ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ · ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

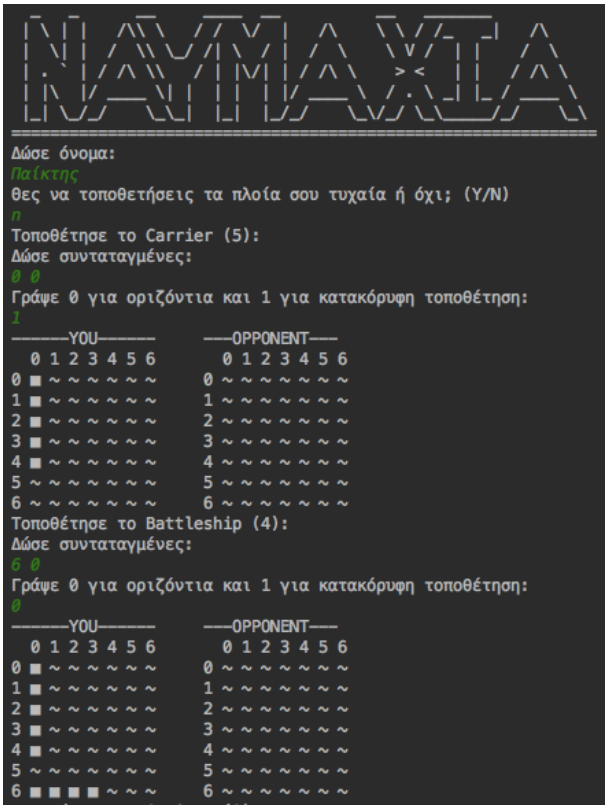
ΑΝΑΦΟΡΑ PROJECT – BATTLESHIP

ΟΝΤΟΚΕΝΤΡΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ - JAVA, C++

Καλούμαστε να υλοποιήσουμε το παιχνίδι «**Ναυμαχία**». Επιλέξαμε το παιχνίδι να τερματίζει όταν βυθιστούν όλα τα πλοία ενός από τους παίκτες. Υλοποιήσαμε τις παρακάτω κλάσεις:

ΚΛΑΣΗ GAME

Η βασική κλάση στην οποία υπάρχει η **`main`** μέθοδος και τρέχει το παιχνίδι. Περιλαμβάνει ως μεταβλητή (με default τιμή `7`) το μέγεθος του ταμπλό του παιχνιδιού (`sizeOfBoard`) ώστε να είναι εύκολα παραμετροποιήσιμος ο κώδικας. Η main μέθοδος μετά την εμφάνιση του λογοτύπου του παιχνιδιού (`printStart()`), ζητάει το όνομα του παίκτη, ο οποίος το εισάγει με την βοήθεια της `Scanner` και αποθηκεύεται στην `String` μεταβλητή `personName`. Δημιουργούνται δύο αντικείμενα τύπου `Player` για τον υπολογιστή και τον παίκτη (`person`, `computer`), όπως επίσης και δύο αντικείμενα τύπου `Board` όπου θα αποθηκεύσουν τα δύο ταμπλό του παιχνιδιού (`boardPerson`, `boardComputer`).



Στην αρχικοποίηση του ταμπλό του παίκτη, αυτός ερωτάται για το αν επιθυμεί τυχαία ή χειροκίνητη τοποθέτηση και απαντάει με το γράμμα `y` ή `n`. (Αποδεχόμαστε και πεζά και κεφαλαία γράμματα για να είναι πιο φιλικό προς το χρήστη). Στην περίπτωση που επιθυμεί τυχαία τοποθέτηση καλείται η μέθοδος `placeAllShips()` για την `boardPerson`. Στην χειροκίνητη τοποθέτηση καλείται η μέθοδος `placeShips()`. Υπάρχει αμυντικός προγραμματισμός ώστε να μην μπορεί ο παίκτης να δώσει οποιαδήποτε γράμμα πέρα από `y` και `n`. Σε αντίθετη περίπτωση επαναρωτάται.

Η αρχικοποίηση του ταμπλό του υπολογιστή γίνεται σε κάθε περίπτωση τυχαία με χρήση της `placeAllShips()` για την `boardComputer`. Το βασικό παιχνίδι τρέχει σε μία επανάληψη όπου στην αρχή ζητούνται οι συντεταγμένες (γραμμή, στήλη) με την βοήθεια της `getInput()` στις οποίες ο παίκτης επιθυμεί να χτυπήσει το ταμπλό του υπολογιστή. Στην συνέχεια ο υπολογιστής κάνει την δική του κίνηση με τυχαίες συντεταγμένες μέσω της `getRandInput()`. Στο τέλος εμφανίζονται τα δύο ταμπλό, και επαναλαμβάνεται η διαδικασία από την αρχή μέχρις ότου όλα τα πλοία από κάποιο ταμπλό βυθιστούν (ελέγχονται μέσω της `allShipsSunk()` και για το `boardPerson` και για το `boardComputer`).

Στο τέλος του παιχνιδιού τυπώνονται τα στατιστικά του παίκτη και του υπολογιστή, ascii art πυροτεχνήματα (`printEnd()`) και το όνομα του νικητή.

Πέρα από την main, η Game περιλαμβάνει την **`getInput()`** στην οποία ζητούνται από τον χρήστη δύο ακέραιες τιμές – συντεταγμένες (γραμμή και στήλη), οι οποίες επιστρέφονται σε έναν μονοδιάστατο πίνακα δύο κελιών. Υπάρχει αμυντικός προγραμματισμός ώστε οι τιμές που δίνει ο χρήστης να μην είναι αρνητικές και να μην υπερβαίνουν το μέγεθος του ταμπλό (`0` ως `sizeOfboard-1`). Σε περίπτωση που ο χρήστης δώσει

παραπάνω φορές μη αποδεκτές τιμές τυπώνεται πρόσθετο μήνυμα που εξηγεί στον χρήστη τι τιμές να δώσει, κάνοντας το περιβάλλον πιο φιλικό.

Επίσης περιλαμβάνεται η **`getRandInput()`** που και αυτή επιστρέφει έναν μονοδιάστατο πίνακα με δύο τιμές, οι οποίες όμως αυτή τη φορά δημιουργούνται τυχαία μέσω της κλάσης Random και που φυσικά είναι προσαρμοσμένες στο διάστημα (`0` μέχρι `sizeOfBoard-1`).

Επίπλέον περιλαμβάνεται η μέθοδος **`getOrientation()`** που καλεί τον χρήστη να δώσει την τιμή `1` για κατακόρυφη τοποθέτηση του πλοίου ή την τιμή `0` για οριζόντια τοποθέτηση, την οποία και επιστρέφει. Υπάρχει πάλι αμυντικός προγραμματισμός ώστε να μην δέχεται οποιαδήποτε άλλη τιμή πέρα από αυτές.

ΚΛΑΣΗ TILE

Η `Tile` είναι η κλάση από την οποία θα δημιουργηθούν όλα τα κελιά του ταμπλό. Περιλαμβάνει ακέραιες μεταβλητές x και y για τις συντεταγμένες του ταμπλό όπως επίσης και μια μεταβλητή type τύπου `typeList` (ο οποίος είναι τύπου enumerate με τις επιλογές `Sea`, `Ship`, `Hit`, `Miss`) η οποία θα προσδιορίζει την κατάσταση του κελιού κάποια χρονική στιγμή.

Υπάρχει ένας constructor με ορίσματα τις συντεταγμένες, κάποιες get και set μέθοδοι και μια μέθοδος **`draw()`** που επιστρέφει String ανάλογα με τον τύπου του κελιού και δέχεται όρισμα για την απόκρυψη της περίπτωσης του `Ship`. Η `draw()` ουσιαστικά σχεδιάζει το κελί και θα χρησιμεύσει στην `drawboards()` αργότερα.

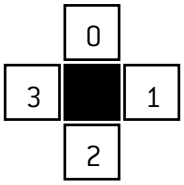
ΚΛΑΣΗ BOARD

Η Board είναι ουσιαστικά το ταμπλό του παιχνιδιού. Υπάρχει ένας δισδιάστατος πίνακας τύπου `Tile` με μέγεθος το `sizeOfBoard` που ορίσαμε στην `Game`. Στην δημιουργία κάθε αντικειμένου ο constructor αρχικοποιεί όλα τα κελιά τύπου `Tile` και θέτει το τύπο τους σε `Sea`. Υπάρχει η μέθοδος **`drawboards()`** που σχεδιάζει όλο το γραφικό περιβάλλον του παιχνιδιού με τα δύο ταμπλό του παίκτη και του υπολογιστή. Κάθε κελί είναι αποτέλεσμα της κλήσης της `draw()` και στον σχεδιασμό του ταμπλό του αντιπάλου αποκρύπτονται τα πλοία με κατάλληλη Boolean μεταβλητή ως όρισμα.

Περιλαμβάνεται η **`placeAllShips()`** στην οποία δημιουργούνται 5 διαφορετικά πλοία και γίνεται προσπάθεια τυχαίας τοποθέτησής τους στο ταμπλό με την προϋπόθεση να μην βγαίνουν εκτός όριων, να μην χτυπάει το ένα πάνω στο άλλο και να μην εφάπτονται. Για να επιτευχθεί αυτό γίνονται προσπάθειες σε μια επανάληψη που περιέχει switch και έναν μετρητή ο οποίος προσδιορίζει ποιο πλοίο θα τοποθετηθεί. Ο μετρητής αυξάνεται κάθε φορά που τοποθετείται επιτυχημένα κάποιο πλοίο και η επανάληψη σταματάει όταν τοποθετηθούν όλα. Τα πλοία τοποθετούνται σε συγκεκριμένη σειρά και με μειούμενο μέγεθος ώστε να χρειαστούν λιγότερες προσπάθειες συνολικά για να τοποθετηθούν τυχαία. Για την τοποθέτηση κάθε πλοίου καλείται η `placeShip()` της κλάσης `Ship` με τυχαίο αρχικό κελί τοποθέτησης (στο οποίο τοποθετούνται τυχαίες συντεταγμένες μέσω της `getRandInput()` της `Game` *(επαναχρησιμοποίηση κώδικα)* και τυχαίο προσανατολισμό με την βοήθεια της `Random`.

Επίσης περιλαμβάνεται η **`placeShips()`** που έχει παρόμοια λογική με την `placeAllShips()` με την διαφορά πως η τοποθέτηση δεν γίνεται τυχαία. Πάλι δημιουργούνται τα 5 διαφορετικά πλοία και υπάρχει ένας μετρητής που καλεί τον χρήστη να τοποθετήσει το καθένα. Ζητούνται οι συντεταγμένες μέσω της `getInput()` *(επαναχρησιμοποίηση κώδικα)* του αρχικού κελιού και καλείται πάλι η `placeShip` για κάθε πλοίο. Ως όρισμα για τον προσανατολισμό έχει την μέθοδο `getOrientation()` της `Game` που όπως αναφέρθηκε ζητά από την χρήστη τον προσανατολισμό και τον επιστρέφει. Στην `placeShips()` τυπώνονται μηνύματα ανάλογα με το αν μπορεί να τοποθετηθεί το πλοίο, και ποιο πρόβλημα (exception) προκύπτει, εν αντιθέσει με την `placeAllShips()` όπου αποκρύπτονται. Σε κάθε τοποθέτηση καλείται η `drawboards()` που τυπώνει τα ταμπλό ώστε να έχει ο χρήστης οπτικοποιημένες τις ενέργειές του.

Επιπλέον περιλαμβάνεται η **getAdjacentTiles()** η οποία επιστρέφει έναν πίνακα τύπου Tile με τα τέσσερα γειτονικά κελιά του κελιού που δίνεται ως όρισμα με την σειρά που φαίνεται δεξιά. Προβλέπονται όλες οι περιπτώσεις στις οποίες τυγχάνει κάποιο κελί να βρίσκεται εκτός ορίων του πίνακα, στις οποίες αντικαθίσταται με ένα εικονικό `outofBoard` κελί με τύπο Sea για το σωστό έλεγχο των εξαιρέσεων στην τυχαία τοποθέτηση των πλοίων.

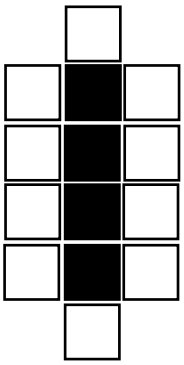


Τέλος περιλαμβάνεται η **allShipsSunk()** που επιστρέφει true ή false ανάλογα με το αν υπάρχει έστω και ένα κελί `Tile` τύπου `Ship` στο ταμπλό και η `getBoard()` που επιστρέφει το πίνακα `Tile[][]` για εσωτερική χρήση στο κώδικα.

ΚΛΑΣΗ SHIP

Η υπερκλάση όλων των πλοίων. Περιλαμβάνει τις μεταβλητές `bTile` (αρχικό κελί τοποθέτησης), τον προσανατολισμό (`orientation`) και το μέγεθος του πλοίου (`shipSize`).

Περιλαμβάνονται 3 μέθοδοι: `Oversize()`, `Overlap()` και `AdjacentTiles()` οι οποίες ελέγχουν αν το πλοίο που πρόκειται να τοποθετηθεί πληροί τις προϋποθέσεις να είναι εντός ορίων, να μην χτυπάει σε άλλο και να μην εφάπτεται σε άλλο. Σε αντίθετη περίπτωση οι μέθοδοι πετάνε exceptions. Η **Oversize()** ανάλογα με τον προσανατολισμό του πλοίου ελέγχει αν οι συντεταγμένες του αρχικού κελιού τοποθέτησης μαζί με το μέγεθος του πλοίου παραμένουν μέσα στο μέγεθος του ταμπλό. Ουσιαστικά ελέγχει το τελευταίο κελί κάθε πλοίου, γιατί αν αυτό είναι εντός ταμπλό, θα είναι και τα υπόλοιπα. Δεν έχει σημασία να κοιτάξει η `Oversize()` αν το αρχικό κελί τοποθέτησης είναι εντός ταμπλό, αφού ελέγχεται από την `getInput()`. Η **Overlap()** ελέγχει αν κάθε δυνητικό κελί τοποθέτησης του πλοίου είναι τύπου `Ship` ή όχι. Η **AdjacentTiles()** με την βοήθεια της `getAdjacentTiles()` ελέγχει τα γειτονικά κελιά τα οποία φαίνονται στο διπλανό σχήμα, σκανάροντας όλα τα κελιά που δυνητικά θα τοποθετούνταν το πλοίο. Χρησιμοποιεί στοχευμένα όποια κελιά χρειάζεται από τον πίνακα που επιστρέφει η `getadjacentTiles()`, γιατί παραδείγματος χάρη στο αρχικό κελί κατακόρυφης τοποθέτησης δεν έχει νόημα ο έλεγχος του από κάτω κελιού, αφού δεν είναι γειτονικό αλλά κελί που θα μπει το ίδιο το πλοίο και συν τοις άλλοις έχει ήδη ελεγχθεί από την `Overlap()`.



Περιλαμβάνεται επίσης η **placeShip()**, η μέθοδος δηλαδή που τοποθετεί το πλοίο. Πριν κάνει την τοποθέτηση, την αλλαγή του τύπου των κελιών δηλαδή από `Sea` σε `Ship`, καλεί τις τρεις προαναφερθείσες μεθόδους με την λογική σειρά που αναφέρθηκαν προηγουμένως και σε περίπτωση που δεν γίνεται να γίνει η τοποθέτηση δεν θα γίνει η τοποθέτηση διότι θα πετάξουν exception το οποίο η μέθοδος θα κάνει catch τυπώνοντας κατάλληλο μήνυμα. Η τύπωση γίνεται με όρισμα μια boolean μεταβλητή σε περίπτωση που δεν θέλουμε να τυπωθούν τα μηνύματα (στην περίπτωση των επαναμβανόμενων προσπάθειων της `placeAllShips()` πχ). Σε διαφορετική περίπτωση θα τοποθετήσει κανονικά το πλοίο, επιστρέφοντας την τιμή true η οποία θα αυξήσει τον μετρητή στις `place(All)Ships()`.

ΚΛΑΣΗ PLAYER

Η κλάση από τη οποία δημιουργούνται οι δύο παίκτες. Περιλαμβάνει το όνομα του παίκτη και ακέραιες μεταβλητές που αποθηκεύουν τα στατιστικά (`fires` [= σύνολο βολών], `misses` [=αποτυχίες], `succededFires` [=επιτυχημένες βολές] και `alreadyFired` [=επαναμβανόμενες βολές]).

Περιέχει την μέθοδο **fire()** που παίρνει ως όρισμα το `board` στο οποίο εκτελείται η κάθε βολή και τις συνταγμένες της βολής. Η μέθοδος σε όλες τις περιπτώσεις αυξάνει την μεταβλητή `fires` και επιπλέον ανάλογα με το τύπου του κελιού στο οποίο γίνεται η βολή αυξάνει την μεταβλητή που αντιστοιχεί σε κάθε περίπτωση. Επίσης τυπώνει μήνυμα με την ενέργεια που έκανε ο παίκτης.

Περιλαμβάνεται επίσης η **getStats()** μέθοδος που τυπώνει τις προαναφερθείσες μεταβλητές ως τα στατιστικά του παίκτη, η οποία καλείται από την `Game` στο τέλος του παιχνιδιού.

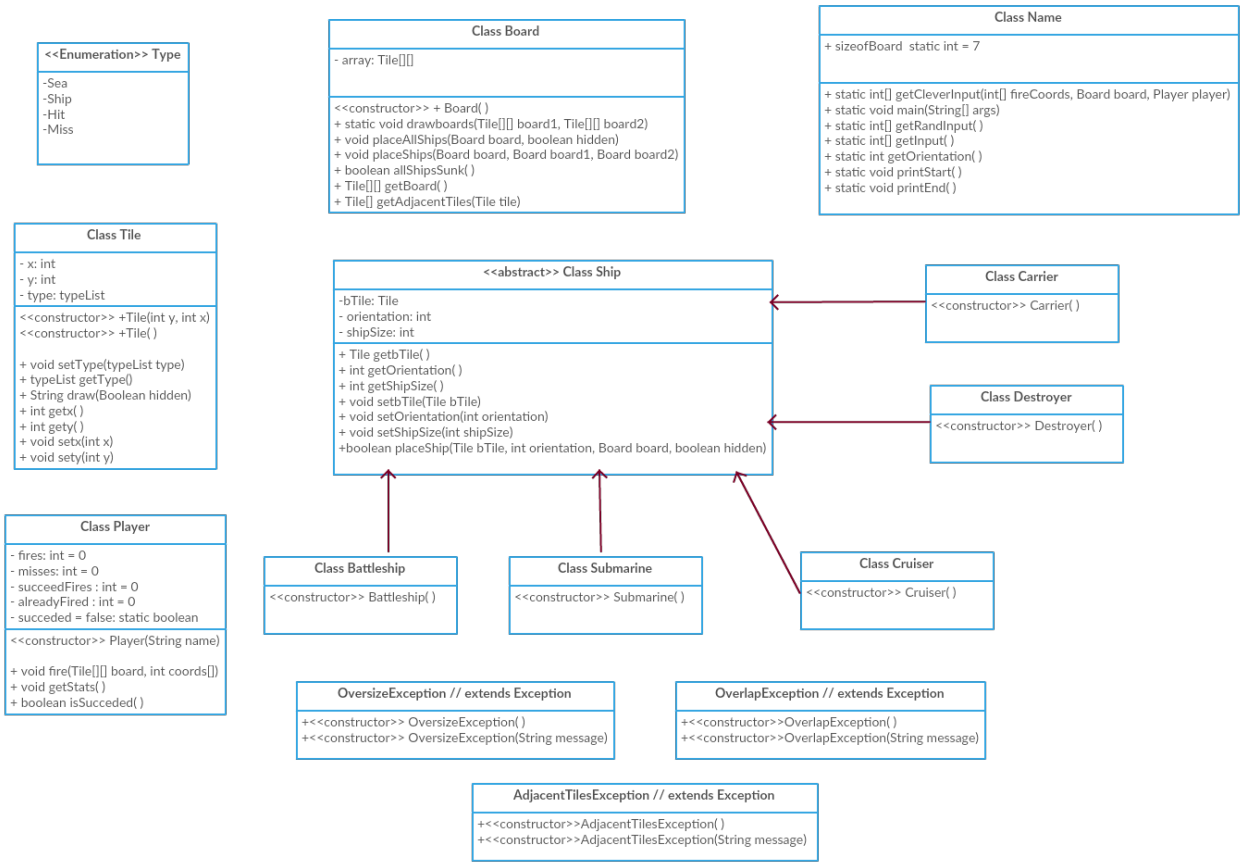
ΚΛΑΣΗ BATTLESHIP, CARRIER, CRUISER, DESTROYER, SUBMARINE

Υποκλάσεις της `Ship`. Η καθεμία περιλαμβάνει έναν constructor με την μέθοδο `setShipSize` με όρισμα ανάλογο του κάθε πλοίου.

ΚΛΑΣΗ OVERLAPEXCEPTION, OVERSIZEEXCEPTION, ADJACENTTILESEXCEPTION

Υποκλάσεις της κλάσης `Exception` της java.

Διάγραμμα κλάσεων:



ΣΤΟΙΧΕΙΑ ΟΜΑΔΑΣ		
Αλέξανδρος Ξιάρχος	1059619	up1059619@upnet.gr
Μαρία-Αναστασία Κυριακάτου	1059656	up1059656@upnet.gr