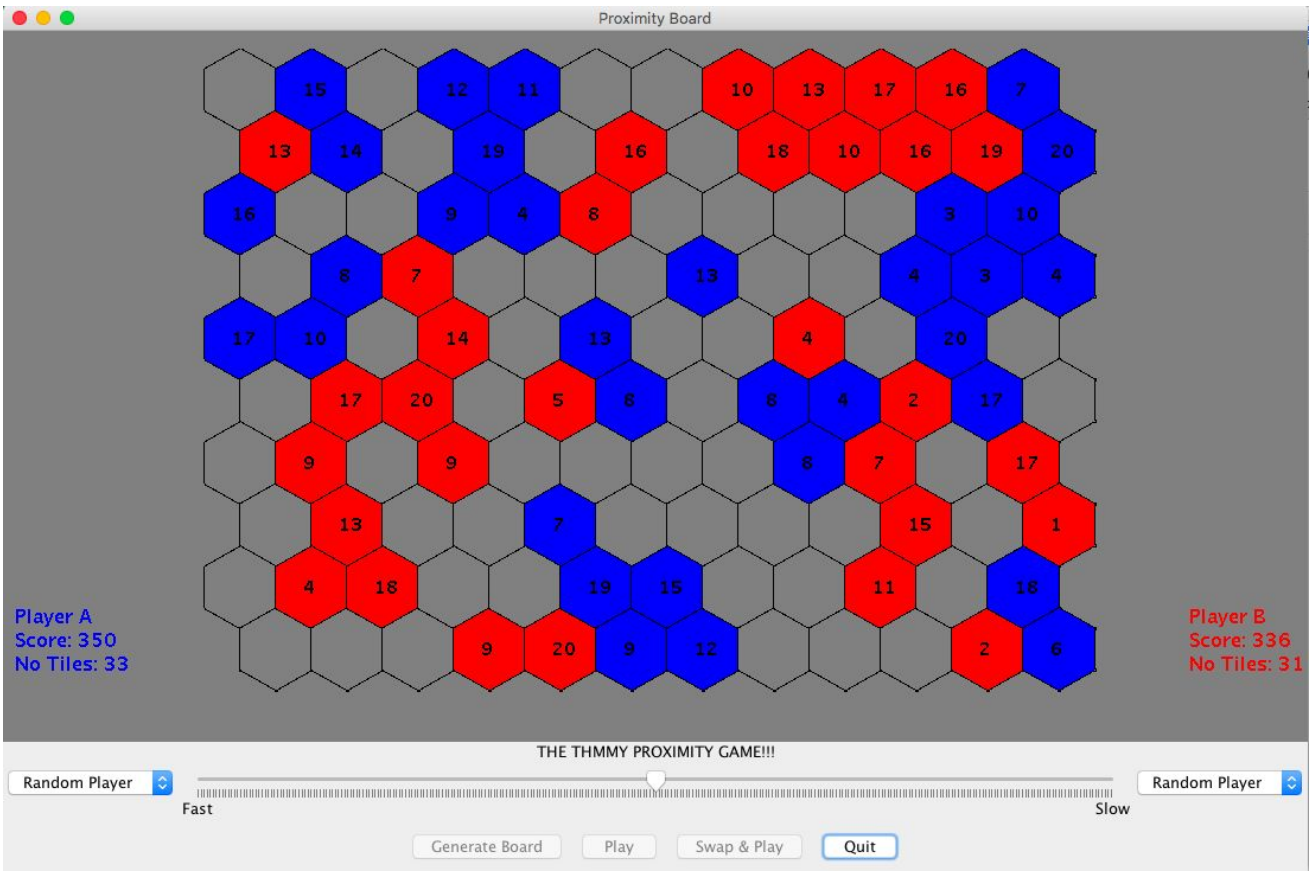
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών

Τομέας Ηλεκτρονικής και Υπολογιστών

Τμήμα Ηλεκτρόγων Μηχανικών και Μηχανικών Υπολογιστών Α.Π.Θ



DS Proximity

Task 1 Report

Mamalakis Evangelos | 8191
Evangelou Alexandros | 8309

Το πρόβλημα

Ζητείται η δημιουργία ενός λειτουργικού παίκτη του ds-proximity ο οποίος θα υλοποιεί το interface AbstractPlayer που έχει δοθεί και θα περιέχει την συνάρτηση getNeighborsCoordinates(). Ο παίκτης θα επιλέγει τις κινήσεις του με τυχαίο τρόπο.

Υλοποίηση Getters και Setters

Στο interface AbstractPlayer περιλαμβάνονται συναρτήσεις που ανακτούν(getters) και τροποποιούν(setters) το περιεχόμενο των τεσσάρων ιδιοτήτων της κλάσης:

```
private int id;
private String name;
private int score;
private int numOfTiles;
```

Ο υλοποίηση γίνεται με τον ίδιο τρόπο για όλες τις ιδιότητες. Για παράδειγμα για την id:

```
public void setId(int id) { this.id = id; }
public int getId() { return id; }
```

Υλοποίηση των Constructors

Ζητείται να υλοποιηθούν δύο constructors οι οποίοι να αρχικοποιούν είτε μόνο το id είτε και τις τέσσερις ιδιότητες της κλάσης

```
public RandomPlayer(Integer id)
public RandomPlayer(Integer id, String name, Integer score, Integer numOfTiles)
```

Η υλοποίηση γίνεται παρόμοια με τους αντίστοιχους setters.

Υλοποίηση της RandomPlayer.getNextMove()

Η συνάρτηση getNextMove() επιλέγει με τυχαίο τρόπο το εξάγωνο στο οποίο θα τοποθετήσει ο παίκτης το πλακίδιο του. Για το σκοπό αυτό επιλέγει δύο τυχαίες τιμές για τις μεταβλητές x και y και ελέγχει αν το αντίστοιχο εξάγωνο είναι ελεύθερο. Αν δεν είναι, επιλέγει δύο άλλες x, y κοκ. μέχρις ότου βρει ένα διαθέσιμο εξάγωνο:

```
do {
        x = (int)(Math.random() * ProximityUtilities.NUMBER_OF_COLUMNS);
        y = (int)(Math.random() * ProximityUtilities.NUMBER_OF_ROWS);
} while (isTaken(board, x, y));
```

Υλοποίηση της RandomPlayer.isTaken()

```
	private boolean isTaken(Board board, int x, int y) {
		return board.getTile(x, y).getColor() != 0;
	}
```

Η συνάρτηση αυτή αναλαμβάνει να προσδιορίσει αν το δοσμένο εξάγωνο με συντεταγμένες x, y έχει ήδη καταληφθεί από κάποιον παίκτη. Για το σκοπό αυτό αξιοποιείται η ιδιότητα color της κλάσης Tile. Όταν και μόνο όταν το εξάγωνο είναι κενό η color έχει την τιμή μηδέν.

Υλοποίηση της RandomPlayer.getNeighborsCoordinates()

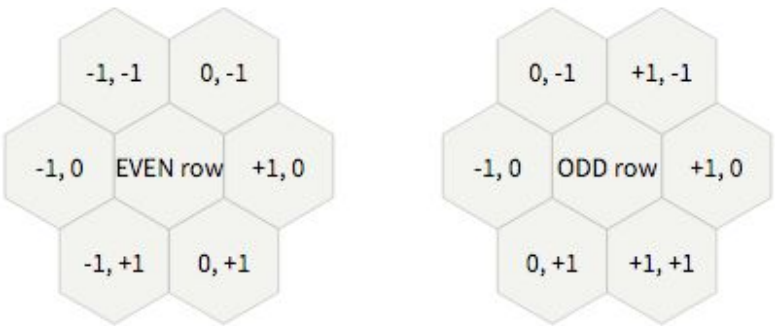
Η συνάρτηση getNeighborsCoordinates() επιστρέφει τις συντεταγμένες των γειτονικών εξάγωνων για ένα δοσμένο εξάγωνο. Οι συντεταγμένες των γειτονικών εξάγωνων εκφράζονται σε σχέση με τις συντεταγμένες του δοσμένου εξάγωνου ανάλογα με το αν αυτό βρίσκεται σε άρτια ή σε περιττή σειρά. Για να ξεχωρίσουμε τις δύο περιπτώσεις ορίζουμε την μεταβλητή yOdd.

Πως παίρνει τιμή η yOdd;

Η έκφραση y & 1 επιστρέφει το τελευταίο byte της δυαδικής αναπαράστασης του ακεραίου y. Οι άρτιοι αριθμοί έχουν πάντα τελευταίο byte ίσο με το 0 ενώ οι περιττοί ίσο με 1. Για παράδειγμα $1=\mathbf{1}_2$, $2=1\mathbf{0}_2$, $3=1\mathbf{1}_2$ κοκ.

```
int yOdd = y & 1; // Check if x is odd
```

Προσδιορισμός των γειτόνων



Αρχικά για τις θέσεις των γειτόνων διακρίνουμε δύο περιπτώσεις, για άρτιες και περιττές σειρές. Γρήγορα όμως παρατηρούμε ότι αυτές διαφέρουν κατά 1 στην συντεταγμένη x, για τους δύο πάνω και τους δύο κάτω γείτονες. Οι μεσαίοι γείτονες έχουν πάντα τις ίδιες σχετικές συντεταγμένες

Έτσι, αφού δημιουργήσουμε τον πίνακα με τις συντεταγμένες για τις άρτιες σειρές, προσθέτουμε την μεταβλητή yOdd στους γείτονες που διαφοροποιούνται Όταν η yOdd έχει τιμή 0 παίρνουμε τον πίνακα για άρτια σειρά, ενώ για yOdd ίσο με ένα παίρνουμε τον πίνακα για περιττή σειρά

```
/* Define the neighbors table
 * When y is even yOdd=0 and we get Even table
 * When y is odd  yOdd=1 and we get Odd table. */
int neighbors[][] = new int[][] {
        {x+1,      y},   // East
        {x  +yOdd, y+1}, // South-East
        {x-1+yOdd, y+1}, // South-West
        {x-1,      y},   // West
        {x-1+yOdd, y-1}, // North-West
        {x  +yOdd, y-1}  // North-East
};
```

Προσδιορισμός μη έγκυρων γειτόνων

Αν το επιλεγμένο εξάγωνο βρίσκεται κοντά στα άκρα του ταμπλό, κάποιο γείτονες θα έχουν μη έγκυρες συντεταγμένες οι οποίες ξεφεύγουν από τα όρια του ταμπλό. Οι τιμές αυτές ζητείται να αντικατασταθούν από την τιμή -1. Για το σκοπό αυτό χρησιμοποιείται η δοσμένη συνάρτηση isInsideBoard() για κάθε πιθανό γείτονα.

```
for ( int i = 0; i < res.length; i++ ) {
        if ( !board.isInsideBoard( res[i][0], res[i][1 ] ) ) {
                res[i][0] = res[i][1] = -1;
        }
}
```

Πηγές και χρήσιμοι σύνδεσμοι

http://www.redblobgames.com/grids/hexagons/