Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τελική εργασία στο μάθημα Τεχνολογία Πολυμέσων
2020-2021
Ανδρωνά Χριστίνα-Μαρία
Α.Μ.03116099

# Εφαρμογή υλοποιημένη σε Java -- Παιχνίδι Ναυμαχία

## Περιγραφή υλοποίησης

Για την εφαρμογή υλοποιήθηκαν 3 βασικές κλάσεις :

- **class Ship** --Σε αυτή την κλάση περιγράφονται τα χαρακτηριστικά του κάθε πλοίου (seats,points,bonus,row,column,orientation,health).Η κάθε υποκλάση θέτει στις μεταβλητές τις κατάλληλες τιμές ανάλογα με το είδος του πλοίου.
    - **class Battleship extends Ship**
    - **class Cruiser extends Ship**
    - **class Destroyer extends Ship**
    - **class Submarine extends Ship**

- **class Ocean** -- Σε αυτή την κλάση κατασκευάζουμε τον κάθε ωκεανό ,δηλαδή το ταμπλό του χρήστη και του αντιπάλου.Ο κάθε ωκεανός έχει τα δικά του χαρακτηριστικά (shotsFired,hitCount,shipSunk, shipsAlive,shipShot,points) .
    - Για την κατασκευή του κάθε ωκεανού απαιτείται η χρήση της **class Cell,**της οποίας κάθε αντικείμενο καθορίζει το κάθε κελί του ταμπλό, με το δικό του αριθμό γραμμής και στήλης. Στο κάθε κελί τοποθετείται και μια θέση του εκάστοτε πλοίου ( δηλαδή ένα αντικείμενο της αντίστοιχης κλάσης πλοίου) αν έχει οριστεί από τον χρήστη αλλιώς παραμένει ελεύθερο. Με τη μέθοδο ***public boolean shoot()*** της κλάσης Cell , το κάθε κελί δέχεται τη βολή από τον κάθε παίκτη.
- **class BattleShipGame** -- Η κύρια κλάση (περιέχει τη main μέθοδο) στην οποία “στήνεται” και πραγματοποιείται το παιχνίδι. Δημιουργούνται τα 2 ταμπλό του χρήστη και του αντιπάλου , τοποθετούνται τα πλοία στα αντίστοιχα κελιά με βάση την είσοδο,εκτελούνται οι βολές και παρουσιάζονται τα αποτελέσματα αυτών .

Για την τοποθέτηση πλοίων από το χρήστη:

- **class ReadInput** -- Με αυτή την κλάση λαμβάνεται η είσοδος που έδωσε ο χρήστης για την τοποθέτηση των πλοίων και στα 2 ταμπλό,μετατρέποντάς την στην κατάλληλη μορφή για αξιοποίηση.

  Περιορισμοί και Εξαιρέσεις που δημιουργήθηκαν:
    - **class InvalidCountException**-- η μέθοδος ***validate()*** της κλάσης ReadInput ελέγχει ότι έχουμε ακριβώς ένα πλοίο από το κάθε είδος.
    - **class OversizeException** -- η μέθοδος ***okToPlaceShipAt()*** της κλάσης Ship ελέγχει αν το κάθε πλοίο τοποθετήθηκε εντός των ορίων του ταμπλό.
    - **class AdjacentTilesException** -- η μέθοδος ***checkNeighbors()*** της κλάσης Ocean ελέγχει ότι κανένα πλοίο δεν εφάπτεται με άλλο .

- **class OverlapTilesException** -- η μέθοδος ***placeShipAt()*** της κλάσης Ocean ελέγχει ότι δεν υπάρχει επικάλυψη πλοίων κατά την τοποθέτηση , και αφού καλέσει και την *checkNeighbors()* τοποθετεί τα πλοία στα κελία.

  Η μέθοδος ***shipPlacement()*** της κλάσης Ocean ,ελέγχοντας όλα τα πιθανά παραπάνω exceptions κατασκευάζει και τοποθετεί μέσω της *placeShipAt()* τα πλοία στον ωκεανό.

Για να ξεκινήσει το παιχνίδι , μετά την επιτυχή τοποθέτηση πλοίων , η εφαρμογή επιλέγει τυχαία με τη μέθοδο ***startGame()*** της κλάσης BattleShipGame ποιος θα ξεκινήσει πρώτος, ενημερώνει με alertBox της **κλάσης AlertBox** τον χρήστη και το παιχνίδι ξεκινάει είτε με κίνηση του υπολογιστή είτε αναμένοντας την κίνηση του χρήστη, ανάλογα.

Για να παίξει ο υπολογιστής καλείται η μέθοδος **enemyMove()** της κλάσης BattleShipGame , όπου ο υπολογιστής παίζει “έξυπνα” όπως ο χρήστης, με διάφορους ελέγχους πριν την κάθε κίνηση του, εκτός αν αυτή πρέπει να είναι τυχαία. Μετά από κάθε κίνηση καλείται και μέθοδος ***addToHistory()*** για να παρουσιαζονται στο χ΄ρηστη οι 5 τελευταίες κινήσεις του αντιπάλου ή και οι δικές του.

Το παιχνίδι τερματίζει με τον έλεγχο αν συμπληρώθηκαν οι 40 βολές ή αν βούλιαξαν όλα τα πλοία. Τότε ανακοινώνεται στο χρήστη ο νικητής με alertBox της κλάσης AlertBox και καλείται η μέθοδος ***restart()*** ,η οποία αδειάζει το ταμπλό και αρχικοποιεί όλες τις μεταβλητές .

Για την γραφική διεπαφή χρήστη (GUI) έχουν δημιουργηθεί τα ακόλουθα:

- Με τη μέθοδο ***start()*** της κλάσης BattleShipGame δημιουργείται το κεντρικό παράθυρο. Σε αυτό τοποθετούνται οι 2 ωκεανοί (constructor της κλάσης Ocean) και οι συγκεντρωτικές πληροφορίες του κάθε παίκτη, οι οποίες ανανεώνονται μετά από κάθε βολή ή αρχικοποίηση ωκεανού.
- Ο χρήστης έχει 2 τρόπους για να πραγματοποιήσει τη βολή του στον ωκεανό του αντιπάλου, είτε πατώντας πάνω στο κελί στο οποίο θέλει να στοχεύσει (χρήση Event handler) είτε μέσω της διεπαφής όπου μπορεί να επιλέξει τη γραμμή και τη στήλη του κελιού που θέλει να στοχεύσει και να επιλέξει το κουμπί “shoot”(μέθοδος ***shootArea()***)
- Δημιουργείται ένα menubar με τη μέθοδο ***createMenu()*** με 2 menu Items :
  1) Application
  - Start: καλεί την ***startGame()*** για επιλογή σειράς και έναρξη παιχνιδιού.Το “start” δεν μπορεί να επιλεχθεί ούτε 2η φορά αφού ξεκινήσει το παιχνίδι ούτε αρχικά για να ξεκινήσει το παιχνίδι αν δεν έχει προηγηθεί πρώτα “load” αρχείων για τοποθέτηση πλοίων.
  - Load: εμφανίζει popupBox της **κλάσης PopupBox** για να εισάγει ο χρήστης την τοποθέτηση πλοίων των 2 ωκεανών, και να γίνει ο ελεγχος για την ορθότητα.
  - Exit: εμφανίζει confirmBox της **κλάσης ConfirmBox** για να επιβεβαιώσει ο χρήστης την επιθυμία του για έξοδο.

2) Details
   - Enemy Ships: εμφανίζει με alertBox την κατάσταση των πλοίων του αντιπάλου ελέγχοντας τις αντίστοιχες μεταβλητές του ωκεανού τπυ
   - Player Shots:Εμφανίζει με historyBox της **κλάσης HistoryBox** τις 5 τελευταίες κινήσεις του παίχτη, το αποτέλεσμα και αν ήταν επιτυχημένες και το πλοίο που πέτυχαν.
   - Enemy Shots:Εμφανίζει με historyBox της **κλάσης HistoryBox** τις 5 τελευταίες κινήσεις του αντιπάλου, ομοίως όπως του player.

Λοιπές απαιτήσεις

- Η υλοποίηση ακολουθεί τις αρχές σχεδίασης του αντικειμενοστραφούς προγραμματισμού (OOP design principles).
- Στην κλάση Ship έχει γίνει τεκμηρίωση των public κλάσεων και των μεθόδων με τις προδιαγραφές του εργαλείου javadoc.