Τεχνολογία Πολυμέσων
Ροή Υ: Υπολογιστικά Συστήματα

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

Τελική εργασία στο μάθημα
Τεχνολογία Πολυμέσων

Δάσκος Ραφαήλ – 03116049

9ο εξάμηνο, 2020-2021

Εισαγωγή

Η παρούσα εργασία αναπτύχθηκε στο Eclipe IDE με τη χρήση JDK 14 και JavaFX δεδομένου ότι δεν είναι built in η JavaFX από τη Java 11. Πιθανό είναι να τρέχει και με άλλες εκδόσεις της Java και ανάλογα configuration για τα VM arguments. Η γραμμή που τρέξαμε ως argument στο VM είναι η ακόλουθη:

--module-path path-to-this-file\openjfx-11.0.2_windows-x64_bin-sdk\lib --add-modules=javafx.controls,javafx.fxml

Με το module-path να μπει η διαδρομή για το αρχείο που του javaFX. Εγώ για να το τρέξω κατέβασα από τον ακόλουθο σύνδεσμο:

https://gluonhq.com/products/javafx/

Σε περίπτωση που δεν έχει τη JavaFX χρειάζεται να δημιουργηθεί νέα βιβλιοθήκη με τα jars που υπάρχουν στο lib της JavaFX στο workplace του eclipse που δουλεύουμε. https://stackoverflow.com/questions/52144931/how-to-add-javafx-runtime-to-eclipse-in-java-11.

Επιπλέον να πούμε ότι η κλάση που επέλεξα να τεκμηριώσω με τις προδιαγραφές του εργαλείου javadoc είναι η DecisionBox.java η οποία είναι υπεύθυνη, όπως λέμε και μέσα στο αρχείο (και θα αναλύσουμε και παρακάτω) για την επιλογή του επόμενου scenario id αν επιθυμεί ο χρήστης να αλλάξει.

Υλοποίηση

Να πούμε ότι όλες οι ζητούμενες λειτουργικότητες είναι υλοποιημένες αλλά έχουμε και μια επιπλέον που είναι στο Menu -> Application -> App Info που μας δίνει ορισμένες απλές πληροφορίες για το πώς είναι υλοποιημένα και πως λειτουργούν κάποια από τα πράγματα κατά τη διάρκεια του παιχνιδιού μας.

Για να τρέξει σωστά η εφαρμογή χρειάζεται ο χρήστης της εφαρμογής να έχει προσθέσει στο φάκελο src/medialab τα αρχεία ενεργοποίησης του γύρου με τη μορφή player_gameID.txt και enemy_gameID.txt όπου το gameID είναι ο αριθμός που πρέπει βάζει ο χρήστης στην ενεργοποίηση του παιχνιδιού όταν του το ζητήσει η εφαρμογή (είτε στην αρχή είτε στο load). Ακόμα θεωρούμε ότι αν υπάρχουν αυτά τα αρχεία έχουν συμπληρωθεί σωστά, δηλαδή, έχουμε 5 γραμμές οι οποίες έχουν και τα ζητούμενα στοιχεία που είτε θα τοποθετούν σωστά τα πλοία στους πίνακες ή θα μας εμφανίζεται με μορφή popup παραθύρου κάποιο exception εξαιτίας λάθος τοποθέτησης όπως μας ζητάται.

Επίσης έχουμε υποθέσει ότι οι βολές του παιχνιδιού θα γίνονται εναλλάξ από τον παίκτη και από τον υπολογιστή μέχρι είτε να ολοκληρωθούν 40 βολές από τον καθένα, και νικητής είναι αυτός που έχει τους περισσότερους πόντους, είτε να βυθιστούν όλα τα πλοία του ενός από τους παίκτες. Ακόμα η επιλογή του αρχικού παίκτη γίνεται τυχαία με τη χρήση της

Math.random() που μας δίνει τιμές μεταξύ 0 – 1 και αναλόγως με το αν είναι μεγαλύτερο του 0.5 διαλέγουμε ποιος θα αρχίσει.

Στη γραφική διεπαφή που βλέπει ο χρήστης πραγματοποιεί τις κινήσεις του με τη χρήση των επιλογών column (στήλη) και row (γραμμή) και πατώντας το κουμπί attack Που πραγματοποιεί την κίνησή του αν γίνεται στο ταμπλό του αντιπάλου. Αν δε γίνεται θα ενημερώνεται με μήνυμα. Ο αντίπαλος κάνει αμέσως την κίνησή του μέσω random επιλογής εκτός αν έχει ήδη πετύχει πλοίο που τότε αρχίζει να κάνει κινήσεις κάθετα και μετά οριζόντια. Δεν έβαλα μεγαλύτερη ευφυία στο σύστημα ώστε να κοιτάει εάν έχει λογική η επίθεση που πάει να κάνει. Ακόμα χρειάζεται να πούμε πως οι επιθέσεις που παραμένουν και εμφανίζεται ο αριθμός τους στο κάτω μέρος αφορούν το χρήστη. Δηλαδή ανανεώνονται μετά από κίνηση του και όχι του υπολογιστή. Αυτό σημαίνει ότι ο υπολογιστής έχει ακόμα είτε ίδιο αριθμό κινήσεων (αν άρχισε πρώτος ο χρήστης) είτε μία λιγότερη γιατί έχει κάνει ήδη τη δική του.

Τεχνικά κομμάτια

Στη συνέχεια θα αναλύσουμε τις κλάσεις που έχουμε δημιουργήσει, και τις μεθόδους τους αλλά όχι όλες τις μεταβλητές που έχουμε, ώστε να είναι πιο εύκολο σε κάποιον που θέλει να δει την υλοποίηση να καταλάβει τι έχουμε ορίσει. Όλες οι κλάσεις είναι υλοποιημένες στο φάκελο src και τους αντίστοιχους υποφακέλους που θα αναλύσουμε στη συνέχεια,

Υποφάκελος application

Main.java: είναι υπεύθυνη για όλες τις βασικές λειτουργίες του παιχνιδιού αλλά και την υλοποίηση εμφανίσιμου GUI για να παίξει ο χρήστης μας. Η υλοποίηση του GUI (εμφανισιακά είναι όπως μας ζητάται, δηλαδή ένα Menu bar με τα αιτήματα που θέλουμε, ένα μέρος με πληροφορίες για τα εναπομείναντα πλοία, πόντους και ποσοστό επιτυχημένων βολών για τους παίκτες, τα ταμπλό των παικτών και η δυνατότητα να ρίξει βολής ο χρήστης και να δει πόσες απομένουν μέχρι να λήξει το παιχνίδι) γίνεται μέσω της μεθόδου start που φτιάχνει το κατάλληλο παράθυρο για το παιχνίδι και δρομολογεί όλα τα πατήματα κουμπιών στις ανάλογες συναρτήσεις προς υλοποίησή τους.
Στη συνέχεια έχουμε μία μέθοδο που υλοποιεί έναν separator για να μην είναι κολλημένα τα ταμπλό όταν μεγαλώσει το παράθυρο και ακολούθως όλες τις μεθόδους που καλούνται όταν πατηθεί ένα κουμπί από το menu που με τη σειρά τους καλούν μια εκ των συναρτήσεων που βρίσκονται σε κάποια από τις κλάσεις των μοντέλων που θα εξηγήσουμε στη συνέχεια. Αναλόγως με το ποια θέλουμε να υλοποιήσουμε φτιάχνουμε τα κατάλληλα μηνύματα και τις κατάλληλες απαντήσεις. Ιδιαίτερο ενδιαφέρον έχουν η load και η start. Και οι δύο παίρνουν από τα αρχεία που αντιστοιχούν στο Scenario id, η πρώτη που θα δώσουμε στο popup που θα εμφανιστεί, και η δεύτερη που έχει και τρέχει ήδη στο παιχνίδι. Για την πρώτη αν δώσουμε λάθος id δε μας χαλάει το παιχνίδι που παίζαμε οπότε

και μπορούμε να το συνεχίσουμε και για τη Start αν δεν είχαμε δώσει ποτέ id τότε μας λέει ότι πρώτα πρέπει να δώσουμε. Ταυτόχρονα κάνουν restart (με τη χρήση της restartVariables) όλες τις μεταβλητές του συστήματος (σε επιτυχημένη κλήση τους) και ξαναρχίζει νέο παιχνίδι που πάλι διαλέγουμε νέο πρώτο παίκτη.
Έπειτα έχουμε τις συναρτήσεις playerMove και enemyMove που υλοποιούν τις κινήσεις του κάθε παίκτη, η πρώτη μετά το πάτημα του κουμπιού και η δεύτερη μετά την κίνηση του παίκτη. Η enemyMove μπορεί να κληθεί και στην αρχή εάν αρχίζει το παιχνίδι από τον υπολογιστή.
Τέλος έχουμε την check_end που βλέπει ανανεώνει τις εναπομείναντες βολές και κοιτάει αν έχει τελειώσει το παιχνίδι εξαιτίας των 40 βολών. Ο έλεγχος για το αν έχουν μείνει πλοία για τον παίκτη γίνεται στις προηγούμενες δύο συναρτήσεις αμέσως μετά τις βολές του καθενός.
Υπάρχει και ένα αρχείο application.css από κατασκευή του JavaFx project που όμως δε χρησιμοποιήθηκε.

Υποφάκελος exceptions

Σε αυτόν τον φάκελο έχουμε υλοποιήσει όλα τα custom exceptions που θέλουμε να υπάρχουν όταν για κάποιο λόγο η τοποθέτηση των πλοίων μας είναι λάθος. Οι κλάσεις αυτές κάνουν extend από την γενική κλάση Exception που υπάρχει και έχουν ίδιο κώδικα με αλλαγές μόνο στο όνομά τους και στο μήνυμα που αλλάζει το super() που έτσι ορίζεται το ανάλογο μήνυμα να εμφανιστεί. Έχουμε λοιπόν:

- OversizeException: πετάγεται (throw) όταν κάποιο πλοίο βγει εκτός ορίων του πίνακα και εμφανίζει το μήνυμα "Ship placement out of bounds"
- OverlapTilesException: πετάγεται όταν πάμε να τοποθετήσουμε πλοίο σε σημείο που έχει ήδη τοποθετηθεί κάποιο άλλο και εμφανίζει το μήνυμα "Overlaping ships in grid"
- AdjacentTilesException: πετάγεται όταν πάει ένα πλοίο να μπει σε γειτονικό κελί με ένα που έχει ήδη τοποθετηθεί και εμφανίζει το μήνυμα "Two ships are adjacent"
- InvalidCountExeception: πετάγεται (throw) όταν κάποιο πλοίο βγει εκτός ορίων του πίνακα και εμφανίζει το μήνυμα "Ship placement out of bounds"
- InvalidInputExeception: είναι ένα δικό μας exception έξτρα αυτών που μας ζητούνταν και πετάγεται όταν για κάποιο λόγο τα στοιχεία των αρχείων μας είναι λάθος. Δηλαδή οι συντεταγμένες πάνε να αρχίσουν έξω από τα όρια του πίνακα, τα πλοία δεν έχουν τύπο 1-5 ή η κατεύθυνση δεν είναι 1 ή 2. Επίσης επειδή καλύπτει 3 διαφορετικά σενάρια, ο constructor του λαμβάνει και τον τύπο που έχει προκαλέσει το λάθος και εμφανίζει μετά το μήνυμα "Invalid" + errorType + "input" όπου το errorType είναι ένα εκ των coordinate, type, orientation όπως εξηγήσαμε προηγουμένως.

Χρειάζεται να πούμε πως είναι πιθανό κάποιο αρχείο να έχει περισσότερα από ένα σφάλματα, όμως εμείς θα εμφανίσουμε μόνο το πρώτο και δε θα κοιτάξουμε το υπόλοιπο.

Υποφάκελος models

Board.java: σε αυτή την κλάση υλοποιείται το κάθε ταμπλό. Έχει ένα GridPane με τις θέσεις του ταμπλό και τα labels γραμμών και στηλών, μια μέθοδο που γυρίζει αν το πλοίο είναι ζωντανό ή όχι καλώντας την αντίστοιχη συνάρτηση από την Ship. Μια μέθοδο τοποθέτησης πλοίων (placeShips) που είτε φτιάχνει το ταμπλό είτε πετάει Exception από αυτά που αναλύσαμε προηγουμένως. Τέλος, μέσα σε αυτή ορίζεται και η κλάση Cell (θα μπορούσε να γίνει και ξεχωριστή βέβαια) που είναι το δομικό στοιχείο του κάθε πίνακα κάνοντας extend το Rectangle για ευκολότερο σχήμα.

ConfirmBox.java: είναι η κλάση υπεύθυνη για την εμφάνιση παραθύρου με yes-no απάντηση και αντιστοίχως με το αποτέλεσμα επιστρέφει είτε true είτε false.

DecisionBox.java: είναι υπεύθυνη για την αλλαγή scenario (έχει κληθεί από τη load ή στην αρχή) και αναζητεί τα αρχεία αν υπάρχουν.

DisplayBox.java: η μέθοδος displayShips παίρνει τα στοιχεία για την κατάσταση των πλοίων του αντιπάλου και εμφανίζει στο χρήστη πληροφορίες για το αν είναι χτυπημένα, ανέπαφα ή βυθισμένα ενώ η displayShots εμφανίζει από 0 (αν είμαστε ακόμα στην αρχή) έως 5 τελευταίες βολές από τον χρήστη ή τον υπολογιστή αναλόγως με το πώς έχει κληθεί με την πιο πρόσφατη στην κορυφή. Σε περίπτωση επιτυχίας δείχνει τον τύπο του πλοίου που πετύχε.

Information.java: είναι υπεύθυνη για το παράθυρο πληροφοριών όταν πατηθεί από το Menu το App Info και αναλόγως με την επιλογή προσπαθεί να βοηθήσει τον χρήστης να καταλάβει τα βασικά στοιχεία που υπάρχουν.

Messages.java: είναι υπεύθυνη για την εμφάνιση pop up παραθύρων είτε για σφάλματα είτε για έξτρα πληροφορίες όπως για παράδειγμα όταν τελειώνει ή αρχίζει το παιχνίδι.

OpponentNextMove.java: κρατάει τις επόμενες βολές του υπολογιστή σε περίπτωση που έχει πετύχει κάποιο πλοίο. Ξεκινάει από κενές κινήσεις οριζόντια και κάθετα και αν του επιτρέπει κάνει πρώτα τις κάθετες κινήσεις και μετά τις οριζόντιες. Σε περίπτωση, όμως, που δεν πετύχει κάτι κάθετα ξέρει ότι είναι οριζόντια ενώ αν πετύχει κάθετα συνεχίζει μόνο κάθετα (ο έλεγχος γίνεται με τη βοήθεια της vertical). Τέλος η clear καλείται όταν βυθιστεί το πλοίο και τότε αδειάζουν και οι δύο ουρές και περιμένουν για επόμενη επιτυχημένη βολή του υπολογιστή (αυτό γίνεται στην κλάση Main που παίζει το παιχνίδι).

Ship.java: είναι ένας απλός ορισμός για τα πλοία και διαθέτει τα χαρακτηριστικά όπως είναι ο τύπος και η ζωή του πλοίο με μεθόδους getters, setters γι’ αυτά αλλά και δυνατότητα να μειωθεί η ζωή με τη συνάρτηση hit() και να δούμε αν το πλοίο είναι ακόμα ζωντανό με την alive()

Shot.java: είναι υπεύθυνη για τις βολές που γίνονται. Έχει τις συντεταγμένες αλλά και αν ήταν επιτυχημένη και τον τύπο του πλοίου που πέτυχε με αντίστοιχους setters και getters για όλες αυτές τις private μεταβλητές.

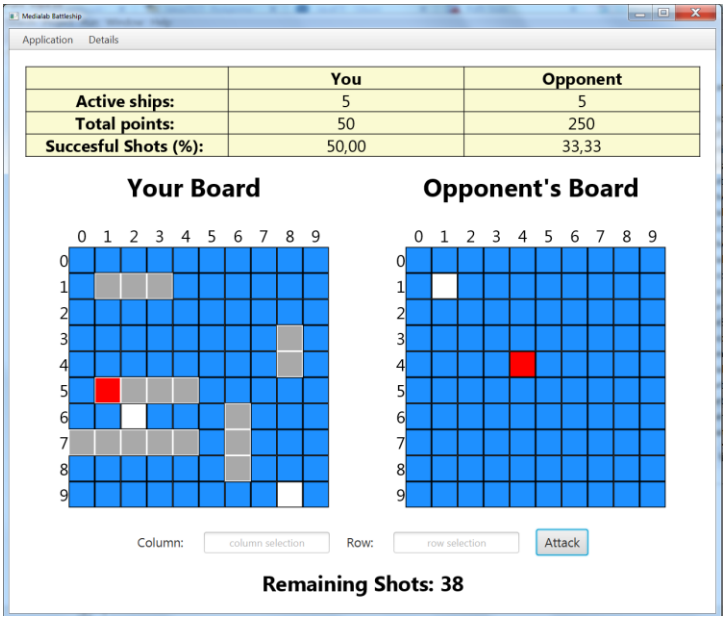
Υποφάκελος players

Player.java: abstract κλάση που ορίζει βασικές ιδιότητες που έχει κάποιος παίκτης όπως το ταμπλό του, τον αριθμό των πλοίων του, τους πόντους, τις κινήσεις και γενικότερα τα δομικά χαρακτηριστικά του παιχνιδιού που αφορούν κινήσεις των παικτών.

User.java: χρησιμοποιεί τα βασικά της κλάσης Player και φτιάχνει ένα ταμπλό που φαίνονται τα πλοία ενώ παράλληλα υπάρχει και η μέθοδος move που κάνει κίνηση στο ταμπλό της Opponent αναλόγως με την είσοδο που έχει δοθεί.

Opponent.java: είναι όμοια με την κλάση User μόνο που αυτή τη φορά οι κινήσεις γίνονται στο ταμπλό του User. Τη λογική ποια κίνηση θα γίνει, όπως έχω πει, έχει υλοποιηθεί στην Main.

Στιγμιότυπο από παιχνίδι που άρχισε πρώτος ο υπολογιστής.



Και παράδειγμα ενός μηνύματος για σφάλμα. Όταν πχ πάει να ρίξει στο ίδιο κελί.