Πρώτη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: 24 Απριλίου 2018, 12 μ.μ.

Στην άσκηση αυτή θα υλοποιήσετε σε Java ένα πρόγραμμα που θα υλοποιεί το παιχνίδι ναυμαχίας (Battleship). Η άσκηση πρέπει να παραδοθεί μέχρι τις 24/4/2018 πριν το μάθημα.

Η άσκηση έχει τρία κομμάτια. Στο πρώτο θα υλοποιήσετε ένα παιχνίδι μεταξύ δύο χρηστών. Στο δεύτερο ένα παιχνίδι μεταξύ ενός χρήστη και του υπολογιστή που παίζει τυχαία. Στο τρίτο ο υπολογιστής θα υλοποιήσει μια πιο έξυπνη στρατηγική όταν χτυπήσει κάποιο πλοίο. Το κάθε κομμάτι είναι υπερσύνολο του προηγούμενου: θα πρέπει να προσθέσετε επιπλέον πεδία και μεθόδους στις κλάσεις σας, και να κάνετε επιπλέον κλάσεις. Αν έχετε ολοκληρώσετε επιτυχώς ένα κομμάτι δεν χρειάζεται να παραδώσετε τα προηγούμενα. Μπορείτε όμως να βαθμολογηθείτε μόνο για τα κομμάτια που έχετε υλοποιήσει αν δεν έχετε καταφέρει να ολοκληρώσετε επιτυχώς τα επόμενα.

Το πρώτο κομμάτι το κάνετε turnin στο assignment1.1@myy205 το δεύτερο στο assignment1.2@myy205 και το τρίτο στο assignment1.3@myy205

Περιγραφή παιχνιδιού
Το παιχνίδι παίζεται με δύο παίκτες. Ο κάθε παίχτης έχει ένα πίνακα 10×10 θέσεων στον οποίο τοποθετεί πέντε πλοία: Ένα μεγέθους 5 θέσεων, ένα μεγέθους 4 θέσεων, δύο μεγέθους 3 θέσεων και ένα μεγέθους 2 θέσεων. Οι παίχτες εναλλάσσονται χτυπώντας θέσεις του αντιπάλου, με σκοπό να βυθίσουν τα πλοία του. Για κάθε βολή, ενημερώνονται αν ήταν επιτυχής (hit) ή όχι (miss). Επίσης ενημερώνονται αν βυθίστηκε κάποιο πλοίο. Ο κάθε παίχτης κρατάει επίσης ένα πίνακα 10×10 θέσεων στον οποίο σημειώνει τις βολές που έχει κάνει και ποιες ήταν επιτυχείς, τον οποίο χρησιμοποιεί για να χαράσσει την στρατηγική του. Κερδίζει ο παίχτης που πρώτος θα βυθίσει όλα τα πλοία του αντιπάλου.

Μέρος 1ο
Για την υλοποίηση σας θα πρέπει να δημιουργήσετε τέσσερεις κλάσεις. Την κλάση **ShipBoard** η οποία κρατάει πληροφορίες για τον πίνακα με τα πλοία του κάθε παίχτη. Την κλάση **StrikeBoard** η οποία κρατάει πληροφορίες για τον πίνακα με τις βολές του κάθε παίχτη. Την κλάση **HumanPlayer** που υλοποιεί το παιχνίδι ενός παίκτη-ανθρώπου. Την κλάση **BattleShip** που υλοποιεί την ροή του παιχνιδιού.

ShipBoard: Η κλάση αυτή θα υλοποιεί τον πίνακα (board) με τα πλοία ενός παίχτη. Το κάθε πλοίο έχει ένα αύξοντα αριθμό (id) από 1 έως 5. Η κλάση θα έχει τα εξής πεδία: (α) Ένα πίνακα με τα μεγέθη των πλοίων. (β) Ένα δισδιάστατο πίνακα ακεραίων 10×10 ο οποίος θα κρατάει τα πλοία του παίχτη. Ο πίνακας θα έχει τιμή 0 σε μία θέση αν δεν έχει τοποθετηθεί ένα πλοίο εκεί, και το id του πλοίου αν υπάρχει πλοίο που καταλαμβάνει αυτή τη θέση. (γ) Ένα δισδιάστατο boolean πίνακα 10×10 ο οποίος θα καταγράφει ποιες θέσεις του πίνακα έχουν χτυπηθεί. (δ) Μια boolean μεταβλητή που θα κρατάει αν το τελευταίο χτύπημα βύθισε το πλοίο.

Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Την μέθοδο **enterShipManually** η οποία παίρνει σαν όρισμα το id του πλοίου και ζητάει από τον χρήστη να δώσει την αρχική θέση και την κατεύθυνση στην οποία θα τοποθετηθεί το πλοίο (οριζόντια/κάθετα). Αν είναι επιτυχής η τοποθέτηση επιστρέφει, αλλιώς ξαναζητάει από τον χρήστη την θέση και την κατεύθυνση. Η μέθοδος αυτή θα οριστεί private.
- Την μέθοδο **enterAllShipsManually** η οποία καλεί την enterShipManually για όλα τα πλοία.
- Μία μέθοδο **getStrike** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους) και προσθέτει ένα χτύπημα, εφόσον δεν έχει ήδη χτυπηθεί η θέση. Επιστρέφει μια boolean τιμή αν η βολή

χτύπησε πλοίο ή όχι. Αν η βολή χτύπησε πλοίο ελέγχει αν όλες οι θέσεις του πλοίου έχουν χτυπηθεί. Στην περίπτωση αυτή εκτυπώνει και ενημερώνει και το αντίστοιχο boolean πεδίο της κλάσης. (Υπόδειξη: Μπορείτε να χρησιμοποιήσετε τον πίνακα με τα μεγέθη των πλοίων, ή/και ένα βοηθητικό πίνακα, για να κρατάτε λογαριασμό πόσες θέσεις από κάθε πλοίο δεν έχουν χτυπηθεί).

- Μια μέθοδο **allShipsSank** που μας επιστρέφει μια boolean τιμή αν όλα τα πλοία στον πίνακα έχουν βυθιστεί.
- Μία μέθοδο **lastStrikeSankShip** που επιστρέφει μια boolean τιμή true αν το τελευταίο χτύπημα βύθισε ένα πλοίο και false διαφορετικά.
- Τη μέθοδο **printBoard** η οποία εκτυπώνει τον πίνακα.
- Μια μέθοδο **main** για να τεστάρει την κλάση σας. Η main θα δημιουργήσει ένα ShipBoard αντικείμενο και θα τοποθετήσει όλα τα πλοία. Καλέσετε την getStrike σε θέση με πλοίο, χωρίς πλοίο, θέση που έχει ήδη χτυπηθεί. Τεστάρετε και την lastStrikeSankShip και τις άλλες μεθόδους. Η κλάση αυτή θα βαθμολογηθεί τρέχοντας την main και θα κριθείτε και από τα τεστ που επιλέξατε να κάνετε.

StrikeBoard: Η κλάση αυτή θα υλοποιεί τον πίνακα με τα χτυπήματα ενός παίχτη. Θα κρατάει ένα πίνακα 10×10 από ακεραίους ο οποίος θα έχει τιμή μηδέν για κάθε θέση που δεν έχει χτυπηθεί ακόμη, τιμή 1 αν σε εκείνη τη θέση έγινε επιτυχημένο χτύπημα (hit), και -1 αν έγινε αποτυχημένο χτύπημα (miss). Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Την μέθοδο **addStrike** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους), και μια boolean τιμή που μας λέει αν το χτύπημα ήταν επιτυχές, και ενημερώνει τον πίνακα κατάλληλα.
- Τη μέθοδο **printBoard** η οποία εκτυπώνει τον πίνακα.

HumanPlayer: Η κλάση αυτή θα υλοποιεί το παιχνίδι ενός χρήστη. Ο κάθε παίχτης θα έχει ένα όνομα που αρχικοποιείται στον constructor. Θα έχει επίσης ένα ShipBoard και ένα StrikeBoard. Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Τον constructor όπου αρχικοποιείται το όνομα και ο παίχτης τοποθετεί τα πλοία του.
- Την μέθοδο **nextStrike** η οποία ζητάει από τον χρήστη να μας δώσει την επόμενη θέση που θέλει να χτυπήσει και μας επιστρέφει την θέση αυτή (ένα δισδιάστατο πίνακα από ακέραιους).
- Την μέθοδο **update** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους), και μια boolean τιμή που μας λέει αν το χτύπημα σε αυτή την θέση ήταν επιτυχές, και ενημερώνει την StikeBoard.
- Την μέθοδο **getStrike** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους) στην οποία ο παίχτης δέχεται ένα χτύπημα. Επιστρέφει μια boolean τιμή αν η βολή χτύπησε πλοίο ή όχι.
- Την μέθοδο **allShipsSank** που μας επιστρέφει μια boolean τιμή αν όλα τα πλοία του παίχτη έχουν βυθιστεί.
- Μία μέθοδο **lastStrikeSankShip** που επιστρέφει μια boolean τιμή true αν το τελευταίο χτύπημα που δέχτηκε ο παίχτης βύθισε πλοίο.
- Τη μέθοδο **toString** η οποία επιστρέφει το όνομα του παίχτη.

BattleShip: Η κλάση αυτή θα υλοποιεί το παιχνίδι μεταξύ δύο χρηστών. Η κλάση θα έχει μια βοηθητική private μέθοδο **HumanVsHuman** η οποία θα δημιουργεί δυο παίχτες με δύο ονόματα που θα διαλέξετε εσείς και θα τους βάλει να παίξουν μεταξύ τους. Σε κάθε γύρο ο κάθε παίχτης διαλέγει μια θέση, χτυπάει τον αντίπαλο, παίρνει την απάντηση και ενημερώνει το StrikeBoard του. Πριν και μετά από την κίνηση ενός παίχτη εκτυπώνεται το StrikeBoard του (μπορεί να χρειαστεί επιπλέον μέθοδος στην Player για την εκτύπωση). Αν η τελευταία βολή βύθισε το πλοίο του αντιπάλου, εκτυπώνεται ένα μήνυμα. Αν όλα τα πλοία ενός παίχτη βυθιστούν, ο άλλος παίχτης ανακηρύσσεται νικητής και το παιχνίδι σταματάει. Η μέθοδος **main** απλά καλεί την HumanVsHuman.

Μέρος 2ο

Στο δεύτερο κομμάτι της άσκησης θα υλοποιήσετε το παιχνίδι μεταξύ χρήστη και υπολογιστή. Για την υλοποίηση σας θα πρέπει να τροποποιήσετε τις κλάσεις ShipBoard, StrikeBoard και BattleShip που δημιουργήσατε και θα δημιουργήσετε δύο νέες κλάσεις: Την κλάση **RandomStrategy** η οποία η οποία υλοποιεί τις τυχαίες επιλογές του παίχτη-υπολογιστή και την κλάση **ComputerPlayer** που υλοποιεί ο παιχνίδι ενός παίχτη-υπολογιστή.

ShipBoard: Για την κλάση αυτή χρειαζόμαστε τις εξής επιπλέον μεθόδους:

- Την μέθοδο **enterShipRandomly** η οποία παίρνει σαν όρισμα το id του πλοίου και το τοποθετεί σε μια τυχαία θέση στον πίνακα. Ο αλγόριθμος τοποθέτησης δουλεύει ως εξής: Πρώτα επιλέγεται τυχαία η κατεύθυνση στην οποία θα μπει το πλοίο (οριζόντια ή κάθετα). Στην συνέχεια επιλέγεται τυχαία η αρχική θέση (δηλαδή, ένα ζευγάρι (x,y)) και το πλοίο τοποθετείται είτε από αριστερά προς τα δεξιά αν οριζόντια, είτε από κάτω προς τα πάνω αν κάθετα, εφόσον οι θέσεις είναι κενές. Η επιλογή των αρχικών τιμών των x,y θα πρέπει να λαμβάνει υπόψιν τον μήκος του πλοίου (π.χ., αν τοποθετούμε ένα πλοίο μεγέθους 5 θέσεων κάθετα, η αρχική τιμή του y θα πρέπει να είναι το πολύ 5). Αν η τοποθέτηση δεν είναι επιτυχής, τότε επιλέγεται μια νέα θέση (x,y) και γίνεται εκ νέου προσπάθεια. Η μέθοδος αυτή θα οριστεί private.
- Την μέθοδο **enterAllShipsRandomly** η οποία καλεί την enterShipRandomly για όλα τα πλοία.
- Τροποποιήστε την μέθοδο **main** που δημιουργήσατε στο Μέρος 1 ώστε να δημιουργεί ένα επιπλέον ShipBoard αντικείμενο όπου τα πλοία θα τοποθετούνται τυχαία.

RandomStrategy: Η κλάση αυτή θα υλοποιεί την απλή στρατηγική του παίχτη-υπολογιστή όπου επιλέγει τυχαία την επόμενη θέση που θα χτυπήσει. Η κλάση χρειάζεται κάποια δομή που θα χρησιμοποιεί για να επιλέξει τυχαία την επόμενη διαθέσιμη θέση (π.χ., ένα ArrayList με τις διαθέσιμες θέσεις από τις οποίες θα επιλέγετε μια τυχαία, ή ένα πίνακα με τυχαίες τιμές στο [0,1] από τον οποίο θα διαλέγεται τη θέση με την μεγαλύτερη τιμή). Θα πρέπει να μπορείτε να διαλέξετε μια τυχαία θέση, καθώς και να αφαιρέσετε μία θέση από τις υποψήφιες θέσεις.

Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Την μέθοδο **nextStrike** η οποία επιστρέφει μια τυχαία θέση από αυτές που δεν έχει ήδη χτυπήσει ο παίχτης-υπολογιστής.
- Την μέθοδο **update** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους) και ενημερώνει τις βοηθητικές δομές κατάλληλα ώστε η θέση να μην είναι πλέον υποψήφια.

ComputerPlayer: Η κλάση αυτή θα υλοποιεί το παιχνίδι του παίχτη-υπολογιστή. Θα έχει πεδία παρόμοια με αυτά του HumanPlayer, δηλαδή, ένα ShipBoard, ένα StrikeBoard, και ένα String name το οποίο θα έχει τιμή “HAL”. Επίσης θα έχει και ένα πεδίο RandomStrategy. Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Τον constructor όπου τοποθετούνται τα πλοία του παίχτη-υπολογιστή σε τυχαίες θέσεις.
- Την μέθοδο **nextStrike** η οποία επιστρέφει την επόμενη θέση (ένα δισδιάστατο πίνακα από ακέραιους) που θα χτυπήσει ο παίχτης υπολογιστής.
- Την μέθοδο **update** η οποία παίρνει σαν όρισμα μια θέση και μια boolean τιμή που μας λέει αν το χτύπημα στον αντίπαλο σε αυτή την θέση ήταν επιτυχές, και ενημερώνει την StikeBoard και την RandomStrategy.
- Την μέθοδο **getStrike** η οποία παίρνει σαν όρισμα μια θέση στην οποία ο παίχτης δέχεται ένα χτύπημα, και επιστρέφει μια boolean τιμή αν η βολή χτύπησε πλοίο ή όχι.
- Την μέθοδο **allShipsSank** που μας επιστρέφει μια boolean τιμή αν όλα τα πλοία του παίχτη έχουν βυθιστεί.
- Τη μέθοδο **toString** η οποία επιστρέφει το όνομα του παίχτη.

Μπορεί να χρειαστείτε και μεθόδους για την εκτύπωση των StrikeBoard ή/και ShipBoard αν το κρίνετε απαραίτητο.

BattleShip: Προσθέστε μια βοηθητική private μέθοδο **HumanVsComputer** η οποία θα δημιουργεί ένα παίχτη-χρήστη και ένα παίχτη-υπολογιστή, και θα υλοποιεί το παιχνίδι μεταξύ τους. Η λογική του παιχνιδιού είναι ακριβώς ίδια με πριν, απλά ο ένας παίχτης πλέον είναι ο υπολογιστής. Η μέθοδος **main** ρωτάει αν θέλουμε να παίξουμε με άλλο χρήστη ή με τον υπολογιστή και ανάλογα καλεί την HumanVsHuman ή την HumanVsComputer.

Μέρος 3ο

Στο τρίτο κομμάτι της άσκησης θα υλοποιήσετε μια πιο έξυπνη στρατηγική για τον υπολογιστή. Για την υλοποίηση σας θα πρέπει να τροποποιήσετε τις κλάσεις StrikeBoard, ComputerPlayer και BattleShip που δημιουργήσατε, και θα δημιουργήσετε δύο νέες κλάσεις: Την κλάση **ExplorationStrategy** η οποία η οποία υλοποιεί την στρατηγική του παίχτη-υπολογιστή όταν έχει χτυπήσει ένα πλοίο, και την κλάση **ComputerStrategies** που συνδυάζει τις διαφορετικές στρατηγικές.

StrikeBoard: Για την κλάση αυτή χρειαζόμαστε μια μέθοδο **isValidCandidate** η οποία παίρνει σαν όρισμα μια θέση (σαν ένα δισδιάστατο πίνακα από ακέραιους) και επιστρέφει μια boolean τιμή αν η θέση αυτή είναι υποψήφια για το επόμενο χτύπημα (θα πρέπει να είναι εντός ορίων και να μην έχει χτυπηθεί ήδη).

ExplorationStrategy: Η κλάση αυτή υλοποιεί την στρατηγική του παίχτη-υπολογιστή όταν έχει επιτυχημένο χτύπημα όπου εξερευνά την περιοχή γύρω από το επιτυχημένο χτύπημα.

Η κλάση έχει ένα πεδίο StrikeBoard το οποίο αρχικοποιείται στον constructor το οποίο είναι το StrikeBoard του παίχτη. Επίσης χρειάζεται και κάποιες μεταβλητές για να ξέρει την περιοχή που έχει χτυπηθεί. Για το σκοπό αυτό κρατάει δύο θέσεις μία για την αρχή (η κάτω/αριστερά θέση), και μία για το τέλος (την πάνω/δεξιά θέση) του χώρου που έχει μέχρι τώρα χτυπηθεί.

Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Την μέθοδο **initialize** η οποία παίρνει σαν όρισμα μία θέση και αρχικοποιει την αρχική και τελική θέση (στην ίδια θέση).
- Την μέθοδο **nextStrike** η οποία επιλέγει την θέση για το επόμενο χτύπημα από τις υποψήφιες θέσεις γειτονικές προς τον χώρο που έχει ήδη χτυπηθεί. Αυθαίρετα, η σειρά με την οποία εξερευνάει τον χώρο είναι πρώτα οριζόντια (πρώτα δεξιά, μετά αριστερά) και μετά κάθετα (πρώτα πάνω, μετά κάτω). Προφανώς, αν ο χτυπημένος χώρος είναι μεγαλύτερος από μία θέση η εξερεύνηση γίνεται προς την ίδια κατεύθυνση.
- Την μέθοδο **update** η οποία παίρνει σαν όρισμα μια θέση στην οποία υπήρχε επιτυχημένο χτύπημα και ενημερώνει τι θέσεις του χτυπημένου χώρου κατάλληλα.

Μπορείτε να προσθέσετε επιπλέον βοηθητικές μεθόδους για διαδικασίες που χρησιμοποιείτε συχνά (π.χ., για να εξερευνήσετε την επόμενη θέση προς κάποια κατεύθυνσή).

ComputerStrategies: Η κλάση αυτή συνδυάζει τις στρατηγικές του παίχτη-υπολογιστή. Αν δεν υπάρχει επιτυχημένο χτύπημα επιλέγει τυχαία, αλλιώς αν υπάρχει χτυπημένος χώρος επιλέγει με βάση την στρατηγική εξερεύνησης. Η κλάση έχει πεδία RandomStrategy, και ExplorationStrategy που αρχικοποιούνται στον constructor. Επίσης έχει ένα boolean πεδίο το οποίο κρατάει αν είμαστε σε διαδικασία εξερεύνησης.

Η κλάση θα πρέπει να έχει τις εξής μεθόδους:

- Την μέθοδο **nextStrike** η οποία ανάλογα αν είμαστε σε διαδικασία εξερεύνησης ή όχι, επιστρέφει είτε μια τυχαία θέση, είτε την επόμενη θέση εξερεύνησης.
- Την μέθοδο **update** η οποία παίρνει σαν όρισμα μια θέση και μια boolean τιμή που μας λέει αν το χτύπημα σε αυτή την θέση ήταν επιτυχές. Ενημερώνει το αντικείμενο RandomStrategy και αν το χτύπημα ήταν επιτυχημένο, είτε ξεκινάει μια καινούρια εξερεύνηση είτε ενημερώνεται η τρέχουσα.
- Την μέθοδο **completeExploration** με την οποία σταματάει την εξερεύνηση.

ComputerPlayer: Η ComputerPlayer έχει πλέον ένα αντικείμενο ComputerStrategies αντί για RandomStrategy. Η **update** πλέον καλεί την update της ComputerStrategies και χρειάζεται και μια μέθοδο **completeExploration** για να καλέσει την αντίστοιχη της ComputerStrategies

BattleShip: Η μόνη αλλαγή που χρειάζεται είναι όταν ο παίχτης-υπολογιστής βυθίσει ένα πλοίο του παίχτη-χρήστη να ενημερώνουμε τον παίχτη-υπολογιστή να σταματήσει την εξερεύνηση.

Υποδείξεις-Παρατηρήσεις

- Η στρατηγική που περιγράψαμε για τον υπολογιστή, για να δουλεύει σε κάθε περίπτωση θα πρέπει να μπορεί να χειριστεί την περίπτωση που έχουμε πλοία κολλημένα μεταξύ τους (και άρα η εξερεύνηση μπορεί να περνάει από το ένα πλοίο στο άλλο). Μπορείτε να χειριστείτε αυτή την περίπτωση, αλλά είναι ΟΚ να θεωρήσετε ότι ο χρήστης δεν θα βάλει κολλημένα τα πλοία.
- Για bonus βαθμούς μπορείτε να υλοποιήσετε μια πιο έξυπνη στρατηγική για τον υπολογιστή η οποία να λαμβάνει υπόψιν και το μέγεθος των πλοίων που δεν έχει χτυπήσει ακόμη ο υπολογιστής και να αποκλείει κάποιες θέσεις από τις υποψήφιες.

<u>Οδηγίες υλοποίησης</u>

Στην υλοποίηση των κλάσεων σας δεν θα πρέπει να έχετε public πεδία. Βαθμοί θα αφαιρεθούν για προγράμματα που δεν είναι καλά γραμμένα, δηλαδή δεν είναι σωστά στοιχισμένα ή δεν έχουν καλά επιλεγμένα ονόματα μεταβλητών ώστε να διαβάζονται εύκολα. Προγράμματα που δεν κάνουν compile θα πάρουν το πολύ 1 μονάδα από τις 10. Θα πάρετε περισσότερες μονάδες αν έχετε ολοκληρωμένες σωστές μεθόδους, ή μια ολόκληρη κλάση η οποία είναι σωστή, παρά αν έχετε πολύ κώδικα που όμως δεν κάνει compile ή δεν τρέχει.

Τα αρχεία που θα παραδώσετε θα ελεγχθούν για αντιγραφές. Οι αντιγραφές μηδενίζονται.