1η Εργασία Τεχνητής Νοημοσύνης 2022-2023

Αποστολία Σταματία Μυλωνά [p3200233]

Αλεξάνδρα Μανταλένα Τσιουμποτάριου[p3200212]

Το αρχείο-παιχνίδι αποτελείται από 4 βοηθητικές κλάσεις:

i. Positiom
ii. ComputerMove
iii. Main
iv. Board

Έχουμε ορίσει πως το σύμβολο "B" αναπαριστά τα Black pieces του παιχνιδιού και είναι ο max , ενώ το σύμβολο "W" αναπαριστά τα white pieces του παιχνιδιού και είναι ο min. Με την παύλα αρχικοποιούμε τον πίνακα.

Αναλυτικά έχουμε:

i. **<u>Position</u>**
Χρησιμοποιήσαμε αυτούσια την Κλάση Move του φροντιστηρίου από την στιγμή που μας δόθηκε αυτή η δυνατότητα.

ii. **<u>ComputerMove</u>**
Και αυτή η κλάση είναι εμπνευσμένη από τον κώδικα του φροντιστηριού με την μόνη διαφορά ότι υλοποιήσαμε τον αλγόριθμο με πριόνισμα α-β. Συγκεκριμένα , στις μεθόδους max και min προσθέσαμε τα ορίσματα alpha και beta ως αντιστοιχία των α και β. Αρχικοποιήσαμε αυτές τις τιμές με την μικρότερη δυνατή τιμή από τους ακεραίους και με την μεγαλύτερη αντίστοιχα. Οι τιμές των α και β ορίων μεταβάλλονται στις μεθόδους με τον εξής τρόπο:

- **<u>Τιμή α(alpha)</u>**
  Αλλάζει στην μέθοδο max λαμβάνοντας την μέγιστη τιμή μεταξύ της ήδη υπάρχουσας τιμής α και της τιμής που αντιστοιχεί στην θέση που έχει <<επιλέξει>> ο αλγόριθμος ως κατάλληλης.Με αυτό τον τρόπο ορίζουμε το κατώτατο όριο αναζήτησης.
- **<u>Τιμή β(beta)</u>**
  Αλλάζει στην μέθοδο min λαμβάνοντας την ελάχιστη τιμή μεταξύ της ήδη υπάρχουσας τιμής β και της τιμής που αντιστοιχεί στην θέση που έχει <<επιλέξει>> ο αλγόριθμος ως κατάλληλης.Με αυτό τον τρόπο ορίζουμε το ανώτατο όριο αναζήτησης.

Το πριόνισμα πραγματοποιείται με τον έλεγχο ***if(pos.getSymbol()<=alpha) {return pos;} και if(pos.getSymbol()>=beta){return pos;}*** ανάλογα την ιδιότητα που *τρέχει* κάθε φορά (min /max) διαλέγοντας με αυτόν τον τρόπο την κατάλληλη κίνηση που να βρίσκεται εντός των φραγμάτων.

iii. <u>Board</u>

Σε αυτή την κλάση διαχειριζόμαστε τις κινήσεις των παικτών πάνω στο ταμπλό-πίνακα του παιχνιδιού.Ειδικότερα, έχουμε τις μεθόδους :

- ***<u>public void isValidMove(int row , int col , String currentPlayer)</u>***
  Ελέγχει αν η κίνηση που έχει επιλεχθεί από τον παίκτη ή από την MiniMax είναι επιτρεπτή με βάση τους κανονισμούς του παιχνιδιού και εκτελεί την κίνηση και με ότι αυτό συνεπάγεται(κάλυψη πιονιων αντιπάλου). Για να υλοποιηθούν τα παραπάνω καλούνται σταδιακά και υπό περιπτώσεις οι βοηθητικές μέθοδοι MoveCheck και MakeMove, οι οποίες αναλύονται παρακάτω.
  Με πολλαπλές if και δίνοντας κατάλληλες τιμές(-1,0,1) στους βοηθητικούς τελεστές RowIncrease και CollIncrease , ελέγχουμε προς τα πάνω , προς τα κάτω, διαγώνια , αριστερά και δεξιά από την θέση που μας έχει δοθεί *(βλέπε κώδικα).*
- ***<u>public boolean check(int row , int col,String currentPlayer)</u>***
  Λειτουργεί σαν την isValidMove με την διαφορά ότι δεν πραγματοποιεί κάποια κίνηση και επιστρέφει το αποτέλεσμα του ελέγχου , αν η κίνηση είναι επιτρεπτή ή όχι προς μία οποιαδήποτε κατεύθυνση.
- ***<u>public boolean MoveCheck(int row , int col , int RowIncrease , int CollIncrease , String currentPlayer)</u>***
  Ψάχνει προς την κατεύθυνση που της έχει δοθεί προσθέτοντας τα βήματα από τα δοθέντα ορίσματα.Αν συναντήσει το σύμβολο του αντίπαλου παίκτη , συνεχίζει να ψάχνει μέχρι να συναντήσει κενό κελί (-)ή το σύμβολο του τρέχοντος παίκτη που αφορά την κίνηση. Αν συναντήσει το πρώτο επιστρέφει false καθώς αν τοποθετηθεί στο δοθέντα σημείο και προς την κατεύθυνση που αναζητήσαμε δεν δημιουργείται το απαιτούμενο <<sandwich>> , δηλαδή δεν περιέχονται συνεχόμενα σύμβολα του αντιπάλου. Αντίθετα, αν συναντήσουμε το σύμβολο του παίκτη που επιθυμεί να εκτελέσει συτή την κίνηση σημαίνει πως έχουν <<εγκλωβιστεί>>ένα ή παραπάνω πιόνια του αντιπάλου. Όσο η σάρωση του πίνακα θα συναντάει το σύμβολο του αντιπάλου , θα συνεχίζεται μέχρι να εντοπίσει κάτι από τα παραπάνω.
- ***<u>public void makeMove(int row , int col , int RowIncrease , int CollIncrease ,String currentPlayer)</u>***
  Η μέθοδος αυτή αλλάζει τα σύμβολα του αντιπάλου με το σύμβολο του τρέχοντος παίκτη που είναι τοποθετημένα στον πίνακα προς την

κατεύθυνση που η MoveCheck έγκρινε πως υπάρχουν και καθιστά επιτρεπτή την κίνηση του παίκτη.

- ***public boolean isOver(String symbol)***
  Σαρώνει όλο τον πίνακα και για κάθε κελί του καλεί την check , τσεκάροντας αν είναι εφικτή η κίνηση σε εκείνο το κελί για το παίκτη με το σύμβολο symbol . Αν βρει έστω και ένα κελί στο οποίο η check επιστρέφει true , η isOver επιστρέφει false καθώς υπάρχει έστω και μία επιτρεπτή κίνηση για έναν τουλάχιστον παίκτη.Αν ολοκληρωθεί η σάρωση του πίνακα επιστρέφουμε true καθώς δεν εντοπίστηκε επιτρεπτή κίνηση για το σύμβολο που εξετάσαμε.
- ***public ArrayList<Board> getChildren(String symbol)***
  Είναι η αντίστοιχη μέθοδος του φροντιστηρίου την οποία προσαρμόσαμε στις ανάγκες του κώδικα μας.
- ***public int evaluate()***
  Χρησιμοποιούμε έναν πίνακα weight που σε κάθε θέση έχουμε τοποθετήσει ένα αντίστοιχο βάρος.Οι τιμές των θέσεων συσχετίζονται με την πιθανότητα ο αντίπαλος παίκτης να μπορεί να καλύψει εκείνο το κελί. Για παράδειγμα, αν τοποθετηθεί ένα σύμβολο στις γωνίες , τότε είναι αδύνατη η κάλυψη του και για αυτό δίνουμε την μέγιστη τιμή του πίνακα μας , την τιμή 4 και ταυτόχρονα η τοποθέτηση σε εκείνο το κελί μπορεί να καλύψει πολλά πιόνια του αντιπάλου.
  Με βάση την δυσκολία κάλυψης της θέσης από τον αντίπαλο και την ευκολία κάλυψης όσων των δυνατών περισσότερων θέσεων του αντιπάλου πραγματοποιήσαμε την τοποθέτηση των βαρών.
  Για την υλοποίηση της χρησιμοποιήσαμε και μία άλλη βοηθητική μέθοδο την countPoints την οποία αναλύουμε παρακάτω.
  Σε γενικές γραμμές , βρίσκουμε το MaxUtility που αφορά το άθροισμα των βαρών των μαύρων συμβόλων και το MinUtility που αφορά το άθροισμα των βαρών των άσπρων συμβόλων.Επιστρέφουμε την διαφορά τους.

- ***public int countPoints(String symbol)***
  Αθροίζει τα βάρη των θέσεων που περιέχουν το σύμβολο symbol και επιστρέφει το άθροισμα.
- ***public Position getMove()***
  Επιστρέφει την τελευταία κίνηση που πραγματοποίηθηκε.
- ***public int getSumBlack()***
  Επιστρέφει το score του παίκτη με το σύμβολο B.

- ***public int getSumWhite()***
  Επιστρέφει το score του παίκτη με το σύμβολο W.

Σημείωση: Ως score ορίζουμε το πλήθος των κελιών που έχουν κάποιο σύμβολο , ανάλογα την περίπτωση.

- ***public void printBoard()***
  Εμφανίζει τον πίνακα του παιχνιδιού.

iv. Main

Αρχικά , ζητάμε από τον χρήστη να μας δώσει τις ζητούμενες πληροφορίες καθώς και το όνομα του κάνοντας τους απαραίτητους ελέγχους εγκυρότητας.Για παράδειγμα , το μέγιστο βάθος δεν θα πρέπει να είναι μικρότερο ή ίσο του 0.

Το παιχνίδι εκτελείται όσο υπάρχει έγκυρη κίνηση για τουλάχιστον ένα από τους δύο παίκτες.Αν δεν υπαρχει κάποια πιθανή κίνηση γίνεται break και σταματάει το παιχνίδι. Διαφορετικά, αν ο παίκτης του οποίου είναι η σειρά δεν μπορεί να κάνει κάποια κίνηση , αλλάζουμε την σειρά του παίκτη και παίζει ο αντίπαλος.

Όταν παίζει ο υπολογιστής καλείται η MiniMax και επιλέγει την καλύτερη θέση τοποθέτησης και στην συνέχεια καλούμε την isValidMove για να κάνει την τοποθέτηση και τις αλλαγές.

Όταν παίζει ο παίκτης του ζητάμε την σειρά και την στήλη στην οποία θέλει να τοποθετήσει το σύμβολο του. Ελέγχουμε μέσω της isValidMove αν η τοποθέτηση είναι επιτρεπτή.Αν είναι, συνεχίζουμε ενημερώνοντας τον παίκτη πως έκανε σωστή επιλογή και αλλάζουμε το currentSymbol δίνοντας έτσι την σειρά στον αντίπαλο να παίξει. Αντίθετα , αν δεν είναι τον ενημερώνουμε και κρατάμε ίδιο cuurentSymbol δίνοντας έτσι την ευκαιρία στον ίδιο παίκτη να παίξει.

Τέλος , όταν λήξει το παιχνίδι εμφανίζουμε τα αποτελέσματα και τον νικητή.

Παρακάτω παραθέτουμε μερικά στιγμιότυπα των αποτελεσμάτων του παιχνιδιού έτσι όπως εμφανίζεται στο cmd:

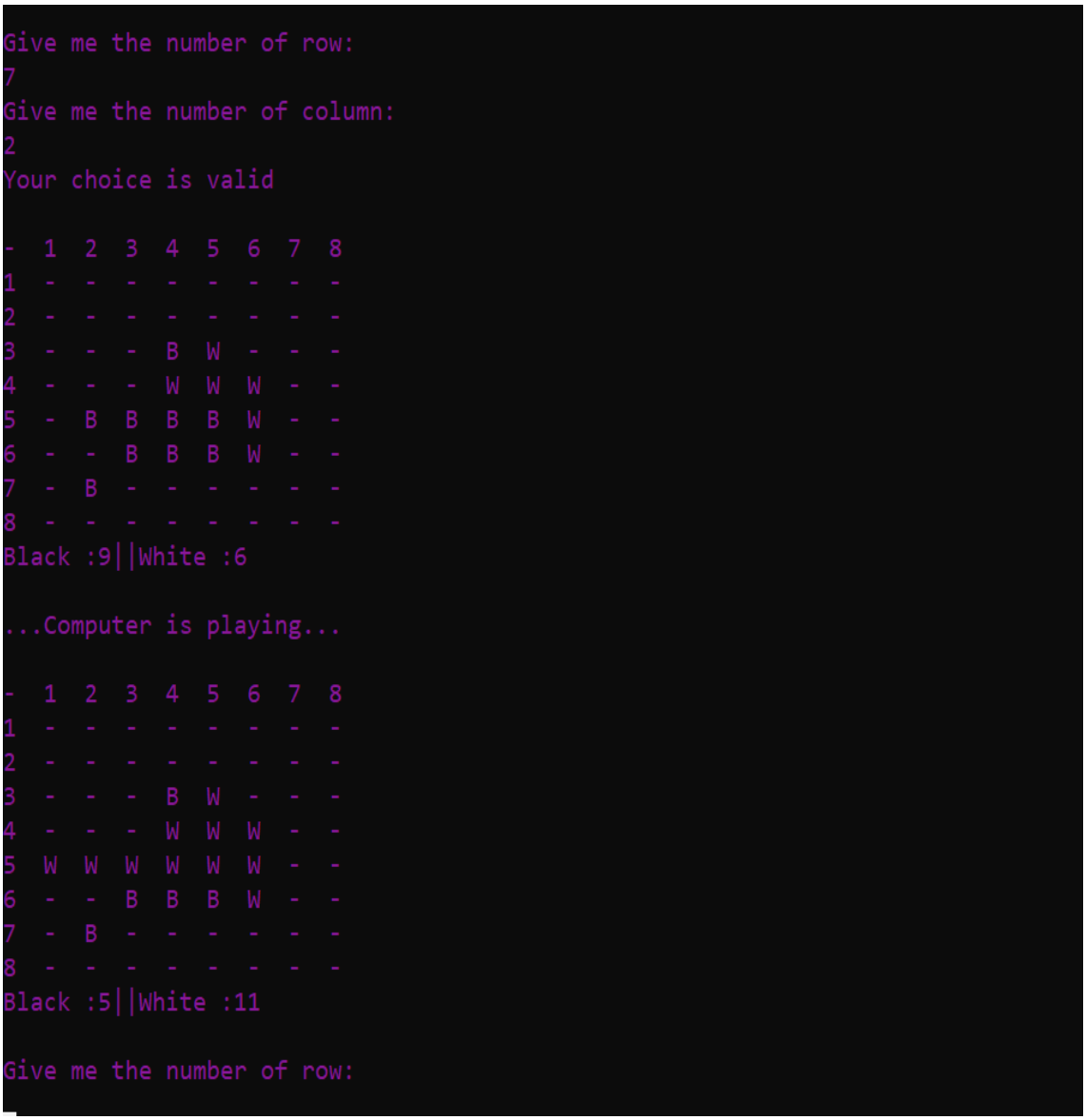
1. Το παιχνίδι ξεκινάει

```
C:\Users\madal\OneDrive\Υπολογιστής\Othello>java Main
....Welcome to the game....
Hello, my name is Madapo!What's your name?
Peggy
Because I am a kind person...Maybe do you want to play first?(Yes/No)
Yes
Now give me the maxDepth..:
3
You have the black pieces.Your symbol is <B>.

-  1  2  3  4  5  6  7  8
1  -  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  W  B  -  -  -
5  -  -  -  B  W  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
8  -  -  -  -  -  -  -  -
Black :2||White :2

Give me the number of row:
```

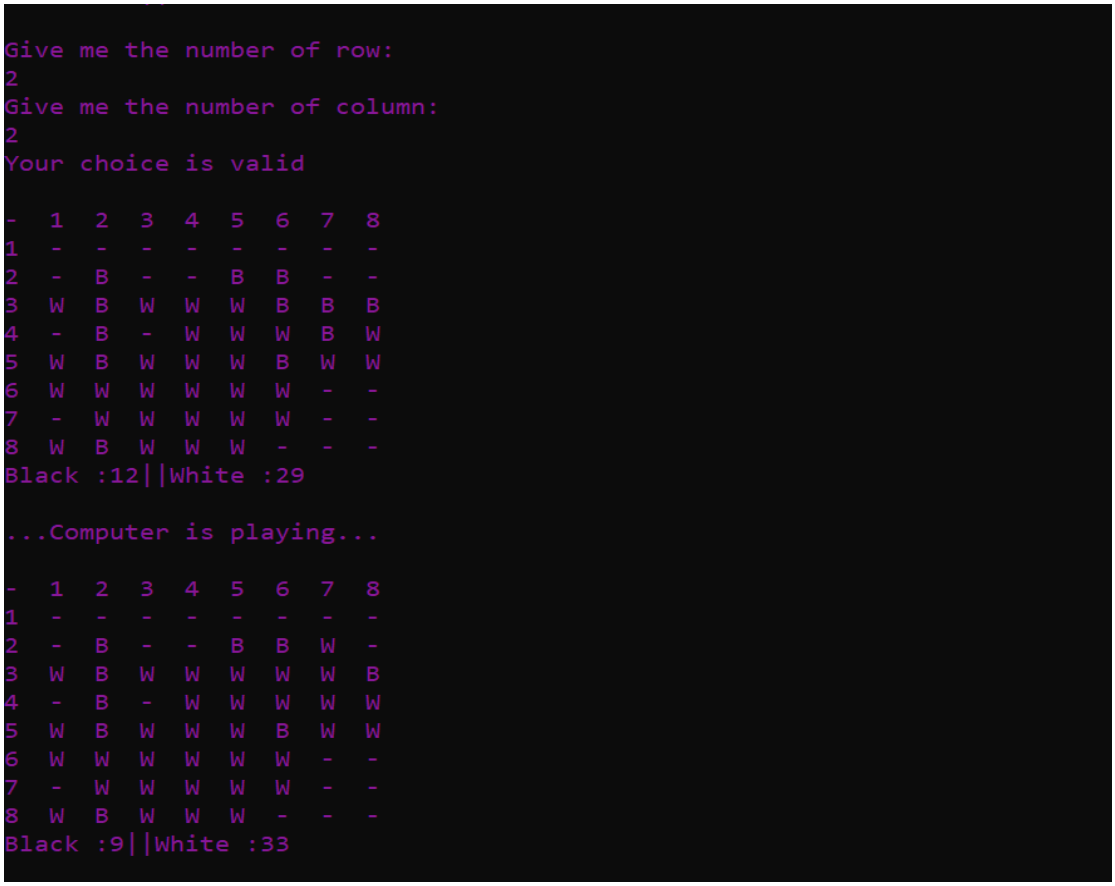
2. Λάθος επιλογή κίνησης

```
-  1  2  3  4  5  6  7  8
1  -  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  W  B  -  -  -
5  -  -  -  B  W  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
8  -  -  -  -  -  -  -  -
Black :2||White :2

Give me the number of row:
2
Give me the number of column:
4
Please try again
```

3. Σωστή επιλογή κίνησης (Περίπτωση1)



4. Σωστή επιλογή κίνησης (Περίπτωση2)



5. Τερματισμός παιχνιδιού