

Τεχνητή Νοημοσύνη

Εργασία 1^η : Reversi

ΜΑΡΙΟΣ ΝΤΕΛΗΣ 3140144

ΓΙΑΝΝΗΣ ΖΕΡΒΟΣ 3160036

ΧΙΛΣΕΝ ΤΣΕΝΑΙ 3160239

Περιληπτική ανάλυση προγράμματος

Η εργασία με την οποία ασχοληθήκαμε είναι το Reversi ή αλλιώς Othello. Ο αλγόριθμος που χρησιμοποιήθηκε για τις κινήσεις που θα κάνει ο υπολογιστής είναι ο MiniMax όπου ήταν προαπαιτούμενο της εργασίας. Στην εργασία έχουμε 4 βασικές κλάσεις την Main, την Reversi, την Board και την Minimax.

Αρχικά, η **κλάση Board** είναι η βασική διεπαφή του χρήστη με το παιχνίδι. Έχουμε σχεδιάσει την διεπαφή έτσι ώστε ο χρήστης να μπορεί να καταλαβαίνει πως παίζεται το παιχνίδι.

Έχουμε χρησιμοποιήσει τις μεταβλητές:

- Char [][] board
- Int YourDisks, CompDisks, remaining
- Char boardX[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'},
- Char boardY[] = {'1', '2', '3', '4', '5', '6', '7', '8'}

Έχουμε τον constructor Board() όπου έχουμε φτιάξει πως θα φαίνεται η διεπαφή του παιχνιδιού στον χρήστη. Φαίνεται παρακάτω:

```
/**
 * Board constructor
 */
public Board(){
    board = new char[][]{
        {'_', '_', '_', '_', '_', '_', '_', '_'},
        {'_', '_', '_', '_', '_', '_', '_', '_'},
        {'_', '_', '_', '_', '_', '_', '_', '_'},
        {'_', '_', '_', 'o', 'x', '_', '_', '_', '_'},
        {'_', '_', '_', 'x', 'o', '_', '_', '_', '_'},
        {'_', '_', '_', '_', '_', '_', '_', '_'},
        {'_', '_', '_', '_', '_', '_', '_', '_'},
        {'_', '_', '_', '_', '_', '_', '_', '_'}
    };
}
```

Έχουμε την μέθοδο **Boolean isValidInput()** όπου ελέγχουμε αν ο χρήστης έχει δώσει σωστές συντεταγμένες και μέσα στα πλαίσια του board για να παίξει.

Έχουμε την μέθοδο **int coordinateX()** όπου μας επιστρέφει την θέση της τιμής του γράμματος το οποίο πληκτρολόγησε ο χρήστης.

Έχουμε την μέθοδο **String coordinateXletter()** όπου επιστρέφει ένα γράμμα του πίνακα boardX[] ανάλογα με τον αριθμό που θα του δώσουμε.

Έχουμε την μέθοδο **void displayBoard()** όπου σχεδιάζουμε την διεπαφή του χρήστη. Βάζουμε τις τιμές του **boardX[]** στον οριζόντιο άξονα πάνω και κάτω από την αρχική διεπαφή του constructor και τις τιμές του **boardY[]** στον κάθετο άξονα δεξιά και αριστερά της διεπαφής. Βασικός σκοπός αυτής της μέθοδου να μπορεί να βλέπει ο χρήστης που μπορεί να το δίσκο του για να παίξει.

Έχουμε την μέθοδο **void showBoardState()** η οποία εμφανίζει την διεπαφή εφαρμόζοντας τις διαθέσιμες κινήσεις που υποδεικνύονται με τελείες (ascii 46).

Έχουμε την μέθοδο **void findLegalMoves()** όπου τσεκάρουμε τις διαθέσιμες κινήσεις του χρήστη ή του υπολογιστή κατά την διάρκεια του παιχνιδιού.

Έχουμε την μέθοδο **ArrayList<Cell> getLegalMoves()** όπου μας επιστρέφει τις διαθέσιμες κινήσεις σε μορφή λίστας μέσω της μεθόδου **findLegalMoves()**.

Έχουμε την μέθοδο **void applyMove()** όπου εφαρμόζουμε την κίνηση του χρήστη στην διεπαφή.

Έχουμε την μέθοδο **int getScore()** όπου μετράμε τους 'δίσκους' του χρήστη και του υπολογιστή τα 'ο' και 'χ' σε όλο τον πίνακα. Έτσι βρίσκουμε το συνολικό σκορ και των δυο.

Έχουμε την μέθοδο **int gameResult()** όπου βλέπουμε 3 περιπτώσεις. Πρώτη περίπτωση είναι να έχουμε βάλει δίσκους σε όλες τις θέσεις και να έχουμε περισσότερους ή λιγότερους ή ίσους με τον υπολογιστή. Δεύτερη περίπτωση είναι να μην έχουμε ή να μην έχει ο υπολογιστής διαθέσιμους δίσκους. Τελευταία περίπτωση να μην έχουμε ή να μην έχει ο υπολογιστής διαθέσιμες κινήσεις. Επιστρέφουμε 1 αν κέρδισε ο χρήστης ή -1 αν κέρδισε ο υπολογιστής ή 0 για ισοπαλία.

Στην συνέχεια, η **κλάση Minimax** περιέχει τον αλγόριθμο MiniMax με τον οποίο ο υπολογιστής θα αξιολογεί τις διαθέσιμες επιλογές του και θα κινείται αντίστοιχα. Επιπλέον, θα δίνεται η δυνατότητα στον χρήστη να επιλέγει το βάθος στο οποίο ο υπολογιστής θα ψάχνει τις ανάλογες κινήσεις.

Έχουμε την βασική μέθοδο **Object[] mmab** όπου γίνεται αξιολόγηση των κινήσεων του υπολογιστή. Κρατάμε τις διαθέσιμες κινήσεις, μέσω της μεθόδου **getLegalMoves()**, σε μια λίστα **ArrayList<Cell> movelist** για να μπορούμε να τις επεξεργαστούμε. Εφόσον υπάρχουν διαθέσιμες κινήσεις και το βάθος στο οποίο ψάχνουμε δεν είναι μηδέν φτιάχνουμε ένα καινούριο **board** για να ελέγξουμε τις κινήσεις. Κρατάμε το καλύτερο σκορ στην μεταβλητή **bestScore**, το τωρινό σκορ που πιθανόν να αλλάξει στην μεταβλητή **tempScore** και τις μεταβλητές **newMove**, **bestMove** για τις καλύτερες κινήσεις σύμφωνα με το σκορ. Άμα υπάρχει καλύτερη πιθανή κίνηση τότε η μέθοδος θα την κρατήσει και θα την επιστρέψει ως **Object[]**.

Έχουμε και την μέθοδο **int evaluate()** όπου επιστρέφει τη διαφορά μεταξύ των αριθμών των 'δίσκων' του υπολογιστή με τον αριθμό των 'δίσκων' του χρήστη.

Τρίτη και βασικότερη **κλάση Reversi** όπου χρησιμοποιούνται μέθοδοι από τις παραπάνω κλάσεις που αναλύσαμε για να μπορεί ο χρήστης να αρχίζει να παίζει.

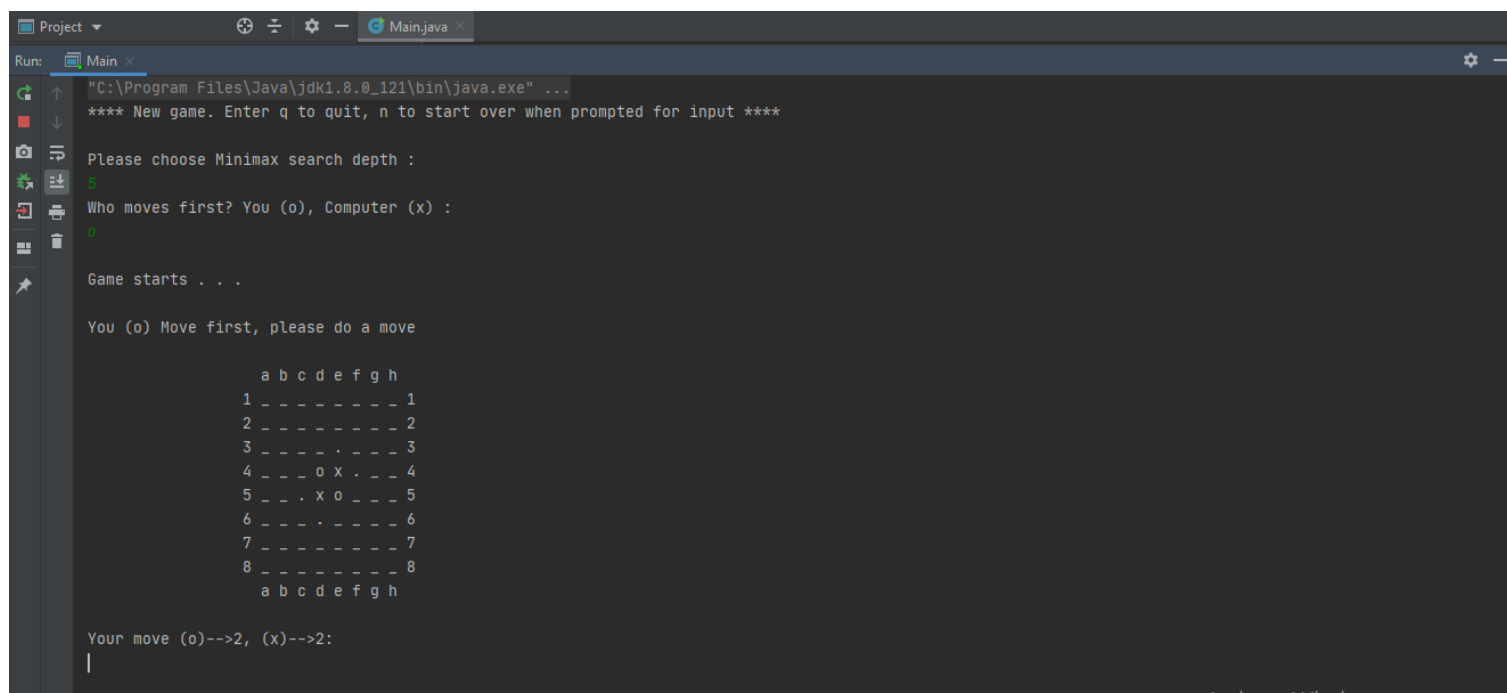
Έχουμε την βασική μέθοδο **static void play()** που είναι η κύρια επικοινωνία με τον χρήστη. Αρχικά ζητείται από τον χρήστη αν θέλει να αποχωρήσει από το παιχνίδι ή αν θα θέλει να ξεκινήσει νέο. Εφόσον θέλει να παίξει ζητούνται το βάθος στο οποίο ο υπολογιστής θα ψάχνει για να κάνει μια πιθανή κίνηση, μέσω του **MiniMax**, και αν θέλει να παίξει πρώτος ο ίδιος ή ο υπολογιστής γράφοντας το αντίστοιχο γράμμα. Αφού γίνει αυτό και έχει επιλεγεί να παίξει πρώτα ο χρήστης εμφανίζεται ο 'πίνακας' στον οποίο θα γίνει το παιχνίδι, μέσω της **showBoardState()**, και οι διαθέσιμες κινήσεις που έχει μέσω της μεθόδου **getLegalMoves()**. Επιπλέον, εμφανίζονται το σκορ του χρήστη και του υπολογιστή μέσω των μεταβλητών **YourDiscs** και **CompDiscs**. Αυτή η διαδικασία επαναλαμβάνεται μέχρι η μέθοδος **getLegalMoves()** να μην έχει άλλες διαθέσιμες κινήσεις και η μέθοδος **gameResult()** να μας δώσει το αποτέλεσμα για το ποιος κέρδισε.

Έχουμε και την μέθοδο **static void wait()** όπου σταματάει την εκτέλεση του αλγορίθμου **MiniMax** για μερικά δευτερόλεπτα.

Τελευταία έχουμε την **κλάση Main** όπου εκτελείται το πρόγραμμα. Δημιουργούμε ένα **Board b** και στην συνέχεια εκτελούμε την μέθοδο **play()** της κλάσης **Reversi**.

Τρόπος χρήσης

Το πρόγραμμα γράφτηκε μέσω του προγράμματος **Intellij IDEA**. Η εκτέλεση του προγράμματος γίνεται μέσω της κλάσης **Main**.



```
Project ▾
Main.java
Run: Main
"C:\Program Files\Java\jdk1.8.0_121\bin\java.exe" ...
**** New game. Enter q to quit, n to start over when prompted for input ****

Please choose Minimax search depth :
5

Who moves first? You (o), Computer (x) :
o

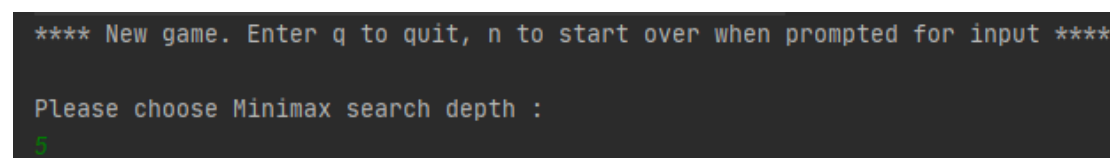
Game starts . . .

You (o) Move first, please do a move

      a b c d e f g h
1  _ _ _ _ _ _ _ 1
2  _ _ _ _ _ _ _ 2
3  _ _ _ . _ _ _ 3
4  _ _ _ o x . _ _ 4
5  _ _ . x o _ _ _ 5
6  _ _ _ . _ _ _ 6
7  _ _ _ _ _ _ _ 7
8  _ _ _ _ _ _ _ 8
      a b c d e f g h

Your move (o)-->2, (x)-->2:
|
```

Παραπάνω μπορούμε να δούμε τι θα φαίνεται στον χρήστη όταν εκτελέσει το πρόγραμμα. Έχουμε δώσει κάποια ενδεικτικά δεδομένα στο πρόγραμμα.



```
**** New game. Enter q to quit, n to start over when prompted for input ****

Please choose Minimax search depth :
5
```

Παραπάνω μπορούμε να δούμε τι θα εμφανίζεται στον χρήστη όταν το πρόγραμμα θα του ζητάει το βάθος στο οποίο να ψάχνει ο υπολογιστής. Το βάθος πρέπει να είναι μεταξύ 1 και 10 για λόγους χρηστικότητας.

Who moves first? You (o), Computer (x) :

o

Game starts . . .

You (o) Move first, please do a move

	a	b	c	d	e	f	g	h	
1	-	-	-	-	-	-	-	-	1
2	-	-	-	-	-	-	-	-	2
3	-	-	-	-	.	-	-	-	3
4	-	-	-	o	x	.	-	-	4
5	-	-	.	x	o	-	-	-	5
6	-	-	-	.	-	-	-	-	6
7	-	-	-	-	-	-	-	-	7
8	-	-	-	-	-	-	-	-	8
	a	b	c	d	e	f	g	h	

Your move (o)-->2, (x)-->2:

|

Παραπάνω μπορούμε να δούμε η επιλογή για το αν ο χρήστης θέλει να παίξει πρώτος ή όχι. Στο δικό μας παράδειγμα έχουμε βάλει την επιλογή να παίξει πρώτος. Πιο κάτω έχουμε τον πίνακα στον οποίο θα παίζει ο χρήστης. Δίπλα από το X και O (τους δίσκους) φαίνονται κάποιες τελείες οι οποίες είναι οι διαθέσιμες κινήσεις του χρήστη. Κάτω από τον πίνακα βλέπουμε το σκορ του χρήστη και του υπολογιστή. Ο χρήστης μπορεί να επιλέξει μια από τις τελείες για να τοποθετήσει το 'δίσκο' του γράφοντας πρώτα το γράμμα (οριζόντια) και μετά τον αριθμό (κάθετα).

Who moves first? You (o), Computer (x) :

x

Game starts . . .

Computer (x) Moves first

	a	b	c	d	e	f	g	h	
1	_	_	_	_	_	_	_	_	1
2	_	_	_	_	_	_	_	_	2
3	_	_	_	.	_	_	_	_	3
4	_	_	.	o	x	_	_	_	4
5	_	_	_	x	o	.	_	_	5
6	_	_	_	.	_	_	_	_	6
7	_	_	_	_	_	_	_	_	7
8	_	_	_	_	_	_	_	_	8
	a	b	c	d	e	f	g	h	

Computer move (x)-->2, (o)-->2 :

Computer is thinking

d3

Computer move (x)-->2, (o)-->2 :

Computer is thinking

d3

	a	b	c	d	e	f	g	h	
1	_	_	_	_	_	_	_	_	1
2	_	_	_	.	_	_	_	_	2
3	_	_	.	x	.	_	_	_	3
4	_	_	_	x	x	_	_	_	4
5	_	_	.	x	o	_	_	_	5
6	_	_	_	_	_	_	_	_	6
7	_	_	_	_	_	_	_	_	7
8	_	_	_	_	_	_	_	_	8
	a	b	c	d	e	f	g	h	

Your move (o)-->1, (x)-->4:

Παραπάνω βλέπουμε τι θα γινόταν αν επιλέγαμε να παίξει ο υπολογιστής πρώτος. Στην δεύτερη εικόνα φαίνεται όπου έχει κάνει μια κίνηση ο υπολογιστής. Οι εικόνες πάνε από αριστερά προς τα δεξιά