

# **ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ ΓΙΑ ΤΗΝ ΕΦΑΡΜΟΓΗ “BATTLESHIP”**

Δόλογλου Δημήτριος

03116075

## **ΕΙΣΑΓΩΓΗ**

Σκοπός της εργασίας είναι η υλοποίηση και παράδοση μιας εφαρμογής η οποία αποτελεί μια παραλλαγή του κλασσικού παιχνιδιού “Ναυμαχία”. Για την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε ως IDE, το IntelliJ IDEA, και η γλώσσα προγραμματισμού μας ήταν η JavaFX. Συγκεκριμένα, χρησιμοποιήσαμε το default JRE, δηλαδή το 11.0.4 . Για την υλοποίηση του κώδικα μας κάναμε χρήση των βασικών αρχών του OOP design και συμπληρώσαμε με σχόλια κάθε κλάση, σύμφωνα με το javadoc πρότυπο. Στη συνέχεια του παρόντος εγγράφου, περιγράφονται οι παραδοχές που κάναμε για όσα δεν ήταν ξεκάθαρα στην εκφώνηση, καθώς και τον τρόπο που υλοποιήσαμε τα διάφορα ζητούμενα.

## **A. ΠΑΡΑΔΟΧΕΣ**

Για όσα δεν ήταν ξεκάθαρα από την εκφώνηση αλλά και για μερικά κομμάτια στα οποία η ελάχιστη υλοποίηση έκανε το παιχνίδι αρκετά δύσκολο και μη-ρεαλιστικό, έγιναν οι παρακάτω παραδοχές.

- Πρώτη και σημαντικότερη διαφοροποίηση της υλοποίησης μου, αποτελεί ο τρόπος τοποθέτησης των πλοίων. Σύμφωνα με τις προδιαγραφές της εκφώνησης η εισαγωγή των πλοίων και η αρχικοποίηση του παιχνιδιού ξεκινά με το φόρτωμα ενός σεναρίου, καθώς δεν περιγραφόταν εναλλακτικός τρόπος. Αν και κατανοώ πως αυτό έγινε για να καταστεί η εργασία ευκολότερη, εγώ προτίμησα να επιτρέπω στον χρήστη να τοποθετεί τα πλοία του με αριστερό ή με δεξί κλικ στο δικό του Board. Με αυτόν τον τρόπο δεν χρειάζεται, κάθε φορά που επιθυμεί κανείς να παίξει, να φτιάξει ένα αρχείο με σενάριο (και το σενάριο του αντιπάλου γνωρίζοντας έτσι τα πλοία του) και να το φορτώσει. Συγκεκριμένα, με το πάτημα του αριστερού κλικ, τοποθετεί τα πλοία κάθετα, ενώ με το πάτημα του δεξί κλικ, τα τοποθετεί οριζόντια. Στη συνέχεια ο υπολογιστής τοποθετεί τα πλοία του παρόμοια, με τυχαίο τρόπο. Αυτό δεν σημαίνει πως η φόρτωση σεναρίου δεν υλοποιείται ακριβώς όπως ζητήθηκε από την εκφώνηση. Ο χρήστης αν δεν θέλει να βάλει από την αρχή τα πλοία του με κλικ, μπορεί απλά να επιλέξει την επιλογή στο μενού Application > Load και να βάλει το όνομα του σεναρίου που θέλει να φορτώσει.

- Δεύτερη σημαντική παραδοχή, ήταν η επιλογή βολής. Προσωπικά διαφώνησα με την επιλογή στόχου μέσω ενός TextField. Κατανοώ και εδώ πως έγινε για ευκολία, αλλά είναι υπερβολικά χρονοβόρο και καθόλου εύχρηστο, σε αντίθεση με την υλοποίηση μου. Συγκεκριμένα ο χρήστης μπορεί να “βαράει” στο Board του αντιπάλου κατευθείαν με ένα αριστερό κλικ. Η υλοποίηση αυτή, κάνει το παιχνίδι αρκετά πιο ευχάριστο, κάτι που είναι και ο σκοπός του.
- Τρίτη παραδοχή αποτελεί το AI του παιχνιδιού. Καθώς δεν ήταν ξεκάθαρο από την εκφώνηση, πόσο καλός μπορεί να είναι ο υπολογιστής, υλοποιήθηκε η καλύτερη δυνατή συνάρτηση για την λειτουργία του. Συγκεκριμένα, ο υπολογιστής δρα ακριβώς όπως ένας άνθρωπος δηλαδή χτυπάει τυχαία μέχρι να πετύχει κάποιο πλοίο. Στη συνέχεια, αρχίζει και χτυπάει τα διπλανά γειτονικά κελιά μέχρι να βρει τη συνέχεια του πλοίου. Όταν την βρει συνεχίζει προς αυτή την κατεύθυνση και αν δεν έχει βυθίσει το πλοίο, θα συνεχίσει από την ανάποδη.
- Τελευταία παραδοχή και αρκετά μικρότερη είναι η υλοποίηση της ισοπαλίας στις 40 βολές, καθώς δεν είναι απίθανο να έχουν ακριβώς ίδιους πόντους οι παίκτες.

## **B. ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΛΟΓΙΚΗΣ ΚΑΙ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ**

Σύμφωνα με τις παραπάνω παραδοχές, έχουμε ένα εύχρηστο και διασκεδαστικό παιχνίδι ναυμαχίας.

Αυτό υλοποιήθηκε ως εξής: Συγκεκριμένα, αρχικά δημιουργήσαμε την κλάση Ship στο package Models, η οποία περιγράφει τα πλοία του παιχνιδιού, που καταλαμβάνουν κελιά στα ταμπλό μας. Συγκεκριμένα ορίζονται στον constructor τα χαρακτηριστικά του όπως το μήκος του (που είναι και η ζωή του), η κατεύθυνση του, οι πόντοι που δίνει και ο τύπος του. Ακόμη υλοποιούνται οι συναρτήσεις για όταν δεχτεί χτύπημα, για τον έλεγχο αν το πλοίο είναι ζωντανό και για τον έλεγχο αν έχει χτυπηθεί. Οι τελευταίες συναρτήσεις είναι απαραίτητες για την λειτουργικότητα του μενού Details.

Στη συνέχεια ορίσαμε το package GameFiles όπου υλοποιήθηκαν οι κλάσεις Cell και Board. Η κλάση Cell (που θα γίνει populate στα Board των παιχτών προφανώς) είναι τα τετράγωνα των ταμπλό μας. Μέσω του constructor της, ορίζονται τα βασικά στοιχεία του κάθε κελιού όπως οι συντεταγμένες x και y, το ταμπλό στο οποίο βρίσκονται, οι διαστάσεις του και το χρώμα του στη γραφική διεπαφή. Ακόμη ορίζεται η συνάρτηση shoot η οποία ελέγχει, αν το κελί έχει πάνω του πλοίο, ώστε να ενημερώσει πως το πλοίο χτυπήθηκε, και να κάνει τις αλλαγές στις τιμές στα αντίστοιχα στατιστικά των παιχτών. Η κλάση Board αρχικά μέσω του constructor ορίζει τα χαρακτηριστικά του κάθε ταμπλό, το οποίο αποτελείται από 100 κελιά από την προηγούμενη κλάση Cell. Στη συνέχεια έχουμε την μέθοδο placeShip η οποία

τοποθετεί τα πλοία στο ταμπλό του κάθε παίχτη. Μάλιστα στα ταμπλό του παίχτη (όχι του υπολογιστή) φαίνονται με ευδιάκριτα χρώματα. Η τοποθέτηση αυτή, γίνεται μόνο όταν επιτραπεί από την συνάρτηση `canPlaceShip` η οποία την ελέγχει. Συγκεκριμένα ανάλογα με τον κάθε περιορισμό, καλεί και την ανάλογη εξαίρεση, ενώ αν όλα είναι οκ, επιτρέπει την τοποθέτηση του πλοίου. Για να γίνουν αυτοί οι έλεγχοι καλούνται συγκεκριμένες μέθοδοι. Συγκεκριμένα για τον έλεγχο μην βγει το πλοίο εκτός ταμπλό, υλοποιούνται οι συναρτήσεις `validPoint` και `isCellValid`. Ακόμη για τον έλεγχο μην αλληλοκαλυφθούν πλοία, καλείται η συνάρτηση `isCellEmpty` και για να μην τοποθετηθούν τα πλοία εντελώς δίπλα, καλείται η `isShipNear`.

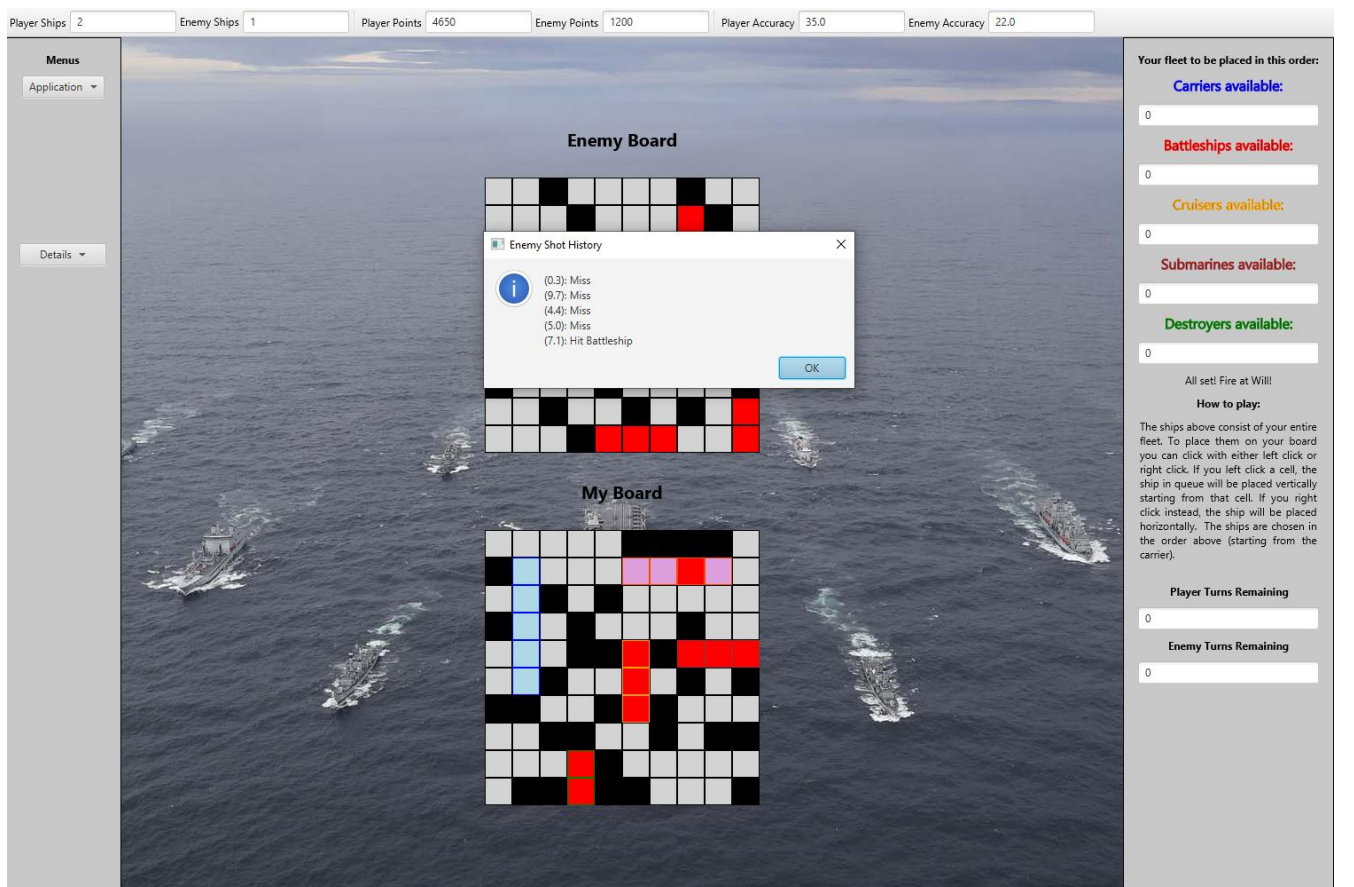
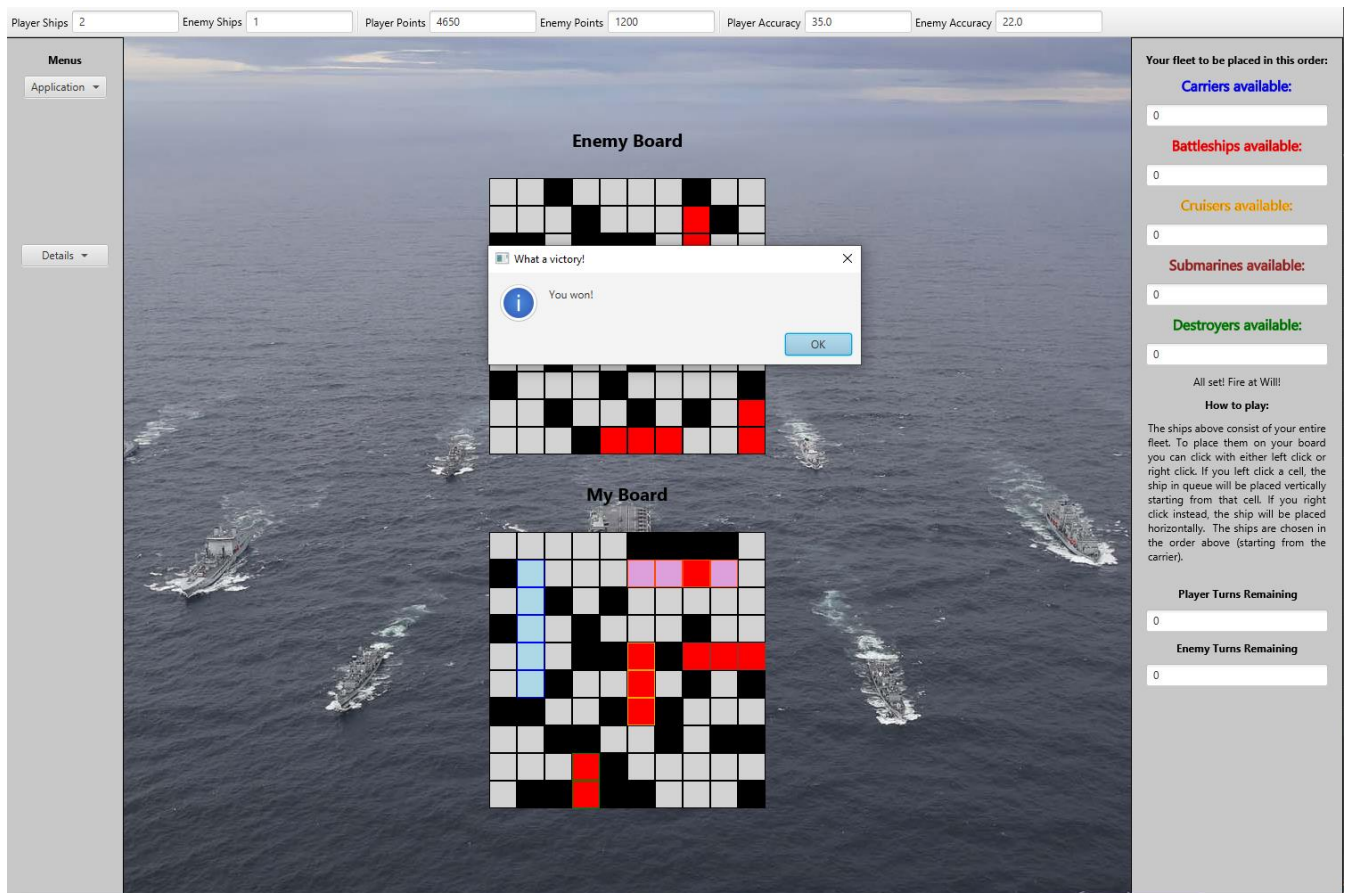
Η κάθε εξαίρεση έχει μια κλάση οι οποίες βρίσκονται σε δικό τους package, το `Exceptions` για την εκτύπωση του κατάλληλου μηνύματος. Συνεπώς θα βρείτε τις κλάσεις: `OversizeException`, `OverlapTilesException`, `AdjacentTilesException`, `InvalidCountException`.

Καθώς χρησιμοποιούμε JavaFX, προσπάθησα να κάνω χρήση του MVC προτύπου. Για αυτό στο package `Views`, βρίσκεται το `fxm1` αρχείο με την αρχική εικόνα για την έναρξη του παιχνιδιού. Όταν έχουμε κλικ στο κουμπί "PLAY" ο `GameBoardController` μας, στο package `Controllors`, εκκινεί το παιχνίδι. Καθώς το παιχνίδι απαιτεί δυναμική ανανέωση των στοιχείων της γραφικής διεπαφής, η συνάρτηση `createContent`, δημιουργεί όσα χρειαζόμαστε στο γραφικό μας περιβάλλον. Στη συνέχεια, η συνάρτηση `enemyMove` καθορίζει τον τρόπο με τον οποίο κινείται ο αντίπαλος και ενημερώνει τα κατάλληλα στοιχεία (ποσοστό εύστοχων βολών, πόντοι κλπ). Η ίδια η συνάρτηση, καλεί για την επιλογή στόχου την συνάρτηση `shootNear`, η οποία αποτελεί το AI του παιχνιδιού. Όπως ήδη αναφέρθηκε από τις παραδοχές, ο υπολογιστής χτυπάει τυχαία μέχρι να πετύχει κάποιο πλοίο. Στη συνέχεια, αρχίζει και χτυπάει τα διπλανά γειτονικά κελιά μέχρι να βρει τη συνέχεια του πλοίου. Όταν την βρει συνεχίζει προς αυτή την κατεύθυνση και αν δεν έχει βυθίσει το πλοίο, θα συνεχίσει από την ανάποδη. Επίσης υλοποιήθηκε η συνάρτηση `startGame` για την τοποθέτηση των πλοίων από τον υπολογιστή, αν δεν υπάρχει σενάριο. Ακόμη, όπως ήδη αναφέρθηκε, υπάρχουν δυο μέθοδοι, η `scenarioLoadPlayer` και η `enemyScenarioLoad` οι οποίες φορτώνουν τα πλοία των δύο παιχτών σύμφωνα με το ID που έδωσε ο χρήστης. Η τοποθέτηση των πλοίων και από τους δύο παίχτες καλεί την `InvalidCountException`, σε περίπτωση παραβίασης του αριθμού των πλοίων. Τέλος υπάρχουν και σημαντικές συναρτήσεις όπως εκείνη για την επιλογή του νικητή (`winConditionCheck`), εκείνη που φορτώνει την σκηνή του παιχνιδιού μετά το "PLAY" (`GameScene`), η συνάρτηση `infoBox` για τα Pop-Ups του παιχνιδιού καθώς και οι προφανείς `close()` και `restart()`.

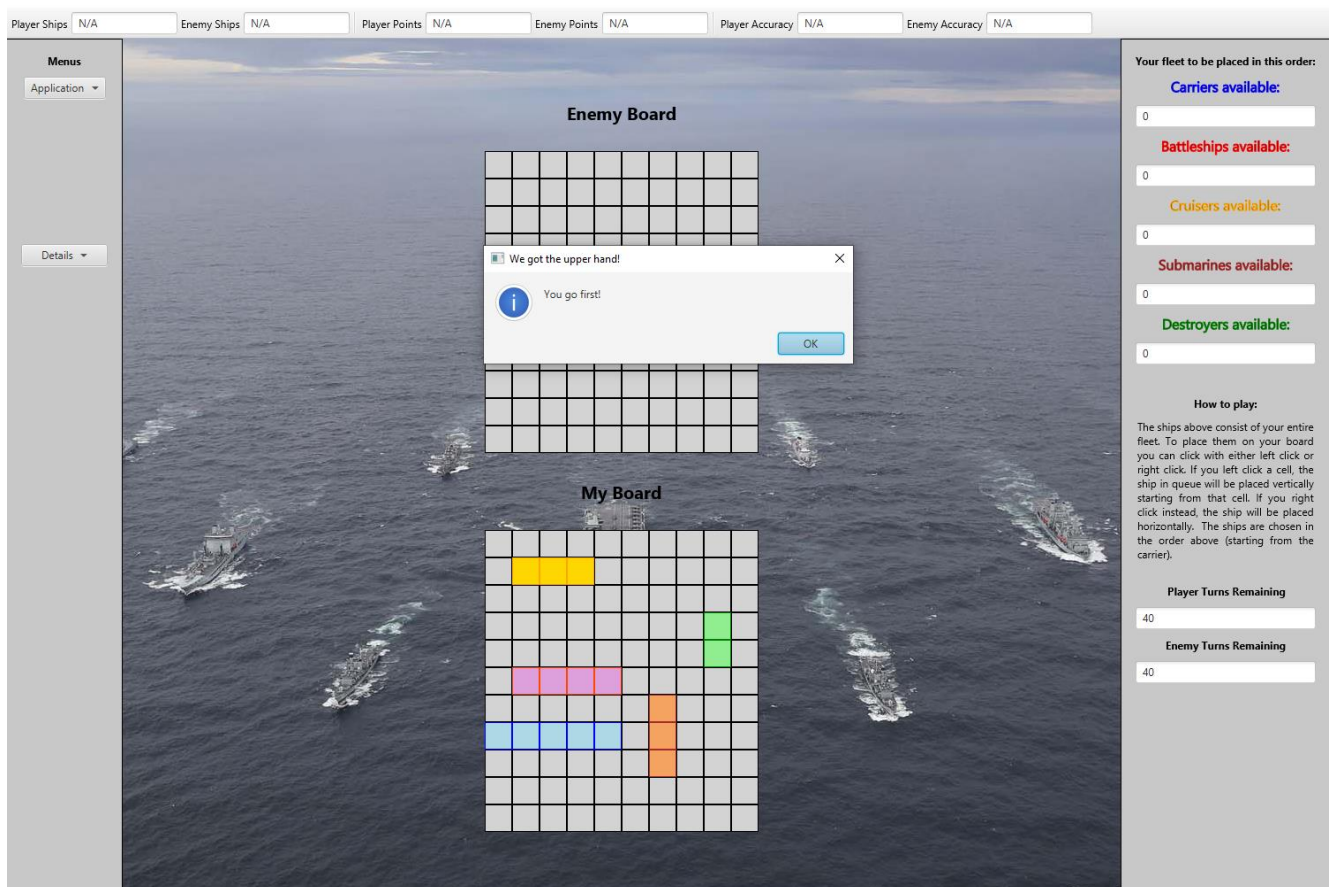
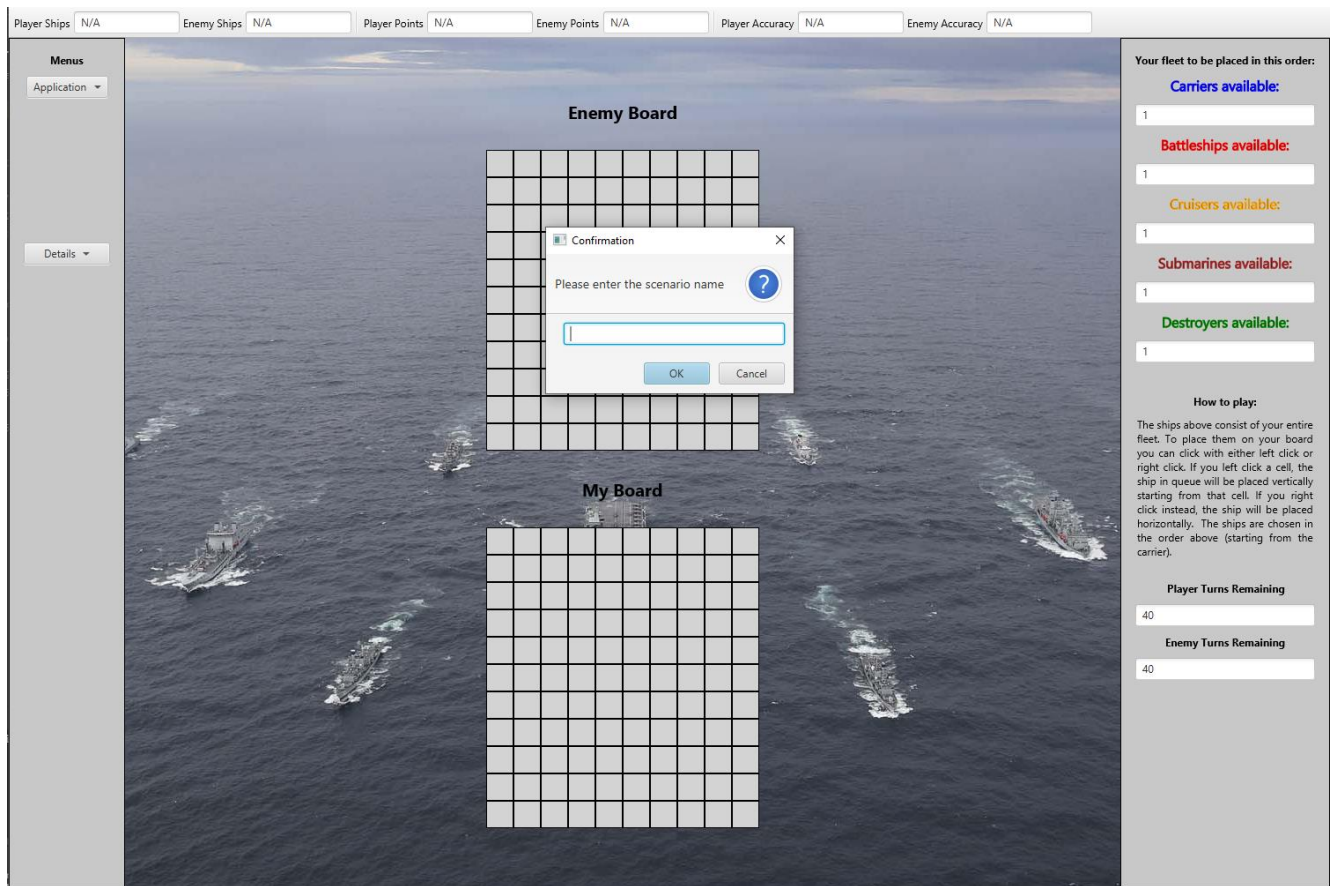
Τέλος θα βρείτε και ένα package `MediaLab` για τα αρχεία σεναρίων.

*Ακολουθούν λίγα Screenshots της εφαρμογής*









Menus

Application

Details

Enemy Board

Bad Luck!  
You lost!

My Board

Your fleet to be placed in this order:

Carriers available:  
0

Battleships available:  
0

Cruisers available:  
0

Submarines available:  
0

Destroyers available:  
0

All set! Fire at Will!

How to play:

The ships above consist of your entire fleet. To place them on your board you can click with either left click or right click. If you left click a cell, the ship in queue will be placed vertically starting from that cell. If you right click instead, the ship will be placed horizontally. The ships are chosen in the order above (starting from the carrier).

Player Turns Remaining  
0

Enemy Turns Remaining  
0