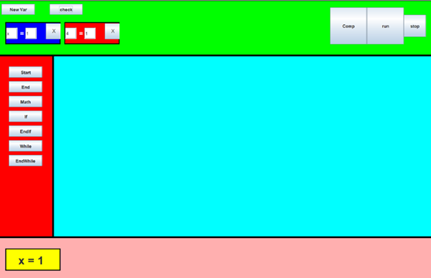
Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Το μπλοκ CallFunction πρέπει να τοποθετηθεί στην main ροή του προγράμματος ώστε τα μπλοκ εντολών που βρίσκονται ανάμεσα στο StartFunction και στο EndFunction να εκτελεστούν.

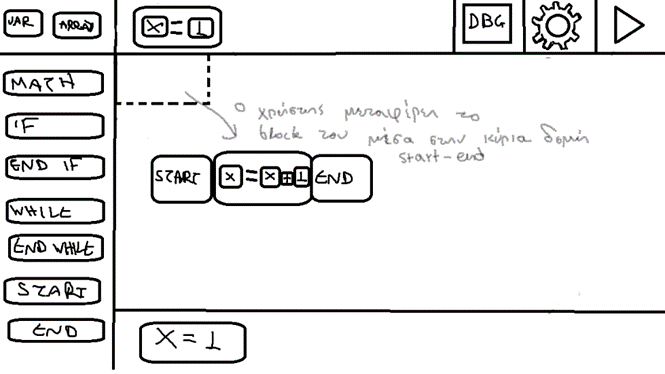
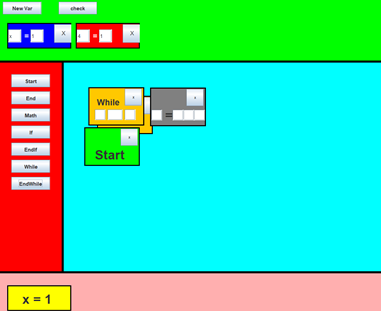
 

**Mock-up οθόνης Τελική οθόνη υλοποίησης**

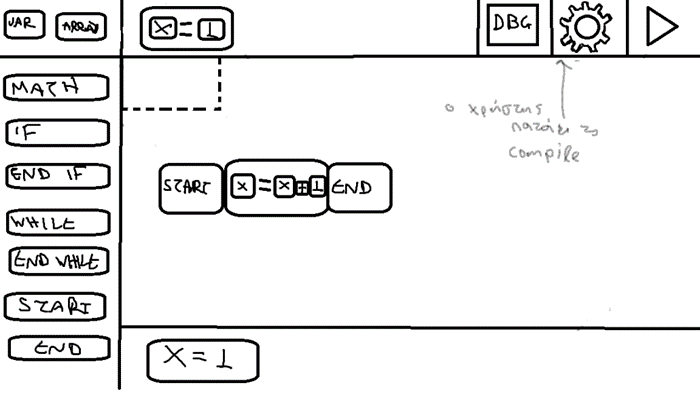
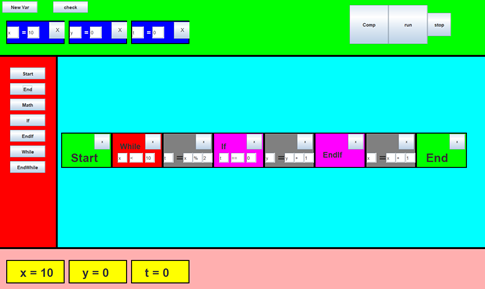
Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμματοσειρά

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο. 

**Mock-up οθόνης δήλωσης μεταβλητών Τελική οθόνη δήλωσης μεταβλητών**

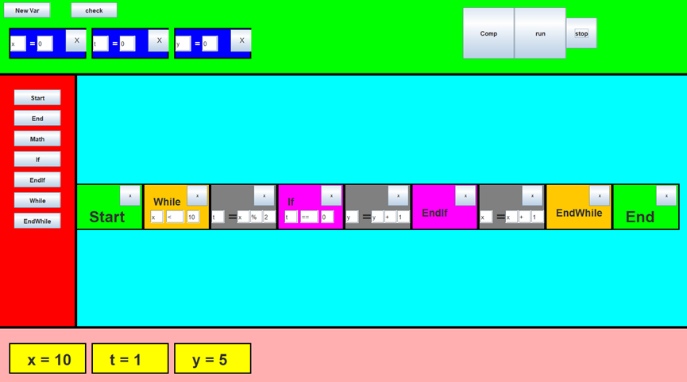
 

**Mock-up οθόνης δημιουργίας μπλοκ εντολών Τελική οθόνη δημιουργίας μπλοκ εντολών**

**Mock-up οθόνης compile Τελική οθόνη υλοποίησης compile**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμματοσειρά

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο. ****

**Mock-up οθόνης εκτέλεσης Τελική οθόνη υλοποίησης εκτέλεσης**

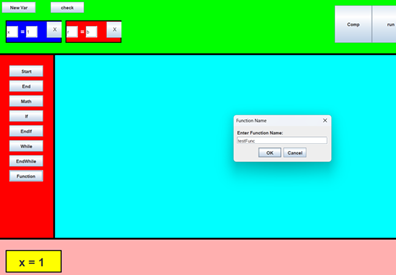
**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμματοσειρά

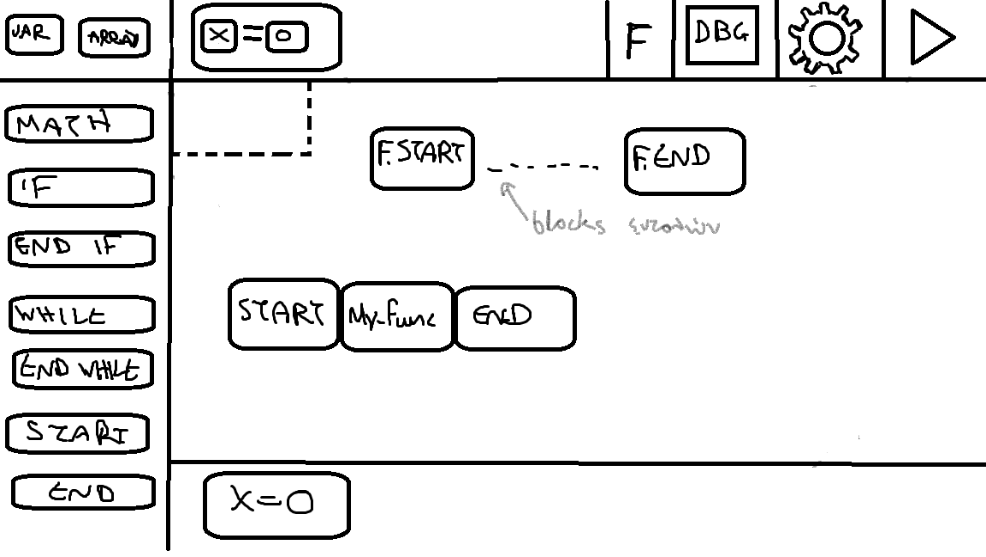
Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο. Εικόνα που περιέχει ορθογώνιο παραλληλόγραμμο, σχεδίαση

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.Mock-up οθόνης μετακίνησης και ένωσης μπλοκ Τελική οθόνη υλοποίησης μετακίνησης και**

**ένωσης μπλοκ**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο. ****

 Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, πολυχρωμία

Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

**Mock-up οθόνης υπορουτίνας Τελική οθόνη υλοποίησης υπορουτίνας**

# **Use-cases-v1.0**

**Δεν υπήρξε αλλαγή στα use cases από την έκδοση blocko v0.3.**

**Use case Diagram:**

****

**Αναλυτική περιγραφή των use cases**

**Use case**: Δήλωση και αρχικοποίηση μεταβλητών

**Βασική ροή:**

**1**. Ο χρήστης επιλέγει το πεδίο δημιουργίας νέας μεταβλητής.

**2**. Το σύστημα εμφανίζει στον χώρο των δηλώσεων των μεταβλητών ένα μπλοκ με δύο κενές θέσεις εισαγωγής.

**3**. Ο χρήστης ορίζει το όνομα της μεταβλητής και εισάγει τιμή και πατάει το κουμπί ελέγχου.

**4**. Το σύστημα ελέγχει την σύνταξη του ονόματος και διαπιστώνει ότι είναι σωστή.

**5**. Το σύστημα ελέγχει αν υπάρχει άλλη μεταβλητή με το ίδιο όνομα και διαπιστώνει ότι δεν υπάρχει.

**6**. Το σύστημα ελέγχει αν η τιμή είναι ακέραια και διαπιστώνει ότι είναι.

**7**.Το σύστημα καταχωρεί την νέα εγγραφή μεταβλητής.

**Εναλλακτική Ροή 1:**

**4.1.1.** Το σύστημα διαπιστώνει ότι η σύνταξη είναι λανθασμένη.

**4.1.2.** Το σύστημα χρωματίζει κόκκινο το μπλοκ.

**Εναλλακτική Ροή 2:**

**6.2.1.** Το σύστημα διαπιστώνει ότι η τιμή που έχει δοθεί δεν είναι ακέραια τιμή.

**6.2.2.** Το σύστημα χρωματίζει κόκκινο το μπλοκ.

**Use case**: Δημιουργία μπλοκ εντολών

**Βασική Ροή:**

**1.** O χρήστης δημιουργεί ένα καινούργιο τύπου block που επιθυμεί.

**2**. Το σύστημα δημιουργεί το μπλοκ που επέλεξε ο χρήστης.

**3**. Το σύστημα αλλάζει το χρώμα του μπλοκ σε κόκκινο.

**4**. Ο χρήστης εισάγει την τιμή και την μεταβλητή.

**5**. Το σύστημα ελέγχει την μεταβλητή και την τιμή της ως προς την εγκυρότητα.

**6**. Το σύστημα ελέγχει αν η μεταβλητή είναι δηλωμένη.

**7.** Το σύστημα επαναφέρει το χρώμα του μπλοκ , ως ένδειξη ότι δεν βρέθηκαν λάθη κατά τον έλεγχο στα βήματα 5 και 6.

**8.** Ο χρήστης παίρνει το νέο του μπλοκ και το τοποθετεί στην βασική ροή του προγράμματος ή του function του.

**Εναλλακτική Ροή 1:**

**6.1.1.** Το σύστημα διαπιστώνει ότι ο χρήστης έχει εισαγάγει μεταβλητή η οποία δεν έχει δηλωθεί.

**6.1.2.** Το σύστημα κάνει κόκκινο το μπλοκ.

**6.1.3**. Ο χρήστης καλείται να εισάγει εκ νέου νέα μεταβλητή.

**6.1.4.** Συνέχεια από το βήμα 4 της βασικής ροής.

**Use case**: Compile

**Βασική Ροή:**

**1**. Ο χρήστης κάνει compile εφόσον έχει ολοκληρώσει την δημιουργία του κώδικα του.

**2**. Το σύστημα ελέγχει αν υπάρχει end block.

**3.** Το σύστημα διαπερνά την λίστα με τα blocks και ελέγχει αν κάποιο από αυτά περιέχει συντακτικό λάθος στο όνομα του ή δεν έχει δηλωθεί.

**4**. Το σύστημα κάνει έλεγχο για δομές όπως if και βρόγχους while για να βρει τα αντίστοιχα end τους και εφόσον υπάρχουν, ανταλλάσσονται οι δείκτες τους .

**5**. Το σύστημα ελέγχει για function block και εφόσον υπάρχουν διαπερνά, με τον ίδιο τρόπο του βήματος 3, την λίστα τους επαναλαμβάνοντας το δεύτερο, το τρίτο και το τέταρτο βήμα .

**6**. Εφόσον όλα είναι έγκυρα το σύστημα εμφανίζει στον χρήστη αντίστοιχο μήνυμα επιτυχίας.

**7**. Το σύστημα εμφανίζει στον χρήστη την επιλογή να τρέξει το πρόγραμμα του.

**Εναλλακτική Ροή 1:**

**2.1.1.** Το σύστημα διαπιστώνει ότι ο χρήστης δεν έχει εισάγει end block

**2.1.2.** Το σύστημα ενημερώνει το χρήστη για απουσία end block χρωματίζοντας το start μπλοκ κόκκινο .

**Εναλλακτική Ροή 2:**

**3.2.1.** Το σύστημα διαπιστώνει ότι ο χρήστης έχει κάνει συντακτικό λάθος κατά τη δημιουργία κάποιου block.

**3.2.2.** Το σύστημα ενημερώνει τον χρήστη επισημαίνοντας πού εντοπίζονται τα συντακτικά λάθη χρωματίζοντας τα αντίστοιχα μπλοκ κόκκινα.

**Εναλλακτική Ροή 3:**

**4.3.1.** Το σύστημα διαπιστώνει ότι ο χρήστης δεν έχει προσθέσει end if και end while blocks για κάθε while και if block.

**4.3.2.** Το σύστημα ενημερώνει το ότι λείπουν τα end blocks χρωματίζοντας ανάλογα τα while ή τα if μπλοκ με κόκκινο χρώμα.

**Εναλλακτική Ροή 4:**

**5.4.1.** Το σύστημα διαπιστώνει ότι υπάρχει κάποιο σφάλμα στον κώδικα της υπορουτίνας .

**5.4.2.** Το σύστημα ενημερώνει για την ύπαρξη σφάλματος χρωματίζοντας το μπλοκ κλήσης υπορουτίνας κόκκινο.

**Use case**: Εκτέλεση

**Βασική Ροή:**

**1.** Ο χρήστης εκτελεί το πρόγραμμα.

**2**. Το σύστημα ελέγχει αν έχει γίνει compile μετά από την τελευταία αλλαγή του κώδικα.

**3.** Ο μετρητής προγράμματος αρχικοποιείται με μηδέν .

**4.** Το σύστημα ανακτά το μπλοκ που δείχνει ο μετρητής προγράμματος.

**5.** Το σύστημα εμφανίζει φωτεινό περίγραμμα γύρω από την εντολή προς εκτέλεση.

**6**. Το σύστημα ελέγχει τον τύπο της εντολής και εκτελείται η εντολή.

**7.** Το σύστημα θα εισάγει χρονική καθυστέρηση για την προσομοίωση χρόνου της εντολής χρησιμοποιώντας progress bar.

**8.** Όταν διαπεράσει το σύστημα όλη την λίστα προγράμματος θα κρατήσει τις τελευταίες τιμές στην οθόνη εξόδου .

**9.** Ο χρήστης πατάει το κουμπί ελέγχου (check).

**10.** Το σύστημα επαναφέρει την οθόνη εξόδου στην αρχική της κατάσταση.

**Εναλλακτική Ροή 1:**

**2.1.1.** Το σύστημα διαπιστώνει ότι δεν έχει γίνει compile.

**2.1.2.**Εμφανίζεται μήνυμα λάθους στον χρήστη.

**Εναλλακτική Ροή 2:**

**6.2.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου δομήελέγχου if και αποτελεί αληθής δήλωση.

**6.2.2.**Ο δείκτης προχωράει κατά ένα και συνεχίζουμε από το βήμα 4.

**Εναλλακτική Ροή 2.1 (Εναλλακτική Ροή της Εναλλακτικής Ροής 2):**

**(6.2.1).2.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου δομήελέγχου if και αποτελεί εσφαλμένη δήλωση.

**(6.2.1).2.2.** Ο δείκτης επιστέφει στο ζευγάρι του end if και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Εναλλακτική Ροή 3:**

**6.3.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου βρόγχου while και αποτελεί αληθής δήλωση.

**6.3.2.** Ο δείκτης μετακινείται και γίνεται είσοδος στον βρόγχο while για να εκτελεστούν όλες οι εντολές μέχρι το end while και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Εναλλακτική Ροή 3.1 (Εναλλακτική Ροή της Εναλλακτικής Ροής 3):**

**(6.3.1).3.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου βρόγχου while και αποτελεί εσφαλμένη δήλωση.

**(6.3.1).3.2.**Μετακίνηση στο end ,ο δείκτης αυξάνεται κατά ένα και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Εναλλακτική Ροή 4:**

**6.4.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου end while.

**6.4.2.** Μετακίνηση στην αρχή του βρόγχου και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Εναλλακτική Ροή 5:**

**6.5.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου block κλήσης υπορουτίνας.

**6.5.2.** Μεταφορά στην λίστα του function όπου εκτελεί ό,τι έχει μέσα.

**6.5.3.** Επιστροφή στην main και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Εναλλακτική Ροή 6:**

**6.6.1.** Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου math block.

**6.6.2.** Αύξηση δείκτη κατά ένα και συνεχίζουμε από το βήμα 4 της βασικής ροής.

**Use case**: Μετακίνηση και ένωση block εντολών

**Βασική Ροή:**

**1.** Ο χρήστης επιλέγει το μπλοκ που επιθυμεί να μετακινήσει μέσω drag.

**2.** Ο χρήστης μετακινεί το block πάνω από την θέση που θέλει να το τοποθετήσει.

**3.** Το σύστημα ελέγχει αν το block που μετακινεί ο χρήστης βρίσκεται σε κοντινή απόσταση από κάποιο άλλο.

**4.** Το σύστημα χρωματίζει την θέση στην οποία αναμένεται να προστεθεί το block.

**5.** Το σύστημα μετακινεί τα block που βρίσκονται δεξιά της θέσης αυτής κατά μία θέση.

**6.** Ο χρήστης αφήνει το block στη θέση που επιθυμεί.

**7.** Το σύστημα επισυνάπτει το μετακινούμενο block στο κοντινό του block, δημιουργώντας την επιθυμητή σύνδεση.

**8.** Το σύστημα ελέγχει αν το block που μετακινήθηκε μπορεί να ενταχθεί στην κύρια ροή του κώδικα ή αν ανήκει σε κάποια υπορουτίνα βάση της θέσης του και της δομής του προγράμματος.

**9.** Ανάλογα με τον έλεγχο, το σύστημα προσθέτει το μετακινούμενο μπλοκ στην κατάλληλη θέση της λίστας.

**Εναλλακτική Ροή 1:**

**2.1.1.** Ο χρήστης προσπαθεί να μετακινήσει το μπλοκ εκτός ορίων οθόνης.

**2.1.2.** Το σύστημα κρατάει το block στα όρια του επιτρεπτού πεδίου.

**Use case**: Διαγραφή block εντολών

**Βασική Ροή:**

**1.** O χρήστης διαγράφει το block που επιθυμεί.

**2.** Το σύστημα εμφανίζει παράθυρο επιβεβαίωσης της διαγραφής.

**3.** O χρήστης επιβεβαιώνει.

**4.** Το σύστημα κλείνει το παράθυρο.

**5.** Το σύστημα διαγράφει το μπλοκ από τη λίστα .

**6.** Tο σύστημα διαγράφει το block.

**7.** Το σύστημα συνδέει το επόμενο μπλοκ από αυτό που διαγράφτηκε στο προηγούμενο και επαναλαμβάνει μέχρι το τέλος των εντολών.

**Εναλλακτική Ροή 1:**

**3.1.1.** Ο χρήστης δεν επιβεβαιώνει και δεν υπάρχουν αλλαγές στο πρόγραμμα του.

**Use case**: Αποθήκευση

**Βασική Ροή:**

**1.** Ο χρήστης αποθηκεύει το πρόγραμμα.

**2.** Το σύστημα εμφανίζει ένα παράθυρο για όνομα αρχείου και τοποθεσία αποθήκευσης και δύο κουμπιά cancel και Save As.

**3.** Ο χρήστης εισάγει το όνομα του αρχείου.

**4.** Το σύστημα ελέγχει το όνομα του αρχείου ως προς την εγκυρότητα.

**5.** Εφόσον το όνομα του αρχείου είναι έγκυρο, ο χρήστης επιλέγει τοποθεσία αποθήκευσης.

**6.** Ο χρήστης κάνει Save As .

**7.** Το σύστημα αποθηκεύει το αρχείο στην επιλεγμένη τοποθεσία.

**8.** Το σύστημα εμφανίζει μήνυμα επιτυχούς αποθήκευσης.

**Εναλλακτική Ροή 1:**

**4.1.1.** Το σύστημα διαπιστώνει ότι το όνομα που εισήγαγε ο χρήστης δεν είναι έγκυρο.

**4.1.2.** Το σύστημα εμφανίζει στον χρήστη μήνυμα λάθους.

**4.1.3.** Η περίπτωση χρήσης συνεχίζεται από το βήμα 3 της βασικής ροής.

**Εναλλακτική Ροή 2:**

**2.2.1.** Ο χρήστης πατάει το κουμπί cancel.

**2.2.2.** Το σύστημα κλείνει το παράθυρο χωρίς να γίνεται η αποθήκευση του αρχείου.

**Use case**: Undo

**Βασική Ροή:**

**1.** Ο χρήστης κάνει αλλαγές στο πρόγραμμα.

**2.** Το σύστημα αποθηκεύει τις τελευταίες δύο σε μορφή στιγμιότυπου.

**3.** Ο χρήστης πατάει το κουμπί Undo.

**4.** Το σύστημα ελέγχει την ύπαρξη στιγμιότυπων.

**5.** Το σύστημα έχει περισσότερα από ένα στιγμιότυπα και αποθηκεύει το τρέχον στιγμιότυπο.

**6.** Το σύστημα κάνει εκκαθάριση οθόνης.

**7.** Το σύστημα ανακατασκευάζει το πρότζεκτ στην προηγούμενη κατάσταση.

8. Το σύστημα ενημερώνει την οθόνη καμβά.

**9.** Το σύστημα ενεργοποιεί το κουμπί Redo.

**Εναλλακτική Ροή 1:**

**5.1.1.** Το σύστημα έχει ένα μοναδικό στιγμιότυπο και αποθηκεύει το τρέχον στιγμιότυπο.

**5.1.2.** Το κουμπί undo απενεργοποιείται.

**5.1.3.** Συνέχεια από το βήμα 6.

**Use case**: Υπορουτίνα

**Βασική Ροή:**

**1.** Ο χρήστης δημιουργεί μια νέα υπορουτίνα εφόσον δεν υπάρχει άλλη υπορουτίνα.

**2.** Το σύστημα εμφανίζει ένα κενό πεδίο στο οποίο ζητά από το χρήστη να εισάγει όνομα .

**3.** Ο χρήστης πληκτρολογεί όνομα για τη νέα function και έπειτα συνεχίζει.

**4.** Το σύστημα ελέγχει την εγκυρότητα του ονόματος της υπορουτίνας.

**5.** Το σύστημα εμφανίζει τα block "Start Function" και "End Function" μεταξύ των οποίων ο χρήστης πρέπει να τοποθετήσει μπλοκ εντολών και το ίδιο το μπλοκ κλήσης του function.

**6.** Ο χρήστης τοποθετεί τα block εντολών που επιθυμεί μεταξύ των μπλοκ "Start Function" και "End Function".

**7.** Το σύστημα αποθηκεύει τους δείκτες τους σε νέα λίστα από αυτή της βασικής ροής του προγράμματος.

**8.** O χρήστης βάζει το block κλήσης της function στην main.

**Εναλλακτική Ροή 1:**

**3.1.1.** Ο χρήστης πληκτρολογεί μη έγκυρο όνομα στο κενό πεδίο και κάνει κλικ στο κουμπί "Continue".

**3.1.2.** Το σύστημα εμφανίζει μήνυμα λάθους.

**Εναλλακτική Ροή 2:**

**1.2.1.** Ο χρήστης πατάει το κουμπί new function, ενώ έχει ήδη δημιουργήσει μια υπορουτίνα.

**1.2.2.** Το σύστημα ενημερώνει τον χρήστη με μήνυμα σφάλματος.

**Εναλλακτική Ροή 3:**

**3.3.1**. Ο χρήστης αντί να συνεχίσει , ακυρώνει την διαδικασία.

**3.3.2.** Δεν υπάρχουν αλλαγές στο πρόγραμμα.

**Use case**: Debug Mode

**Βασική ροή:**

**1.** Ο χρήστης κάνει debug το πρόγραμμα και επιλέγει τα blocks που επιθυμεί να γίνει break της εκτέλεσης.

**2.** Tο σύστημα αλλάζει το χρώμα των επιλεγμένων blocks.

**3.** Εφόσον έχουν μπει breakpoints, το σύστημα ενεργοποιεί το debug mode.

**4.** Ο χρήστης κάνει εκτέλεση του προγράμματος.

**5.** Το σύστημα ελέγχει αν έχει γίνει compile από την τελευταία αλλαγή.

**6.** Το σύστημα ξεκινάει την εκτέλεση σε λειτουργία debug και εκτελεί μέχρι το επόμενο breakpoint.

**7.** Tο σύστημα κάνει ενημέρωση των νέων τιμών στο πεδίο εξόδου.

**8**. Tο σύστημα ελέγχει αν υπάρχει επόμενο breakpoint και περιμένει μέχρι ο χρήστης πατήσει run και επαναληφθούν τα βήματα 6,7,8 της βασικής ροής.

**9**. Μόλις το πρόγραμμα τερματίσει, το σύστημα απενεργοποιεί το debug mode.

**Εναλλακτική Ροή 1**

**5.1.1.** Το σύστημα διαπιστώνει ότι δεν έχει γίνει compile .

**5.1.2.** Εμφανίζεται μήνυμα λάθους στον χρήστη.

**Εναλλακτική Ροή 2**

**5.2.1.** Ο χρήστης κάνει debug το πρόγραμμα του χωρίς να βάλει breakpoints.

**5.2.2.** Το σύστημα εκτελεί όλο το πρόγραμμα.

**Use case:** Redo

**Βασική Ροή:**

**1.** Ο χρήστης έχει πατήσει το κουμπί undo τουλάχιστον μια φορά .

**2.** Το σύστημα ενεργοποιεί το κουμπί Redo.

**3.** Ο χρήστης πατάει το κουμπί Redo.

**4.** Το σύστημα ελέγχει για την ύπαρξη αποθηκευμένων στιγμιότυπων και διαπιστώνει ότι έχει περισσότερα από ένα στιγμιότυπο .

**5.** Το σύστημα αποθηκεύει την τρέχουσα κατάσταση στο ιστορικό-undo .

**6.** Το σύστημα ανακτά το στιγμιότυπο που αναιρέθηκε τελευταίο.

**7.** Το σύστημα κάνει εκκαθάριση οθόνης.

**8.** Το σύστημα ανακατασκευάζει το πρότζεκτ στην κατάσταση που έχει αναιρεθεί τελευταία.

**9.** Το σύστημα αφαιρεί το επαναφερόμενο στιγμιότυπο από το ιστορικό-redo.

**10.** Το σύστημα ενεργοποιεί το κουμπί undo.

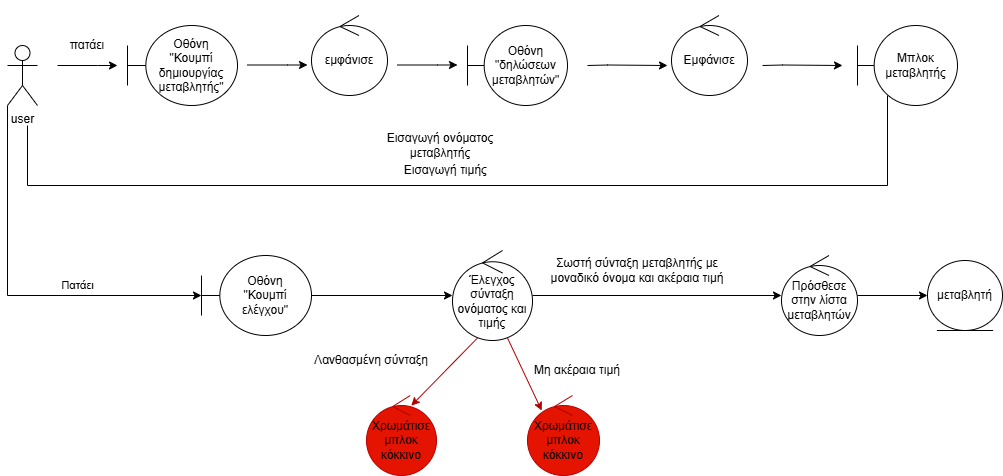
**Εναλλακτική Ροή 1:**

**4.1.1.** Το σύστημα ελέγχει την ύπαρξη στιγμιότυπων και διαπιστώνει ότι υπάρχει μόνο ένα στιγμιότυπο.

**4.1.2.**Το κουμπί redo απενεργοποιείται και το σύστημα συνεχίζει από το βήμα 5 της βασικής ροής.

# **Robustness-diagram-v1.0**

**Δεν υπήρξε αλλαγή στα robustness διαγράμματα από την έκδοση blocko v0.3.**



**Σχ1. Διάγραμμα ευρωστίας για Use Case Δήλωση και αρχικοποίηση μεταβλητών**

**Βασική Ροή: Ο χρήστης δημιουργεί μια νέα μεταβλητή .**

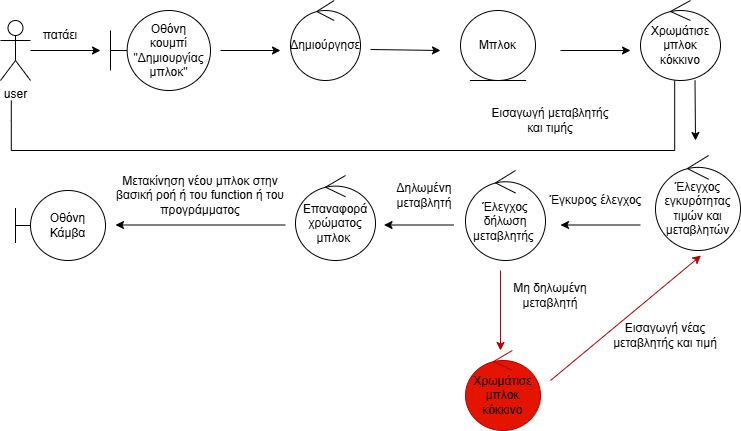
Ο χρήστης επιθυμεί να δημιουργήσει μια νέα μεταβλητή και πατάει το κουμπί δημιουργίας μεταβλητής. Εμφανίζεται το μπλοκ της μεταβλητής στον χώρο των δηλώσεων μεταβλητών και ο χρήστης καλείται να εισάγει το όνομα της μεταβλητής και την τιμή της . Αφού ο χρήστης πατήσει το κουμπί ελέγχου και γίνει ο έλεγχος για την εγκυρότητα της σύνταξης του ονόματος και της τιμής της μεταβλητής τότε το σύστημα προσθέτει τη μεταβλητή στη λίστα μεταβλητών και η διαδικασία της δημιουργίας μεταβλητής ολοκληρώνεται επιτυχώς.

**Εναλλακτική Ροή 1: Ο χρήστης εισάγει όνομα μεταβλητής με λάθος σύνταξη.**

Ο χρήστης εισάγει μη έγκυρο όνομα μεταβλητής και το σύστημα χρωματίζει το μπλοκ κόκκινο.

**Εναλλακτική Ροή 2: Ο χρήστης εισάγει μη ακέραια τιμή στην μεταβλητή.**

Ο χρήστης εισάγει μην ακέραια τιμή στην μεταβλητή και το σύστημα χρωματίζει το μπλοκ κόκκινο.



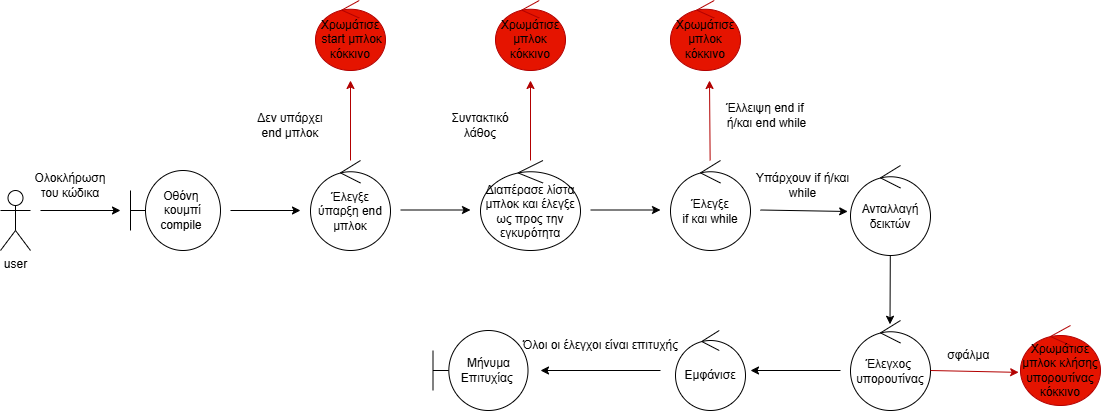
**Σχ2. Διάγραμμα ευρωστίας για Use Case Δημιουργία μπλοκ εντολών**

**Βασική Ροή: Ο χρήστης θέλει να δημιουργήσει ένα νέο μπλοκ.**

Ο χρήστης πατάει το κουμπί Δημιουργίας μπλοκ και δημιουργείται καινούργιο μπλοκ το οποίο χρωματίζεται κόκκινο. Ο χρήστης καλείται να εισάγει την επιθυμητή μεταβλητή και τιμή και εφόσον αυτά που εισήγαγε είναι έγκυρα και η μεταβλητή είναι δηλωμένη τότε το σύστημα επαναφέρει το χρώμα του μπλοκ ως ένδειξη εγκυρότητας. Τέλος ο χρήστης πρέπει να μετακινήσει το μπλοκ που δημιούργησε στην βασική ροή είτε του προγράμματος του είτε της υπορουτίνας του.

**Εναλλακτική Ροή 1: Ο χρήστης δεν έχει δηλώσει την μεταβλητή.**

Ο έλεγχος για την δηλωμένη μεταβλητή είναι ανεπιτυχής και το μπλοκ εξακολουθεί να είναι κόκκινο οπότε ο χρήστης καλείται να εισάγει εκ νέου άλλη μεταβλητή.



**Σχ3. Διάγραμμα ευρωστίας για Use Case Compile**

**Βασική Ροή: Ο χρήστης θέλει να κάνει compile τον κώδικα του.**

Ο χρήστης πατάει το κουμπί compile και το σύστημα αφού ελέγξει για την ύπαρξη start και end μπλοκ διαπερνά την λίστα μπλοκ ελέγχοντας την εγκυρότητα της σύνταξης των μπλοκ και συνεχίζει με τον έλεγχο ορθότητας των δομών if και των βρόγχων while. Στην περίπτωση που υπάρχουν if ή/και while τότε γίνεται ανταλλαγή δεικτών. Έπειτα ελέγχει για την ύπαρξη function μπλοκ και εφόσον οι έλεγχοι είναι επιτυχής εμφανίζεται μήνυμα επιτυχίας.

**Εναλλακτική Ροή 1: Ο χρήστης δεν έχει τοποθετήσει στο πρόγραμμα του end μπλοκ.**

Το σύστημα αφού διαπιστώνει ότι δεν υπάρχει end μπλοκ, ενημερώνει τον χρήστη για την απουσία του χρωματίζοντας το start μπλοκ κόκκινο.

**Εναλλακτική Ροή 2: Ο χρήστης έχει κάνει συντακτικό λάθος.**

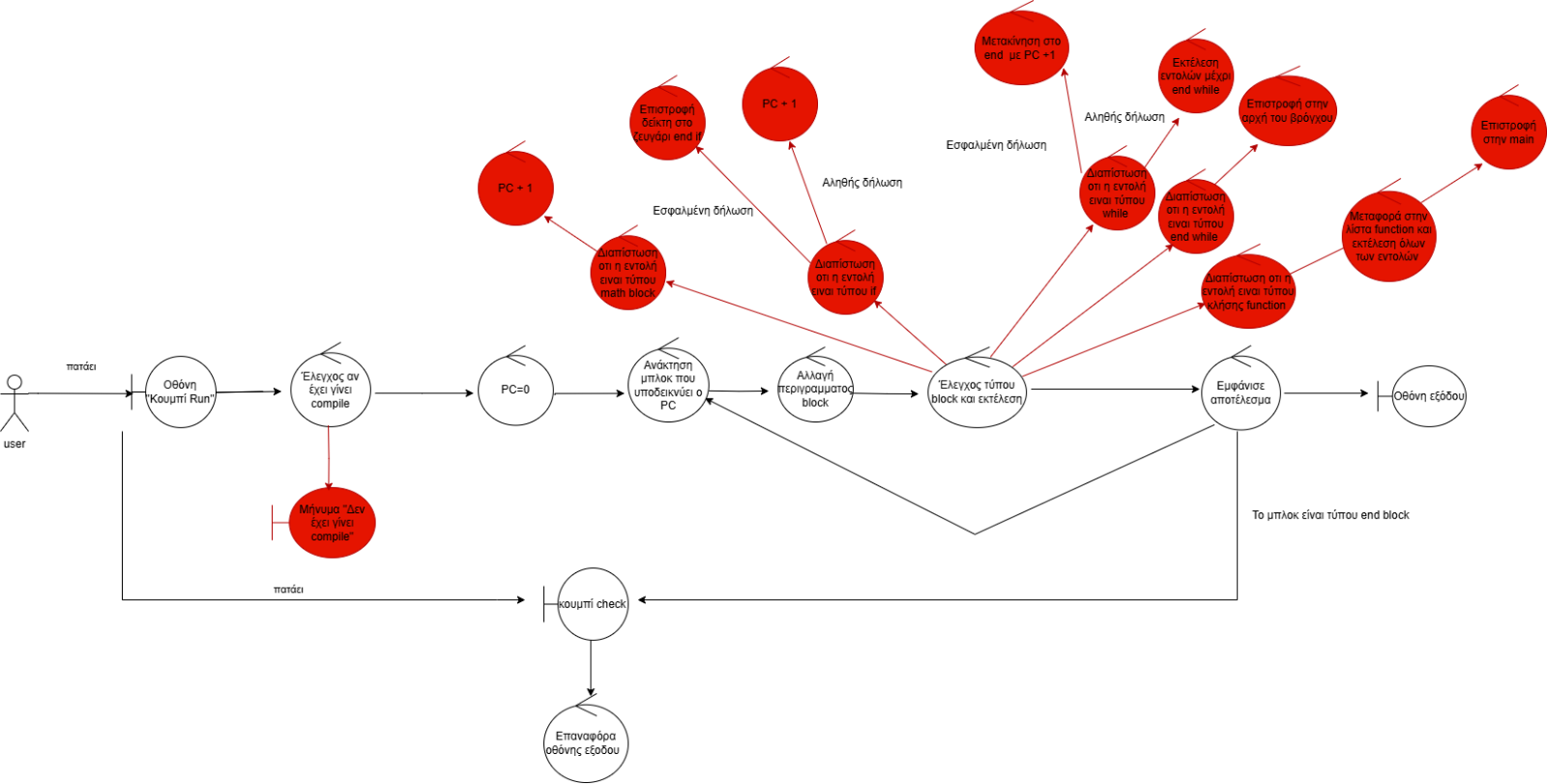
Το σύστημα διαπερνά την λίστα των μπλοκ και βρίσκει συντακτικό λάθος με αποτέλεσμα να ενημερώνει τον χρήστη για την ύπαρξη των συντακτικών σφαλμάτων του χρωματίζοντας κόκκινο το αντίστοιχο μπλοκ .

**Εναλλακτική Ροή 3: Ο χρήστης δεν έχει τοποθετήσει στο πρόγραμμα του end if ή/και end while μπλοκ.**

Το σύστημα εφόσον διαπιστώνει ότι δεν υπάρχει end if ή/και end while μπλοκ, ενημερώνει τον χρήστη για την απουσία τους χρωματίζοντας με κόκκινο χρώμα το μπλοκ αντίστοιχα του if μπλοκ ή/και του while .

**Εναλλακτική Ροή 4: Ο χρήστης δεν έχει τοποθετήσει στο πρόγραμμα του μπλοκ υπορουτίνας ή υπάρχει λάθος.**

Το σύστημα ελέγχει για την ύπαρξη function μπλοκ και διαπιστώνει ότι δεν υπάρχει ή υπάρχει λάθος στα function μπλοκ και ενημερώνει τον χρήστη για την ύπαρξη σφάλματος με χρωματισμό των μπλοκ σε κόκκινο.

****

**Σχ4. Διάγραμμα ευρωστίας για Use Case Εκτέλεση**

**Βασική Ροή: Ο χρήστης εκτελεί το πρόγραμμα του.**

Ο χρήστης πατάει το κουμπί run και γίνεται έλεγχος για το αν έχει γίνει πρώτα compile. Στην συνέχεια γίνεται από το σύστημα αρχικοποίηση του PC και ανάκτηση του μπλοκ που υποδεικνύει ο PC. Το μπλοκ αυτό αποκτά φωτεινό περίγραμμα και γίνεται ο έλεγχος ώστε το σύστημα να ταυτοποιήσει τον τύπο της εντολής. Έπειτα γίνεται η εκτέλεση της εντολής και με μια μικρή καθυστέρηση εμφανίζονται οι τιμές των μεταβλητών στην οθόνη εξόδου. Τέλος ο χρήστης πατάει το κουμπί ελέγχου (check) και γίνεται επαναφορά της οθόνης εξόδου.

**Εναλλακτική Ροή 1: Ο χρήστης δεν έχει κάνει compile και εκτελεί το πρόγραμμα του.**

Το σύστημα διαπιστώνει ότι δεν έχει γίνει compile και εμφανίζει στον χρήστη μήνυμα σφάλματος.

**Εναλλακτική Ροή 2: Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του δομή ελέγχου if και η συνθήκη if αποτελεί αληθής δήλωση.**

Το σύστημα διαπιστώνει ότι υπάρχει εντολή τύπου if με αληθής συνθήκη και αυξάνεται κατά ένα ο PC .

**Σημείωση: Ο PC αποτελεί τον δείκτη του προγράμματος.**

**Εναλλακτική Ροή 2.1 (Εναλλακτική ροή της εναλλακτικής ροής 2): Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του δομή ελέγχου if αλλά η συνθήκη if είναι εσφαλμένη.**

Το σύστημα διαπιστώνει ότι υπάρχει εντολή τύπου if με εσφαλμένη δήλωση και επιστρέφει τον δείκτη στο ζευγάρι end if.

**Εναλλακτική Ροή 3: Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του** **βρόγχο while και αποτελεί σωστή δήλωση.**

Το σύστημα διαπιστώνει ότι υπάρχει εντολή τύπου while και εκτελούνται όλες οι εντολές μέχρι και end while.

**Εναλλακτική Ροή 3.1 (Εναλλακτική ροή της εναλλακτικής ροής 3):** **Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του** **βρόγχο while και αποτελεί εσφαλμένη δήλωση.**

Το σύστημα διαπιστώνει ότι υπάρχει εντολή τύπου while, εκτελούνται όλες οι εντολές μέχρι και end while και αφού το σύστημα διαπιστώσει ότι η συνθήκη while έχει δηλωθεί λανθασμένα υπάρχει αύξηση του PC κατά ένα και μετακίνηση στο end.

**Εναλλακτική Ροή 4: Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του** **βρόγχο while και το πρόγραμμα έχει φτάσει σε τύπο μπλοκ end while.**

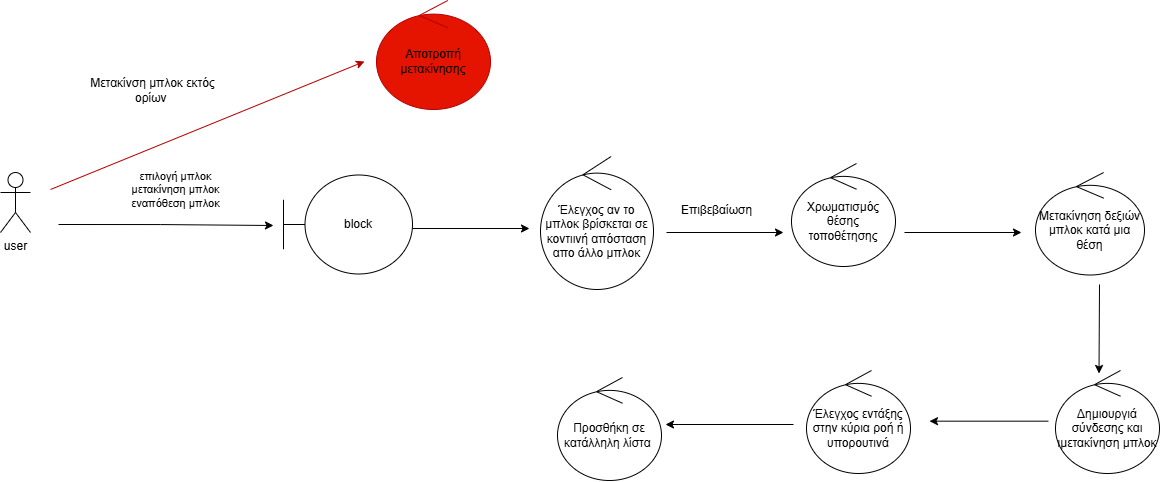
Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου end while και επιστρέφει τον δείκτη στην αρχή του βρόγχου.

**Εναλλακτική Ροή 5: Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του υπορουτίνα.**

Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου κλήσης υπορουτίνας και εκτελεί όλες τις εντολές που υπάρχουν στην λίστα της υπορουτίνας και έπειτα επιστρέφει στην main.

**Εναλλακτική Ροή 6: Ο χρήστης έχει χρησιμοποιήσει στο πρόγραμμα του** **μπλοκ** **τύπου math .**

Το σύστημα διαπιστώνει ότι η εντολή είναι τύπου math και αυξάνεται ο PC κατά ένα.



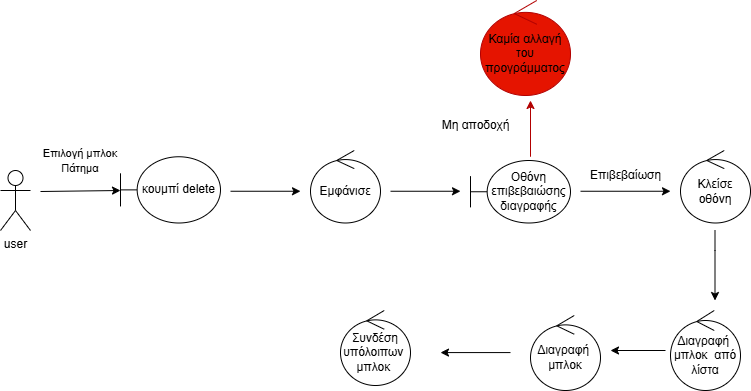
**Σχ5. Διάγραμμα ευρωστίας για Use Case Μετακίνηση και ένωση block εντολών**

**Βασική Ροή: Ο χρήστης μετακινεί και ενώνει μπλοκ εντολών.**

Ο χρήστης επιλέγει το μπλοκ που θέλει να μετακινήσει. Κατά την μετακίνηση το σύστημα ελέγχει αν βρίσκεται κοντά σε άλλο μπλοκ και χρωματίζει τη κατάλληλη θέση που αναμένεται να τοποθετηθεί το μπλοκ. Αν υπάρχει μπλοκ σε αυτή την θέση τότε μετακινούνται τα δεξιά μπλοκ κατά μία θέση . Ο χρήστης τοποθετεί στη επιθυμητή θέση το μπλοκ και το σύστημα το επισυνάπτει στο πιο κοντινό μπλοκ. Έπειτα γίνεται έλεγχος για το αν μπορεί να αποτελεί μέλος της βασικής ροής ή της υπορουτίνας και αναλόγως το σύστημα ενημερώνει τη λίστα κατάλληλα.

**Εναλλακτική Ροή 1: Ο χρήστης μετακινεί το μπλοκ εκτός ορίου καμβά.**

Ο χρήστης προσπαθεί να μετακινήσει το μπλοκ εκτός ορίου καμβά αλλά το σύστημα κρατάει το μπλοκ εντός του επιτρέπων ορίων.



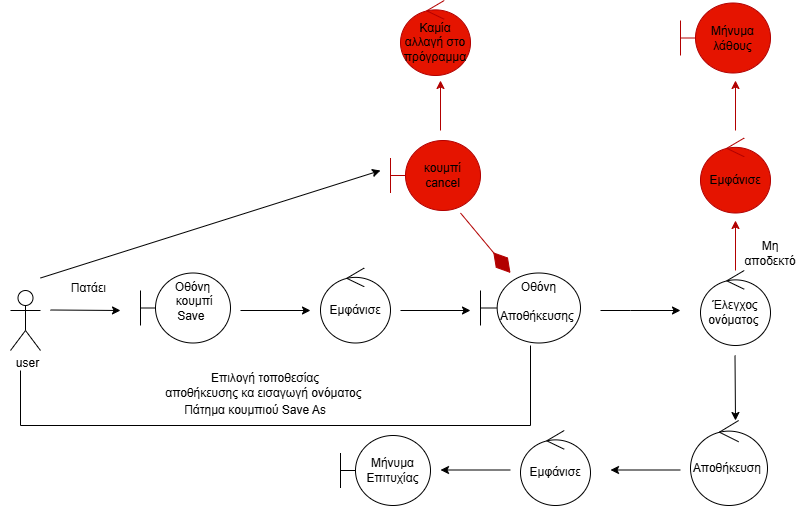
**Σχ6. Διάγραμμα ευρωστίας για Use Case Διαγραφή block εντολών**

**Βασική Ροή: Ο χρήστης θέλει να διαγράψει ένα μπλοκ.**

Ο χρήστης επιλέγει το μπλοκ που επιθυμεί να διαγράψει και πατάει το κουμπί Διαγραφής του μπλοκ με αποτέλεσμα να εμφανίζεται παράθυρο επιβεβαίωσης διαγραφής. Ο χρήστης επιβεβαιώνει και κλείνει το παράθυρο. Έπειτα το σύστημα διαγράφει το μπλοκ από την λίστα και μετά την διαγραφή του μπλοκ συνδέει μεταξύ τους τα εναπομείναντα μπλοκ.

**Εναλλακτική Ροή 1: Ο χρήστης πατάει ακύρωση όταν εμφανίζεται το παράθυρο επιβεβαίωσης.**

Ο χρήστης απορρίπτει το παράθυρο επιβεβαίωσης διαγραφής μπλοκ και δεν υπάρχουν αλλαγές στο πρόγραμμα.



**Σχ7. Διάγραμμα ευρωστίας για Use Case Αποθήκευση**

**Βασική Ροή: Ο χρήστης αποθηκεύει το πρόγραμμα του.**

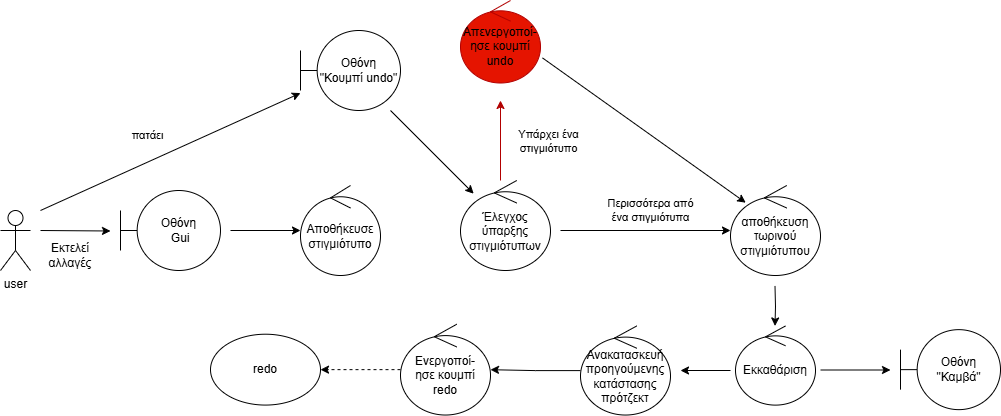
Ο χρήστης αποθηκεύει το πρόγραμμα πατώντας το κουμπί Αποθήκευσης και εμφανίζεται παράθυρο τοποθεσίας για την αποθήκευση του αρχείου του. Ο χρήστης καλείται να εισάγει όνομα στο αρχείο και εφόσον το σύστημα ελέγχει την εγκυρότητα του ονόματος τότε ο χρήστης επιλέγει την τοποθεσία αποθήκευσης που επιθυμεί και με το πάτημα του κουμπιού Save As αποθηκεύεται το αρχείο. Μετά την αποθήκευση ο χρήστης ενημερώνεται με μήνυμα επιτυχίας.

**Εναλλακτική Ροή 1: Ο χρήστης εισάγει μη έγκυρο όνομα στο αρχείο που θέλει να αποθηκεύσει .**

Ο χρήστης βάζει μη έγκυρο όνομα με αποτέλεσμα να εμφανίζεται μήνυμα σφάλματος και ο χρήστης να καλείται να βάλει καινούργιο όνομα αρχείου.

**Εναλλακτική Ροή 2: Ο χρήστης πατάει το κουμπί ακύρωσης στο παράθυρο Αποθήκευσης.**

Ο χρήστης δεν επιθυμεί πια την αποθήκευση του αρχείου και δεν γίνεται καμία αλλαγή στο πρόγραμμα.



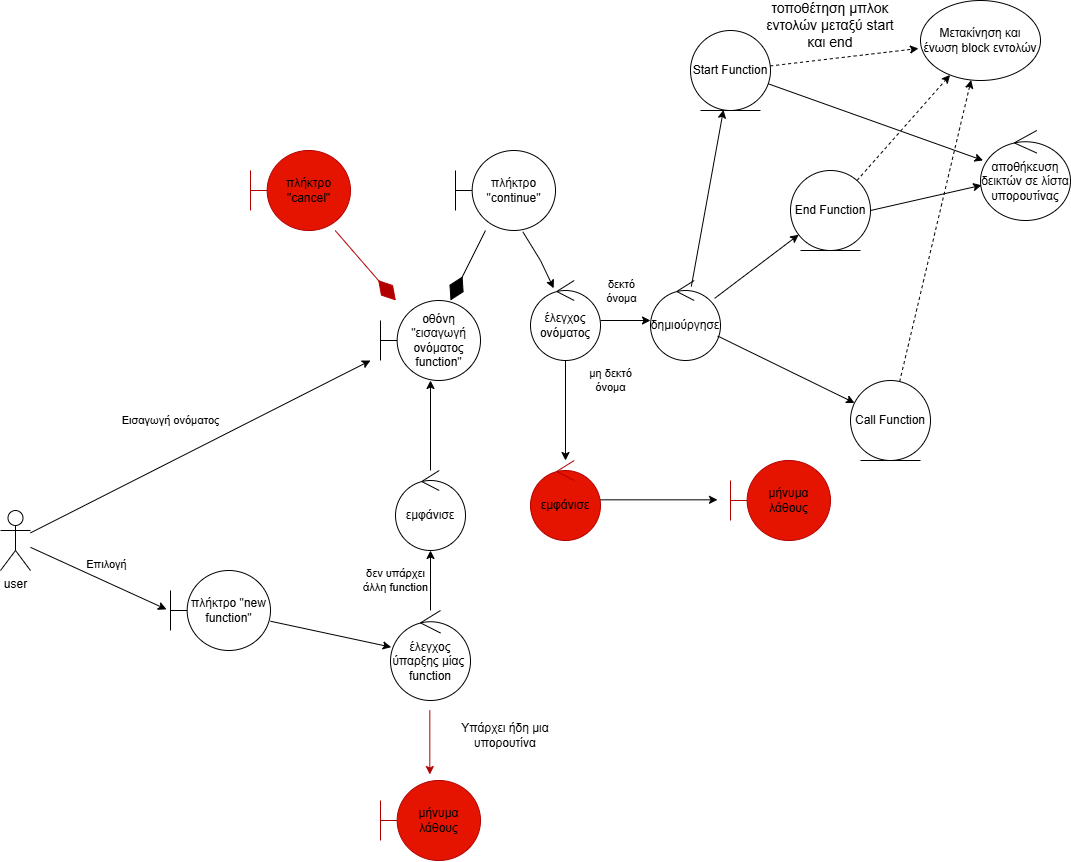
**Σχ8. Διάγραμμα ευρωστίας για Use Case Undo**

**Βασική Ροή: Ο χρήστης θέλει να επιστρέψει στην προηγούμενη κατάσταση του πρότζεκτ του.**

Ο χρήστης εκτελεί αλλαγές στο πρότζεκτ του, πατάει το κουμπί undo και γίνεται έλεγχος ύπαρξης στιγμιότυπων. Τα στιγμιότυπα είναι περισσότερα από ένα και γίνεται η αποθήκευση του τωρινού στιγμιότυπου από το σύστημα καθώς στην συνέχεια γίνεται εκκαθάριση της οθόνης. Το σύστημα ανακατασκευάζει την προηγούμενη κατάσταση του πρότζεκτ και ενεργοποιεί το κουμπί redo.

**Εναλλακτική Ροή 1: Υπάρχει ένα μόνο στιγμιότυπο.**

Ο χρήστης πατάει το undo κουμπί ενώ υπάρχει ένα μόνο στιγμιότυπο αποθηκευμένο και το σύστημα απενεργοποιεί το κουμπί undo και συνεχίζει στην αποθήκευση τωρινού στιγμιότυπου του πρότζεκτ.



**Σχ9. Διάγραμμα ευρωστίας για Use Case Υπορουτίνα**

**Βασική Ροή: Ο χρήστης θέλει να δημιουργήσει μια υπορουτίνα.**

Ο χρήστης δημιουργεί μια νέα υπορουτίνα μέσω του κουμπιού new function και εφόσον δεν υπάρχει άλλη υπορουτίνα εμφανίζεται κενό πεδίο όπου ο χρήστης καλείται να εισάγει όνομα. Ο χρήστης εισάγει όνομα και εμφανίζονται τα μπλοκ start function και end function όπου μεταξύ αυτών πρέπει ο χρήστης να τοποθετήσει μπλοκ εντολών που επιθυμεί και το μπλοκ που να καλεί την υπορουτίνα. Τα μπλοκ της υπορουτίνας αποθηκεύονται σε ξεχωριστή λίστα από αυτή του βασικού προγράμματος.

**Εναλλακτική Ροή 1: Ο χρήστης προσπαθεί να δημιουργήσει υπορουτίνα ενώ ήδη έχει δημιουργήσει μία.**

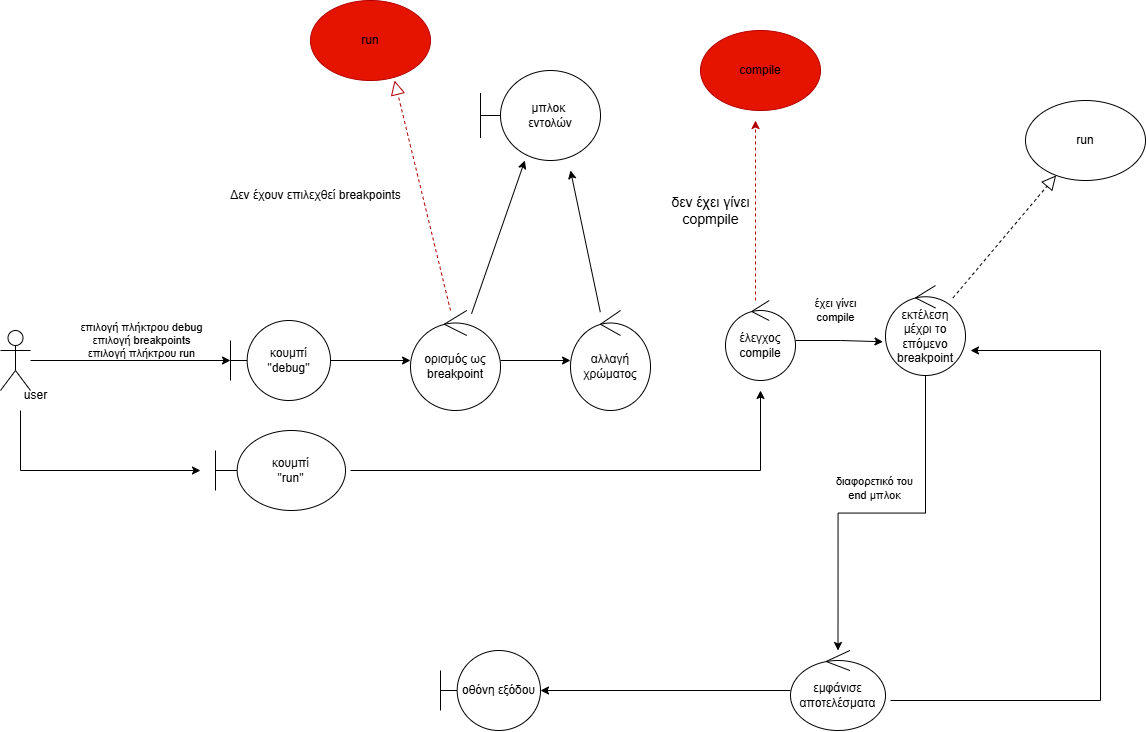
Όταν γίνεται ο έλεγχος ύπαρξης μίας υπορουτίνας, το σύστημα αντιλαμβάνεται ότι υπάρχει ήδη μια υπορουτίνα και εμφανίζει μήνυμα σφάλματος στον χρήστη.

**Εναλλακτική Ροή 2: Ο χρήστης εισάγει μη αποδεκτό όνομα στην υπορουτίνα.**

Ο χρήστης εισάγει μη αποδεκτό όνομα στην υπορουτίνα και εμφανίζεται μήνυμα σφάλματος.

**Εναλλακτική Ροή 3: Ο χρήστης ακυρώνει την διαδικασία δημιουργίας νέας υπορουτίνας.**

Ο χρήστης ακυρώνει την διαδικασία δημιουργίας νέας υπορουτίνας και το πρόγραμμα παραμένει ίδιο.

****

**Σχ10. Διάγραμμα ευρωστίας για Use Case Debug**

**Βασική Ροή: Ο χρήστης κάνει debug το πρόγραμμα του.**

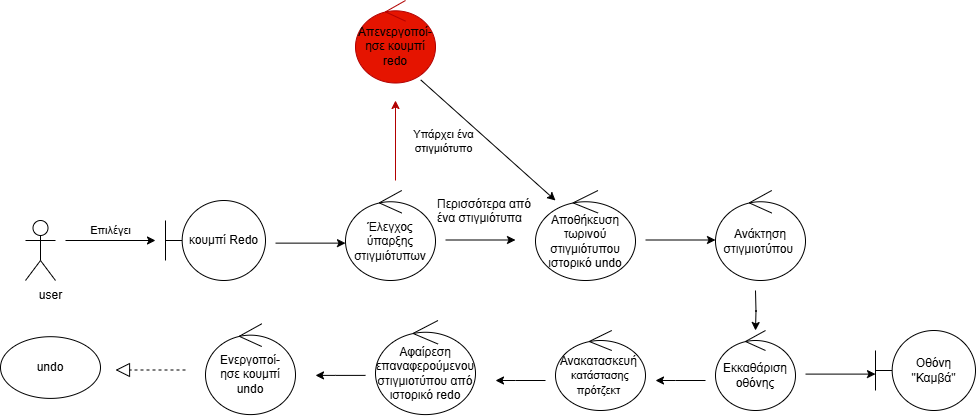
Ο χρήστης επιλέγει το κουμπί του debug και τα μπλοκ εντολών που θα χρησιμοποιήσει ως breakpoints, τα οποία και χρωματίζονται. Το σύστημα τότε ενεργοποιεί το debug mode και ο χρήστης εκτελεί το πρόγραμμα. Το σύστημα ελέγχει αν έχει γίνει compile πριν την τελευταία αλλαγή και το πρόγραμμα εκτελείται μέχρι το πρώτο breakpoint ενημερώνοντας την οθόνη εξόδου με τις νέες τιμές. Έπειτα το σύστημα ελέγχει για την ύπαρξη επόμενου breakpoint και εφόσον υπάρχει, ο χρήστης πατάει ξανά run με αποτέλεσμα να επαναλαμβάνονται τα ίδια βήματα όπως και στο πρώτο breakpoint μέχρι να τερματίσει το πρόγραμμα και να απενεργοποιηθεί το debug mode.

**Εναλλακτική Ροή 1: Ο χρήστης δεν έχει κάνει compile πριν την τελευταία αλλαγή του προγράμματος.**

Το σύστημα διαπιστώνει ότι δεν έχει γίνει compile και ενημερώνει τον χρήστη με μήνυμα σφάλματος.

**Εναλλακτική Ροή 2: Ο χρήστης δεν επιλέγει breakpoints για το πρόγραμμα του.**

Το σύστημα εκτελεί όλο το πρόγραμμα.

****

**Σχ11. Διάγραμμα ευρωστίας για Use Case Redo**

**Βασική Ροή: Ο χρήστης θέλει να ανακτήσει την κατάσταση του πρότζεκτ που αναιρέθηκε τελευταία.**

Ο χρήστης πατάει το κουμπί redo και το σύστημα κάνει έλεγχο για την ύπαρξη στιγμιότυπων. Τα στιγμιότυπα είναι περισσότερα από ένα και γίνεται η αποθήκευση του τωρινού στιγμιότυπου στο ιστορικό undo καθώς και γίνεται η ανάκτηση του στιγμιότυπου που επιθυμεί ο χρήστης. Αφού γίνει η εκκαθάριση οθόνης και η ανακατασκευή του πρότζεκτ, αφαιρείται το στιγμιότυπο από ιστορικό redo και ενεργοποιείται το κουμπί redo.

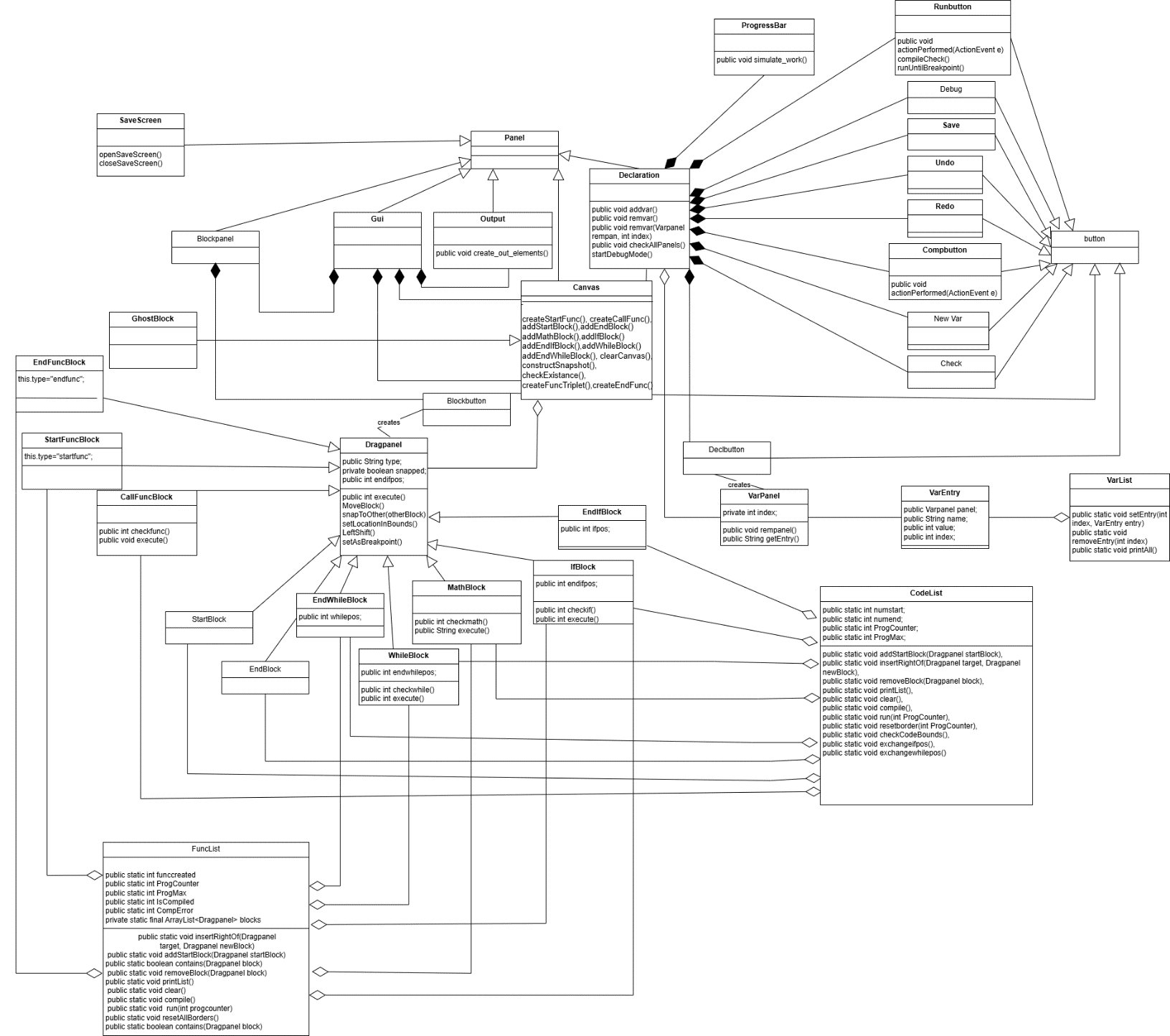
**Εναλλακτική Ροή 1: Υπάρχει ένα μόνο στιγμιότυπο.**

Ο χρήστης πατάει το redo κουμπί ενώ υπάρχει ένα μόνο στιγμιότυπο αποθηκευμένο, το σύστημα απενεργοποιεί το κουμπί redo και το σύστημα συνεχίζει με την αποθήκευση του τωρινού στιγμιότυπου του πρότζεκτ.

# **Class-diagram-v1.0**

**Καινούρια έκδοση από την προηγούμενη έκδοση (blocko-v0.3).**

**Domain Model:**

****

**Κλάσεις:**

**Panel:** Το πλαίσιο που περιέχονται τα User Interface στοιχεία.

**Block panel:** Το πλαίσιο που περιέχονται τα μπλοκ που θα χρησιμοποιήσει ο χρήστης.

**Gui:** Η διεπαφή όπου δίνει την δυνατότητα στον χρήστη να αλληλοεπιδρά με το blocko.

**Canvas**: Το πλαίσιο που ο χρήστης τοποθετεί και μετακινεί τα μπλοκ.

**Declaration**: Το πλαίσιο που διαχειρίζεται τις δηλώσεις.

**Output:** Το πλαίσιο που εμφανίζεται το αποτέλεσμα της εκτέλεσης του προγράμματος.

**Blockbutton:** Κουμπί δημιουργίας μπλοκ.

**MathΒlock:** Block που εκτελεί μαθηματικές πράξεις.

**StartFuncBlock:** Block για την αρχή μιας συνάρτησης.

**EndFuncBlock:** Block για το τέλος μιας συνάρτησης.

**CallFuncBlock:** Block για να κλήση συνάρτησης.  
**StartBlock:** Block έναρξης προγράμματος.

**EndBlock:** Block τέλος προγράμματος.

**IfBlock:** Block για δομή ελέγχου if.

**EndifBlock:** Block για τέλος δομής ελέγχου if.

**WhileBlock:** Block για βρόγχο while.

**EndwhileBlock**: Block για τέλος βρόγχου while.

**Button:** Κουμπιά του συστήματος.

**Runbutton:** Εκτελεί το πρόγραμμα.

**Save:** Αποθηκεύει το πρόγραμμα.

**Undo:** Αναίρεση της τελευταίας ενέργειας.

**Redo:** Επανεμφάνιση της αναιρεθείσας ενέργεια.

**Compbutton:** Μεταγλώττιση του προγράμματος.

**CodeList:** Λίστα που περιέχει τα μπλοκ εντολών του προγράμματος.

**FuncList:** Λίστα που περιέχει τα μπλοκ εντολών της υπορουτίνας.

**VarList:** Λίστα που περιέχει τις δηλωμένες μεταβλητές.

**Varpanel:** Το πλαίσιο διαχείρισης μεταβλητών.

**Declbutton:** Κουμπί δημιουργίας δηλώσεων μεταβλητών.

**VarEntry:** Κλάση αναπαράστασης της μεταβλητής.(αποθήκευση ονόματος και τιμής των μεταβλητών)

**Dragpanel:** Πλαίσιο για όλα τα μπλοκ .

**Check:** Κουμπί ελέγχου δήλωσης μεταβλητών.

**Progress Bar:** Γραμμή προόδου για την προσομοίωση .

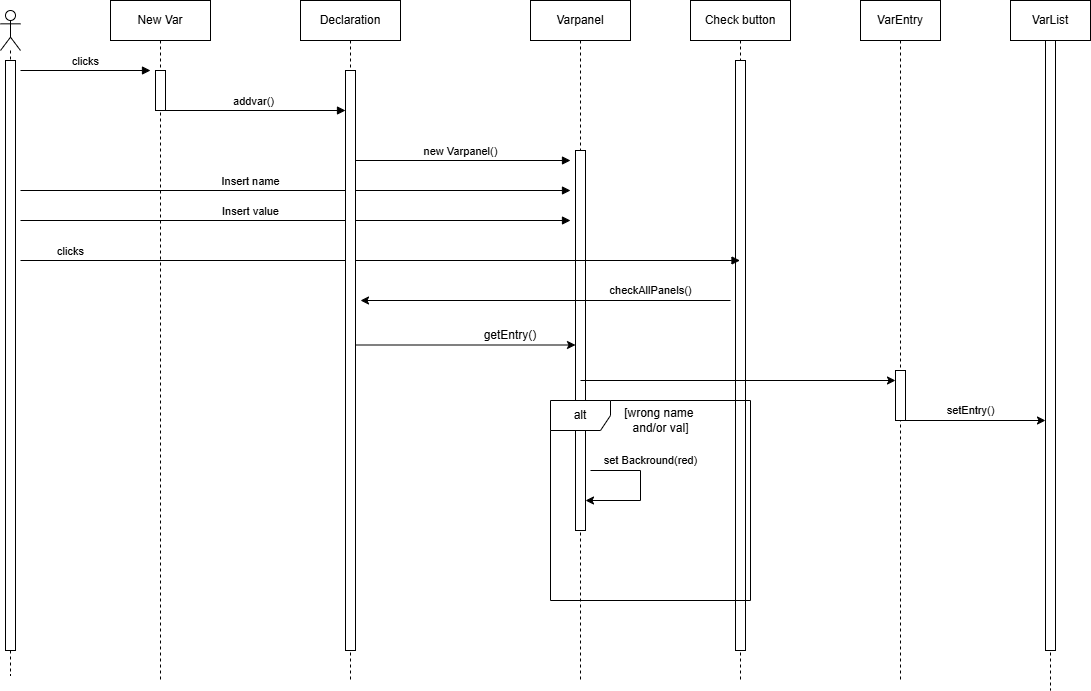
**SaveScreen:** Οθόνη αποθήκευσης.

**New Var:** Κουμπί δημιουργίας νέας μεταβλητής.

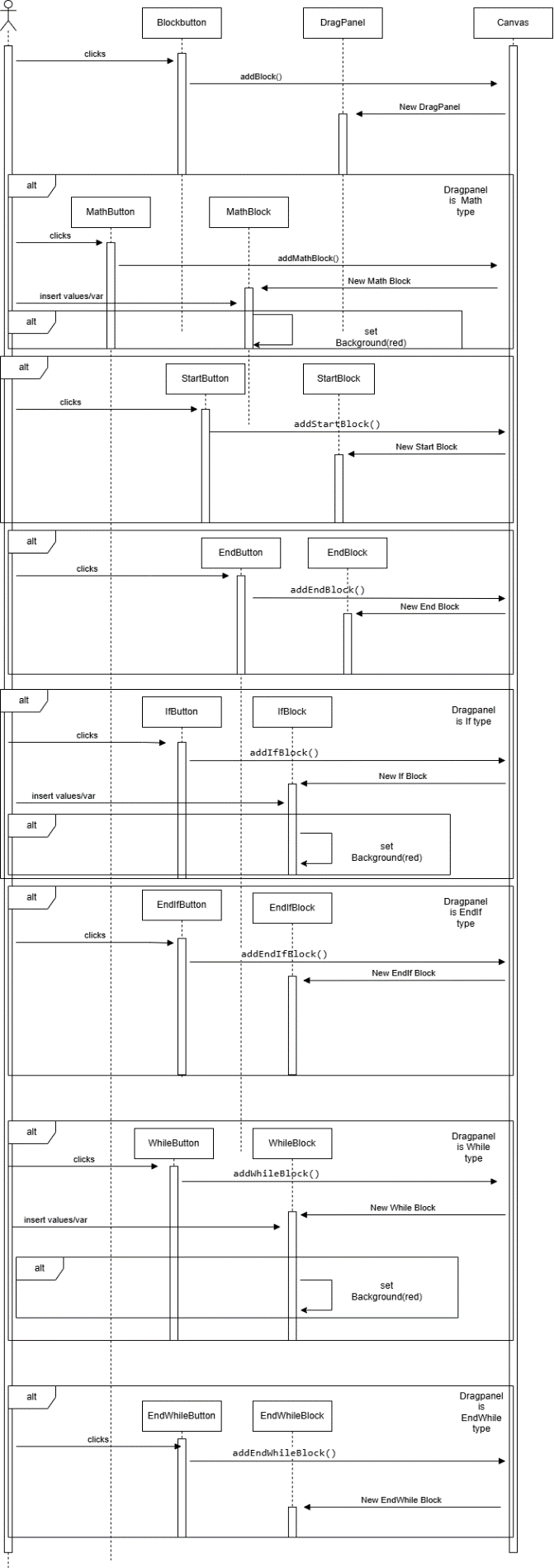
**GhostBlock:** Προεπισκόπηση θέσης μπλοκ πριν την ένωση του μπλοκ στο πρόγραμμα.

# **Sequence-diagram-v1.0**

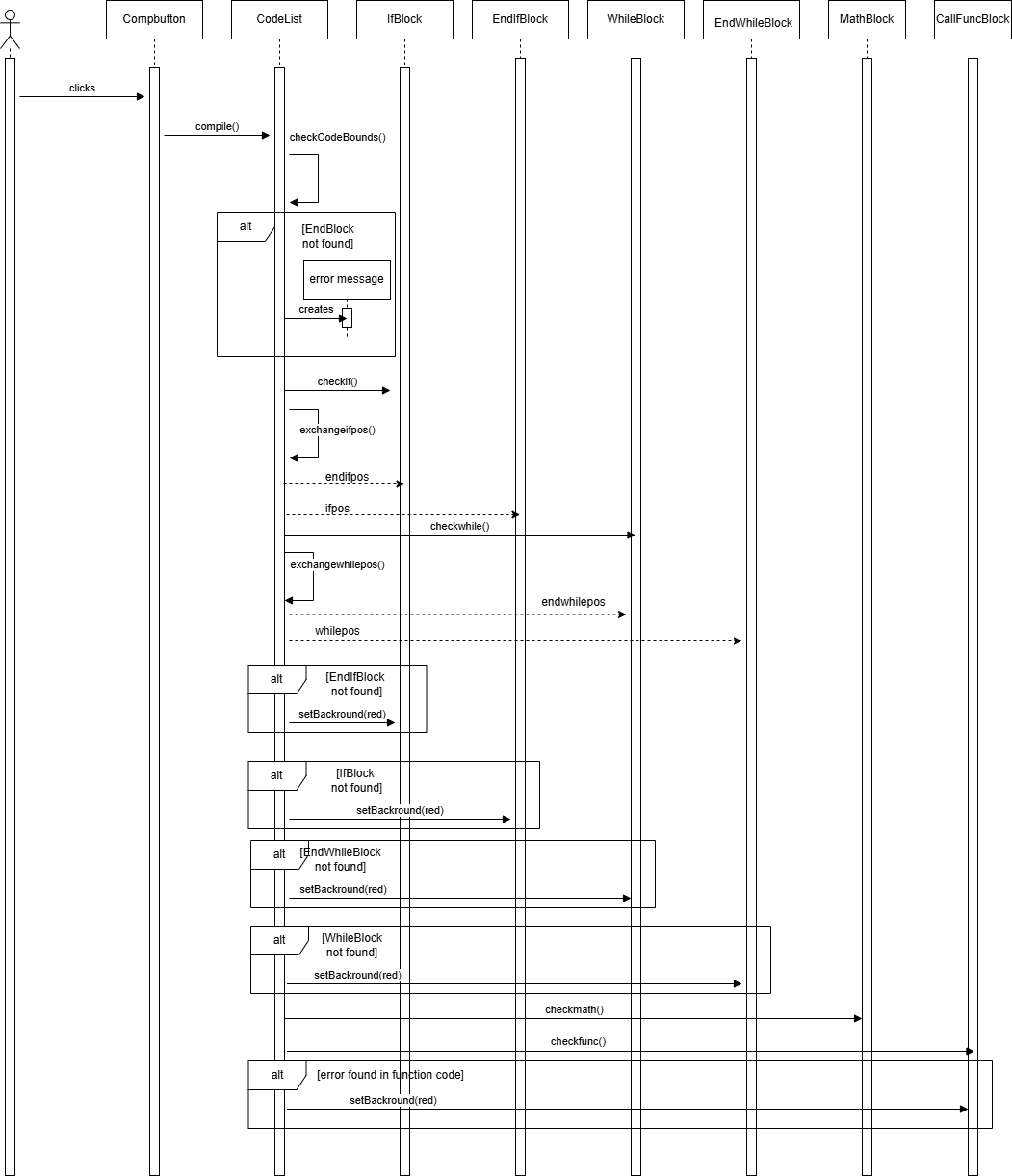
**Τα sequence diagrams είναι ίδια με την προηγούμενη έκδοση blocko-v0.3.**

****

**Δήλωση και αρχικοποίηση μεταβλητών**

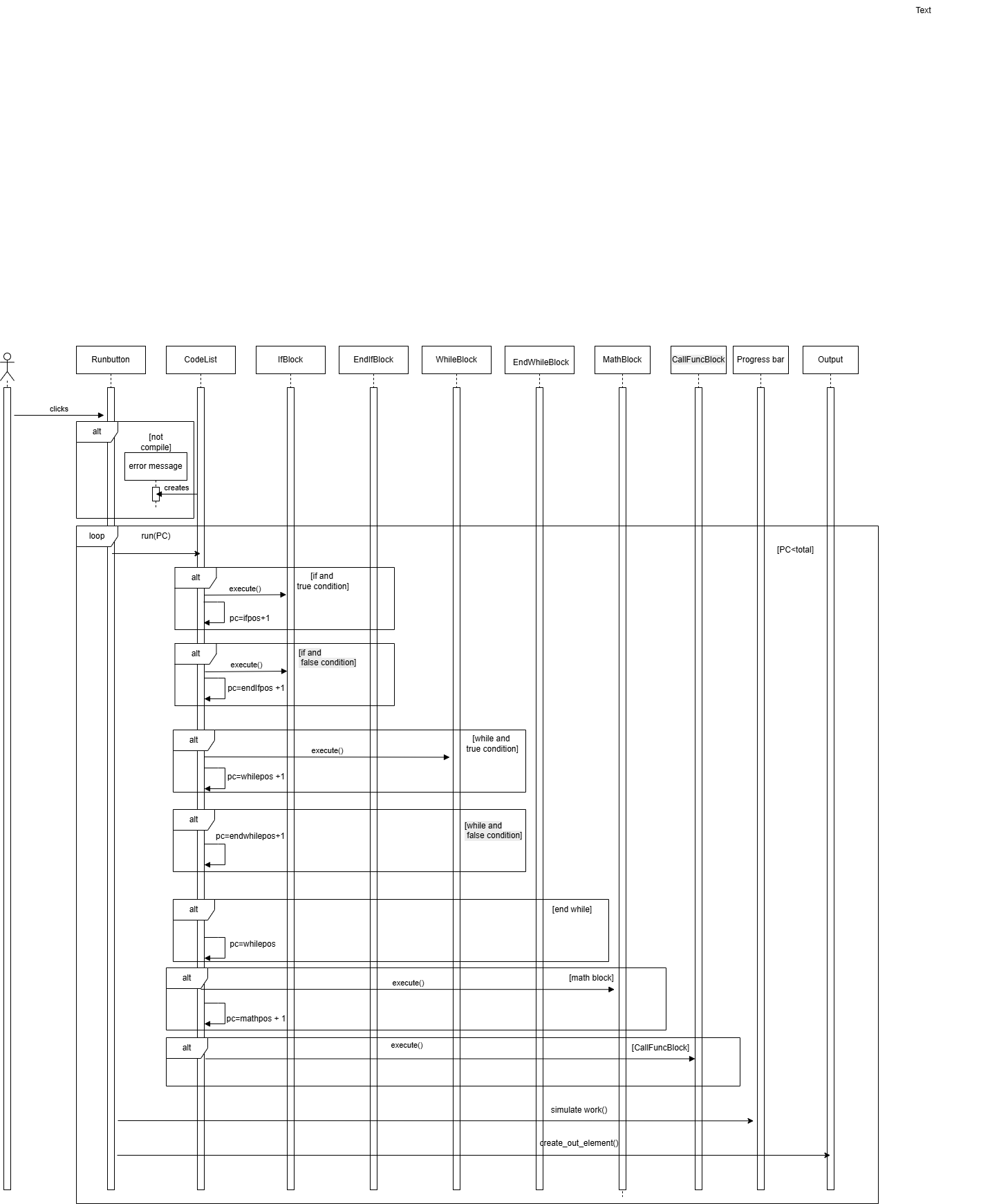
****

**Δημιουργία μπλοκ εντολών**

****

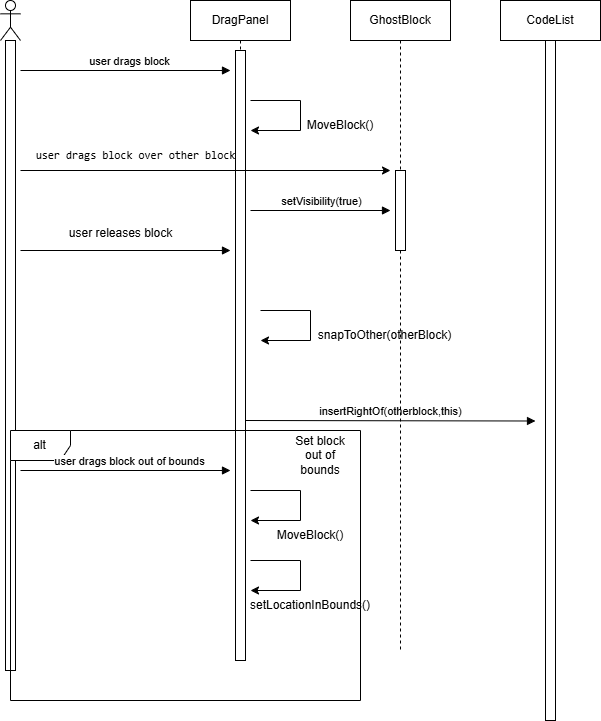
**Compile**

**Σημείωση για use case compile sequence diagram: Λόγω των πολλών συνδυασμών που μπορούν να δημιουργηθούν με τα μπλοκ, αποφασίσαμε να διαλέξουμε αυθαίρετα για την επεξήγηση της συνάρτησης του compile την εξής σειρά: ifBlock, EndifBlock, WhileBlock,EndWhileBlock,MathBlock και τέλος CallFuncBlock.**

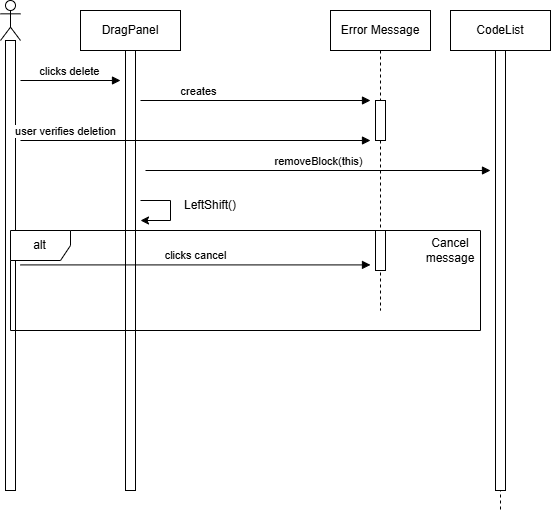
****

**Εκτέλεση**

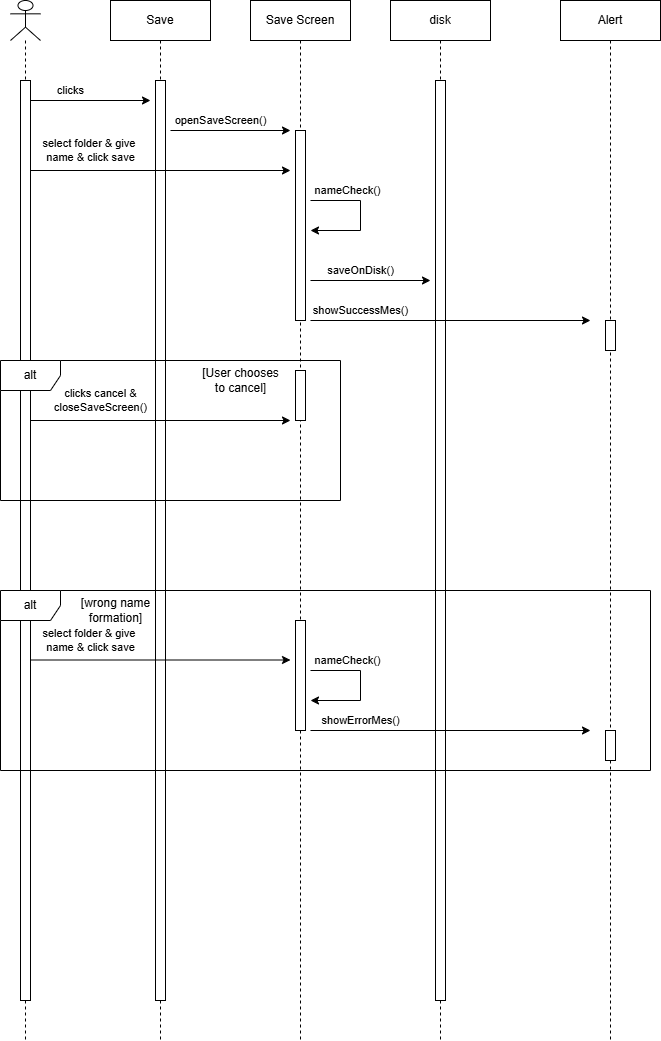
**Το συγκεκριμένο διάγραμμα παρουσιάζει μια επανάληψη ενός βρόγχου while, ο οποίος θα εκτελείται μέχρι ο CodeList.ProgCounter(PC)<CodeList.ProgMax(total)**

****

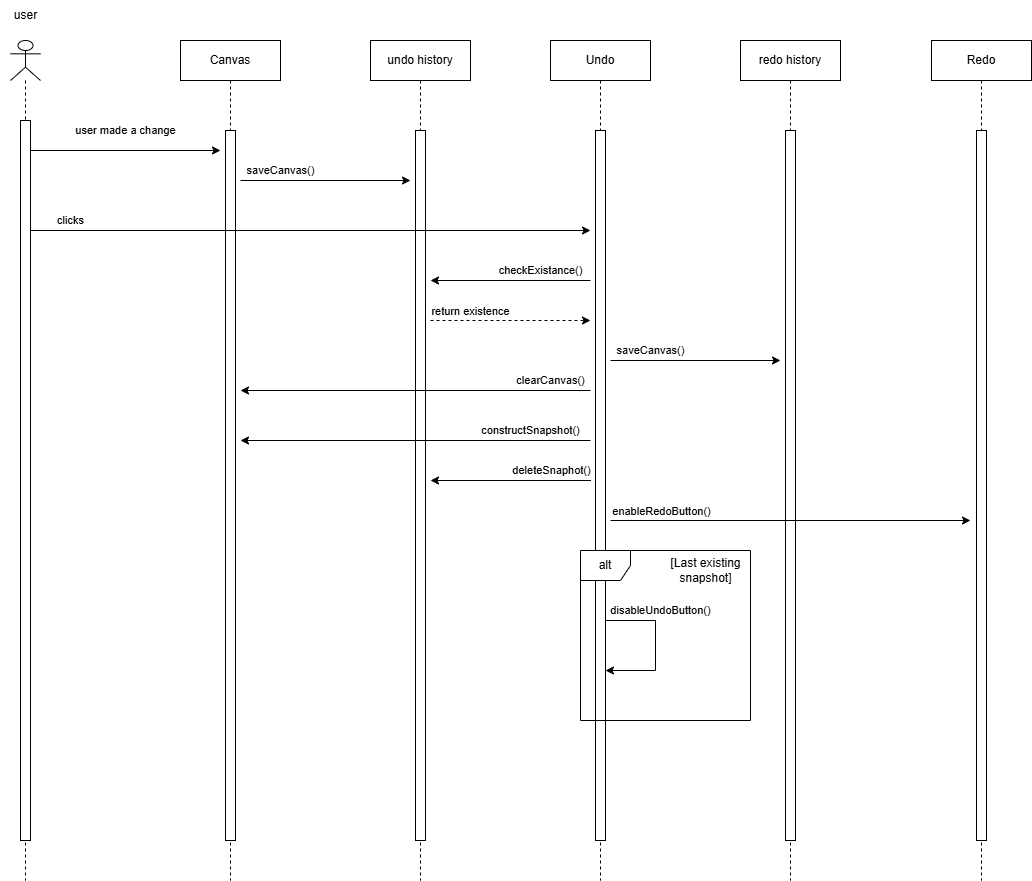
**Ένωση και μετακίνηση block εντολών**

****

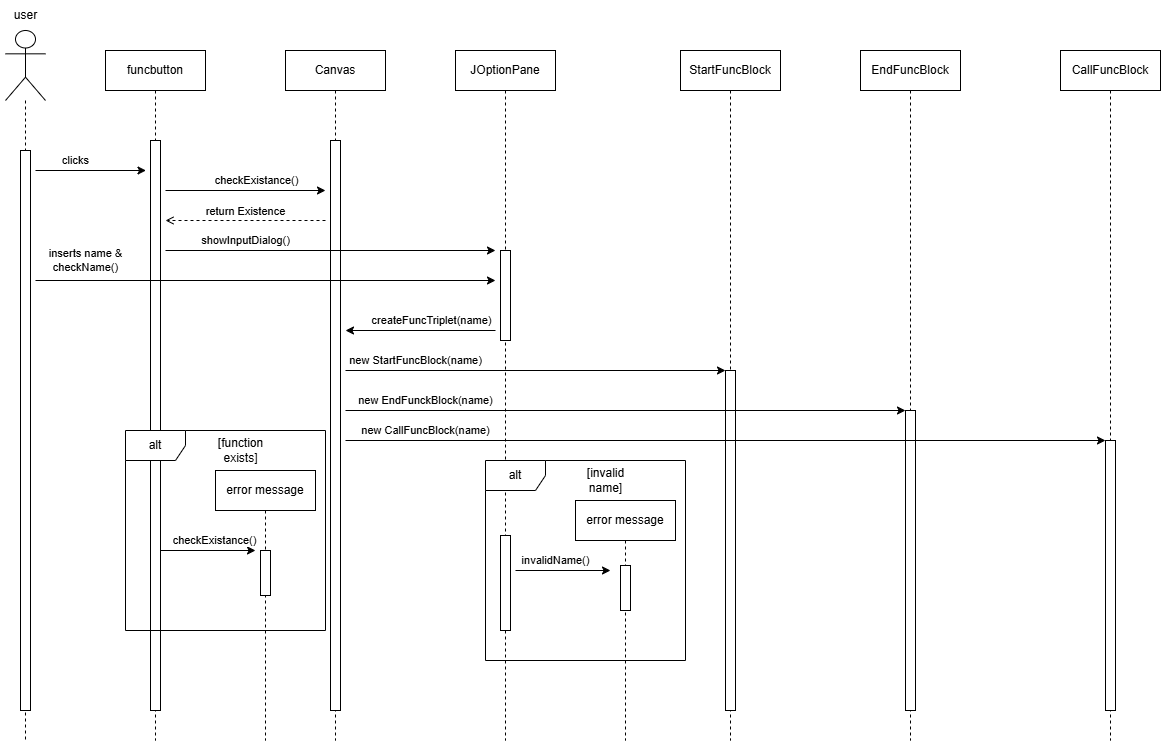
**Διαγραφή block εντολών**

****

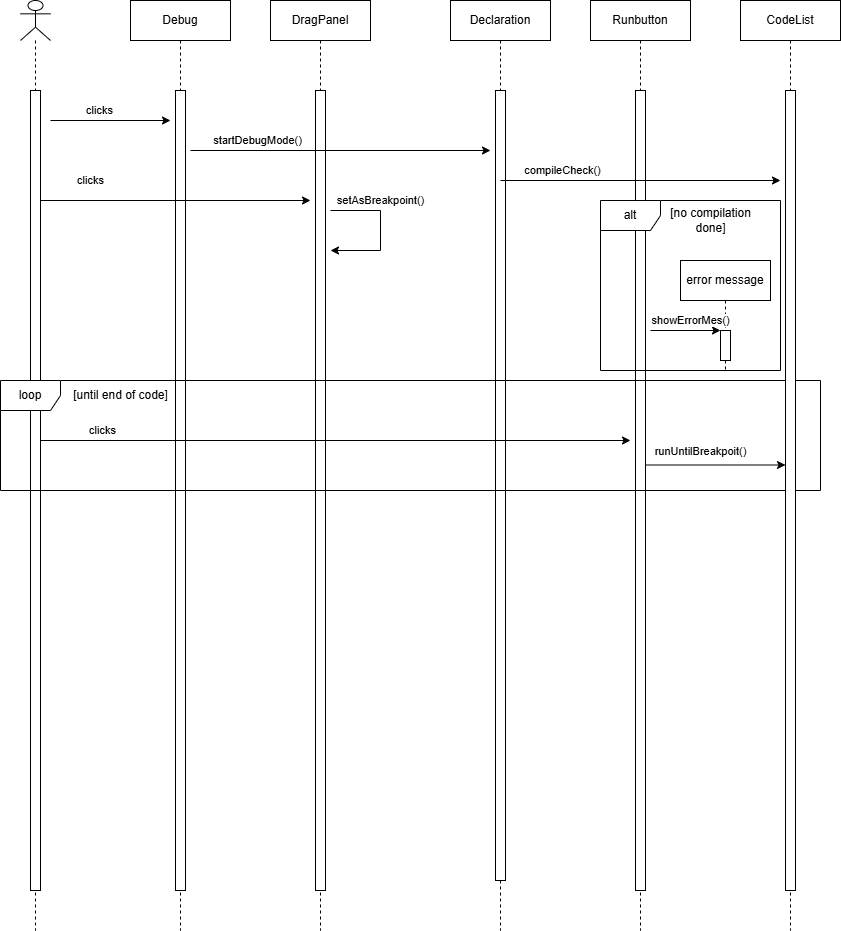
**Αποθήκευση**

****

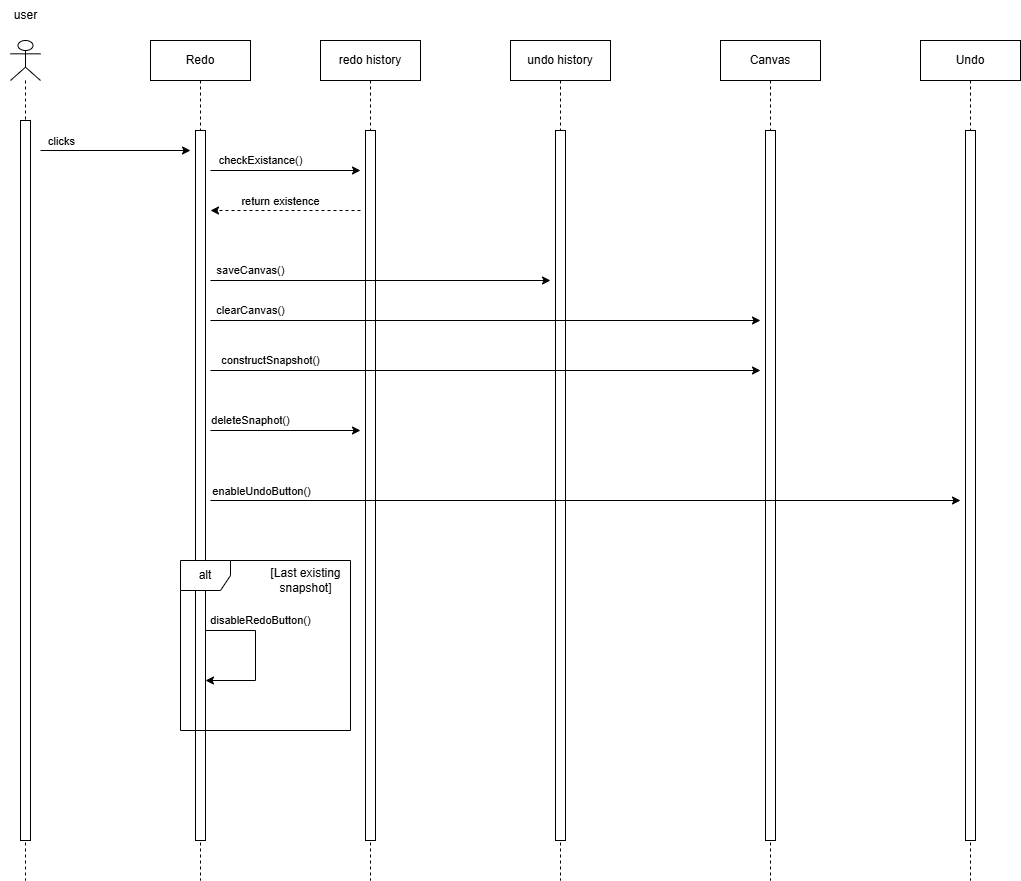
**Undo**

****

**Υπορουτίνα**

****

**Debug**

****

**Redo**

# **Project-code-v1.0**

**Εργαλεία :**

Για την συγγραφή του κώδικα σε java χρησιμοποιήσαμε το eclipse και διάφορο υλικό στο διαδίκτυο πάνω στην βιβλιοθήκη Swing για την υλοποίηση των στοιχείων gui.

**Στo project-code-v1.0 περιέχεται την υλοποίηση των use cases δήλωση και αρχικοποίηση μεταβλητών, δημιουργία μπλοκ εντολών, compile, εκτέλεση, μετακίνηση και ένωση μπλοκ εντολών, διαγραφή εντολών και υπορουτίνα.**

Από την προηγούμενη έκδοση (**project-code-v0.1**) έχουν γίνει οι εξής αλλαγές:

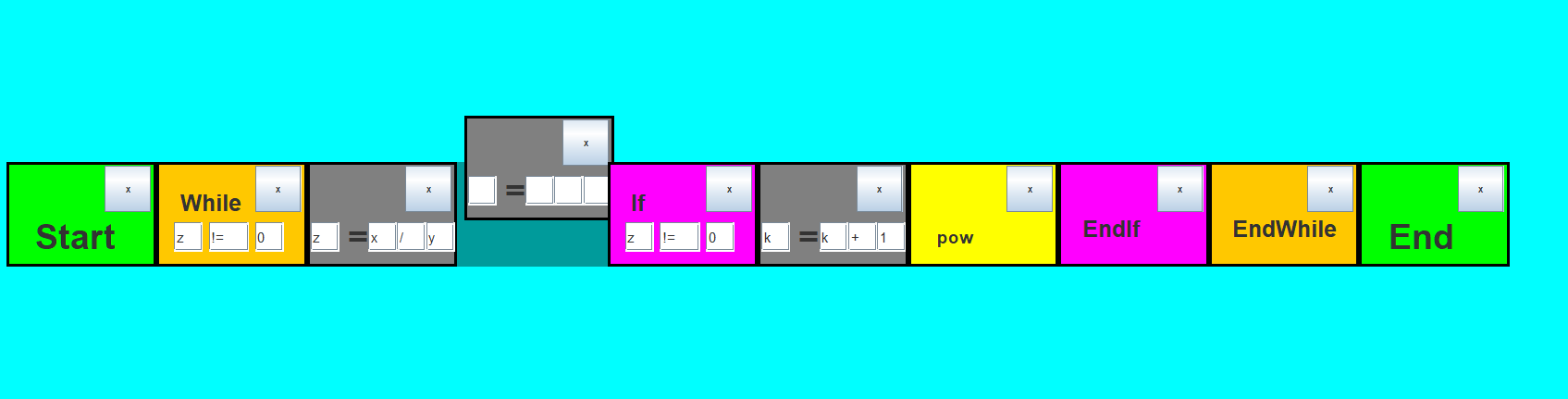
* Όταν γίνεται εισαγωγή τιμών στα μπλοκ εντολών θα γίνεται αυτόματα και ο συντακτικός τους έλεγχος.
* Έγινε τελειοποίηση του μηχανισμού εισαγωγής μπλοκ στην μετακίνηση και ένωση μπλοκ εντολών.
* Διόρθωση του compile και της εκτέλεση.
* Δημιουργία της υπορουτίνας .

**Git code link:**

# <https://github.com/VasilisMichanetzis/Software-engineering-project-2025/tree/main/code/Project-code-v1.0/gui1>

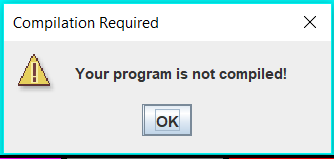
# **Test-cases-v1.0**

Στο συγκεκριμένο test case φαίνεται η λειτουργία ενός προγράμματος που υλοποιείται μέσω της εφαρμογής μας όπου μετράει το πλήθος ψηφίων ενός ακέραιου αριθμού καθώς και παρουσιάζουμε πιθανά σφάλματα που μπορούν να προκύψουν κατά την συγγραφή του προγράμματος από τον χρήστη.

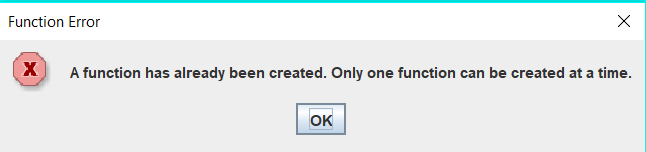


Αρχικά, φαίνεται η λειτουργία του snapping ενός μπλοκ το οποίο ο χρήστης θέλει να τοποθετήσει στο ενδιάμεσο μίας ήδη υπάρχουσας λίστας προγράμματος και βλέπουμε πως τα μπλοκ μετακινούνται προκειμένου να μπει στην σωστή θέση το μπλοκ. Αντίστοιχα λειτουργεί και η περίπτωση που ο χρήστης επιλέγει να αφαιρέσει ένα ενδιάμεσο μπλοκ από την λίστα.

Παραθέτουμε δυο σφάλματα που μπορεί να κάνει ο χρήστης κατά την αλληλεπίδραση του με την εφαρμογή.



Ο χρήστης έχει προβεί σε αλλαγές στο πρόγραμμα του και προσπαθεί να το τρέξει χωρίς να έχει προηγηθεί compile με αποτέλεσμα να εμφανίζεται μήνυμα σφάλματος το οποίο τον ενημερώνει για το λάθος του.

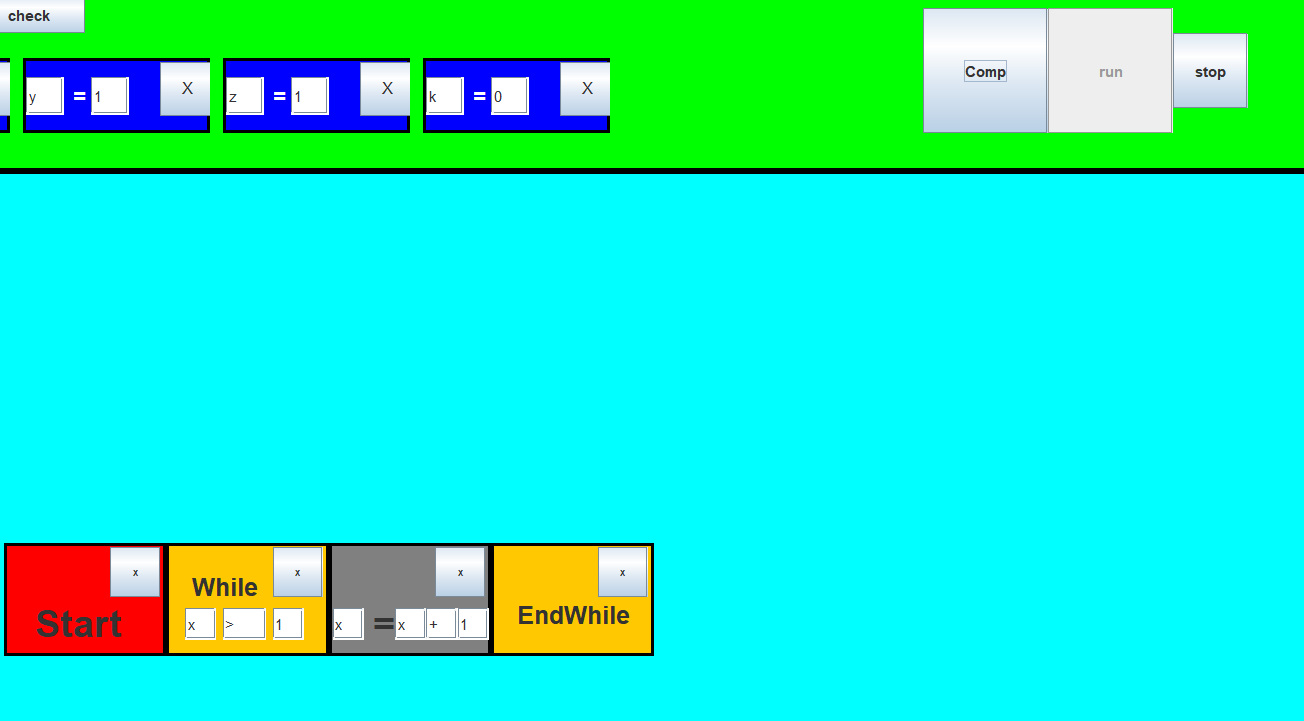


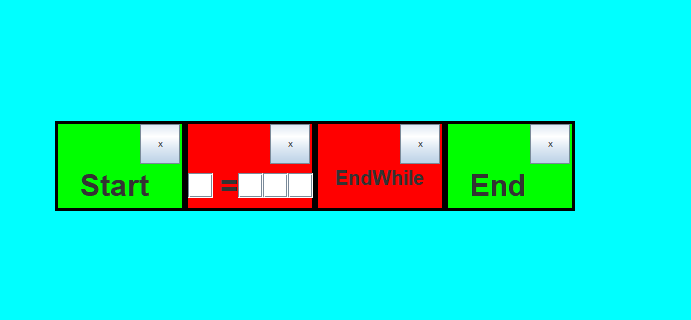
Ο χρήστης προσπαθεί να φτιάξει δεύτερη υπορουτίνα (function) και εμφανίζεται μήνυμα σφάλματος το οποίο ενημερώνει τον χρήστη πως υπάρχει ήδη μία υπορουτίνα και δεν μπορεί να δημιουργήσει καινούρια.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, ορθογώνιο παραλληλόγραμμο, πολυχρωμία

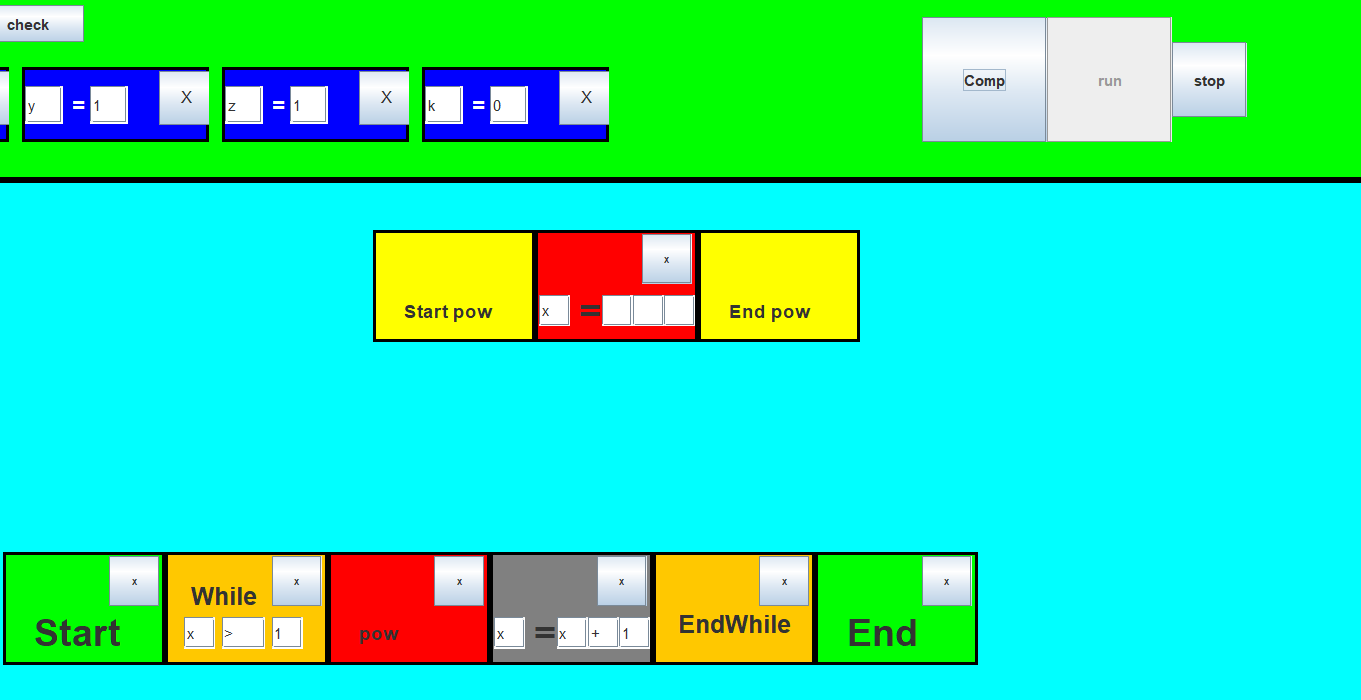
Το περιεχόμενο που δημιουργείται από τεχνολογία AI ενδέχεται να είναι εσφαλμένο.

Ο χρήστης εισάγει στις μεταβλητές μη έγκυρες τιμές ή μη έγκυρα ονόματα τα οποία αποτελούν συντακτικά λάθη με αποτέλεσμα να χρωματίζονται κόκκινα τα μπλοκ ως ένδειξη των σφαλμάτων στο πρόγραμμα.

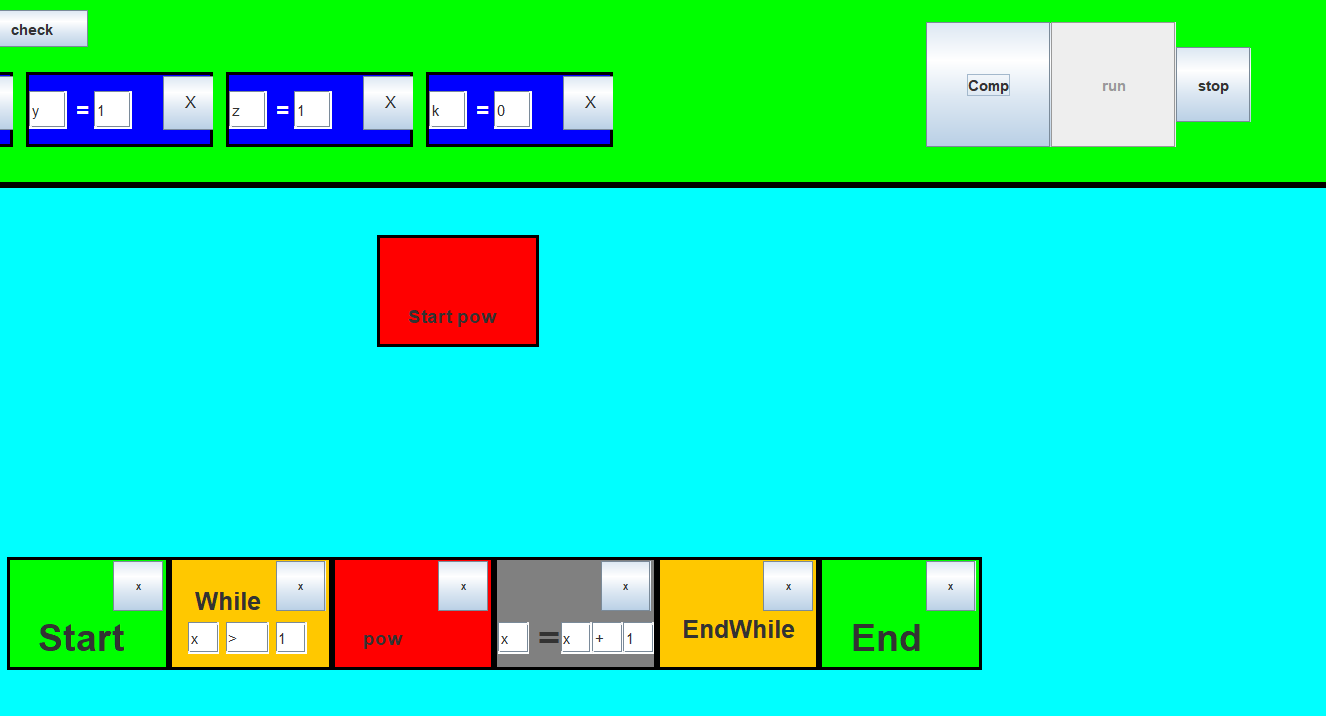




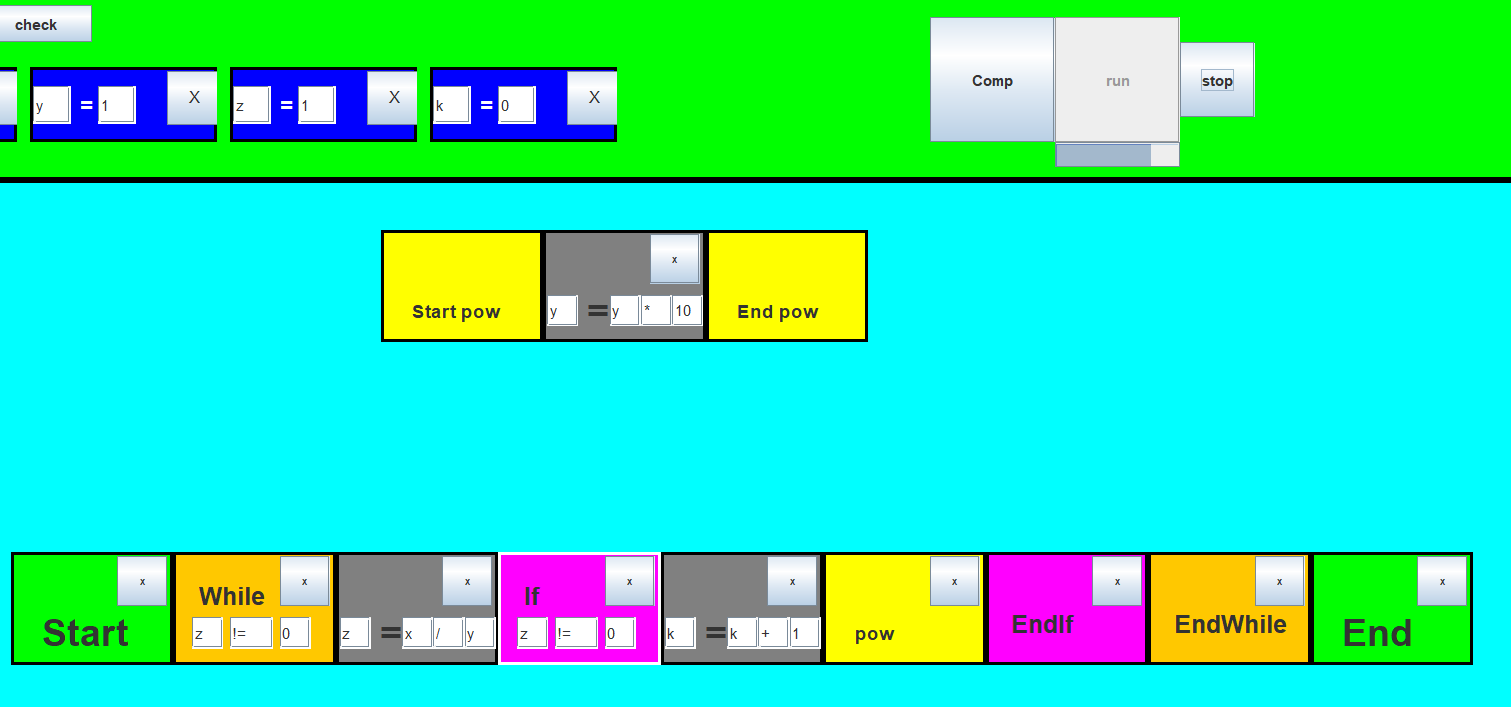
Όταν ο χρήστης χρησιμοποιεί μια εντολή που αποτελείται από δύο μέρη, όπως για παράδειγμα ένας βρόγχος While (While-EndWhile) ή μια δομή If (If-EndIf) και το ένα από τα δύο μέρη λείπει από την λίστα του προγράμματος τότε τα μπλοκ του προγράμματος χρωματίζονται κόκκινα και απενεργοποιείται το κουμπί run.



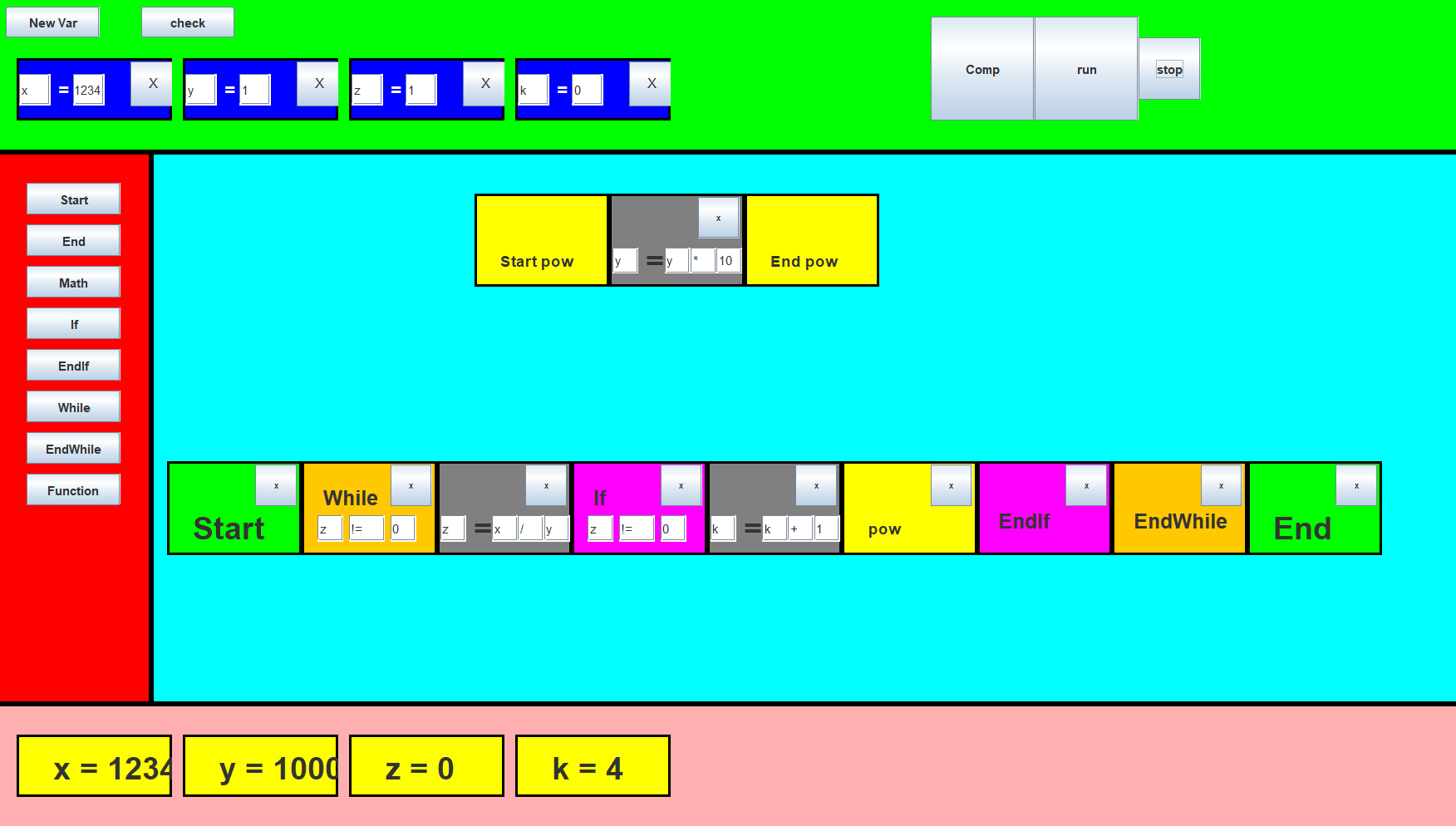
Έστω ότι ένας χρήστης δημιουργεί μια υπορουτίνα την οποία έχει ονομάσει pow. Αν υπάρχει σφάλμα μέσα στην λίστα του function τότε γίνεται κόκκινο το σφάλμα και το μπλοκ που καλεί την function στο main πρόγραμμα .



Στο συγκεκριμένο παράδειγμα ο χρήστης έχει κάνει δυο λάθη στο πρόγραμμα του, τα οποία είναι η έλλειψη του μπλοκ end function και σφάλμα μέσα στην λίστα του προγράμματος.



Το συγκεκριμένο παράδειγμα ο χρήστης έχει δημιουργήσει επιτυχώς το πρόγραμμα του και καθώς γίνεται η εκτέλεση του προγράμματος το progress bar αντικατοπτρίζει την καθυστέρηση μέχρι την εκτέλεση της επόμενης εντολής και η τρέχουσα εντολή που εκτελείται έχει φωτεινό περίγραμμα.



Τέλος φαίνεται το ολοκληρωμένο πρόγραμμα μαζί με τα αποτελέσματά του.

Παρακάτω φαίνεται ο κώδικας στην κάθε επανάληψη τoυ.

x = 1234; // Αρχικά x = 1234

y = 1; // Αρχικά y = 1

z = 1; // Αρχικά z = 1

k = 0; // Αρχικά k = 0

pow() { // Function: πολλαπλασιασμός του y επί το 10

y = y \* 10;

}

while( z != 0 ) { // Πρώτος έλεγχος : z = 1 ≠ 0 → μπαίνει στη λούπα

z = x / y; // Επανάληψη 1: z = 1234 / 1 = 1234

if( z != 0 ) {

k = k + 1; // k = 0 + 1 = 1

pow(); // y = 1 \* 10 = 10

}

z = x / y; // Επανάληψη 2: z = 1234 / 10 = 123

if( z != 0 ) {

k = k + 1; // k = 1 + 1 = 2

pow(); // y = 10 \* 10 = 100

}

z = x / y; // Επανάληψη 3: z = 1234 / 100 = 12

if( z != 0 ) {

k = k + 1; // k = 2 + 1 = 3

pow(); // y = 100 \* 10 = 1000

}

z = x / y; // Επανάληψη 4: z = 1234 / 1000 = 1

if( z != 0 ) {

k = k + 1; // k = 3 + 1 = 4

pow(); // y = 1000 \* 10 = 10000

}

z = x / y; // Επανάληψη 5: z = 1234 / 10000 = 0

if( z != 0 ) {

// δεν μπαίνει στο if —> z == 0

}

} // Τέλος λούπας — z = 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteration** | **y** | **z = x / y** | **z != 0?** | **k (if z ≠ 0)** | **pow() → y** |
| 1 | 1 | 1234 / 1 = 1234 | Yes | 1 | 10 |
| 2 | 10 | 1234 / 10 = 123 | Yes | 2 | 100 |
| 3 | 100 | 1234 / 100 = 12 | Yes | 3 | 1000 |
| 4 | 1000 | 1234 / 1000 = 1 | Yes | 4 | 10000 |
| 5 | 10000 | 1234 / 10000 = 0 | **No** | - (loop ends) | - |