Τεχνητή Νοημοσύνη

[ΕΡΓΑΣΙΑ 1η]

ΟΝΟΜΑ: Σταύρος Ζαχαρόπουλος , ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 3200052

ΟΝΟΜΑ: Σωτήριος-Παναγιώτης Κουλουρίδης , ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 3200082

ΟΝΟΜΑ: Δημήτριος Δρυς , ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: 3200045

ΣΤΟΧΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ: Δημιουργία παιχνιδιού Othello με αντίπαλο AI που επιλέγει τις κινήσεις του με βάση τον αλγόριθμο Minimax.

ΑΡΧΕΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ: Main.java, Player.java, Pair.java, Board.java

Pair.java

Το πιο απλό αρχείο του προγράμματος, περιέχει μια κλάση της οποίας ο σκοπός είναι να να δημιουργεί απο μια μεταβλητή χ και μια μεταβλητή ψ ένα ζευγάρι συντεταγμένων με την μορφή (χ,ψ) . Επίσης η κλάση περιέχει τα στοιχεία colour και value τα οποία δείχνουν το χρώμα και την αξία που έχει στο σκορ του παιχνιδιού το πούλι που βρίσκεται σε αυτές τις συντεταγμένες.

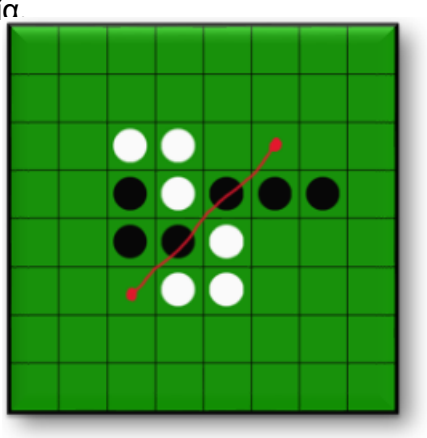
Επιπροσθέτως το αρχείο περιέχει τις συναρτήσεις **getX**, **getY**, **getColor** που επιστρέφουν τις αντίστοιχες μεταβλητές ενώ η συνάρτηση **giveColor** αλλάζει το χρώμα του πουλιού με βάση αυτό που παίρνει σαν όρισμα.

Board.java

Στο αρχείο αυτό έχουμε ουσιαστικά την υλοποίηση του παιχνιδιού. Αρχικά λοιπόν έχουμε την δημιουργία ενος array της μορφής char[8][8] με το όνομα **board** το οποίο γεμίζουμε με ‘ ‘ ,με κενά δηλαδή ενώ στο κέντρο του πίνακα τοποθετούμε δύο μαύρα και δύο άσπρα πούλια διαγώνια το ένα με το άλλο. Έτσι λοιπόν έχουμε δημιουργήσει τον αρχικό πίνακα.

Με την συνάρτηση **availablePairs** επιστρέφουμε μια λίστα η οποία περιέχει όλες τις διαθέσιμες θέσεις που μπορεί να παίξει ο παίκτης. Η λογική της είναι ότι διατρέχει τον πίνακα εώς οτου συναντήσει πούλι του αντιπάλο, τότε καλεί την συνάρτηση **helpPair** για τις θέσεις πάνω, κάτω, δεξιά και αριστερά **helpPairDiag** ενώ για τις θέσεις πάνω δεξιά διαγώνια, κάτω δεξιά διαγώνια, πάνω αριστερά διαγώνια και κάτω αριστερά διαγώνια. Οι συναρτήσεις αυτές λοιπόν αρχικά ελεγχουν αν τα κελιά αυτά είναι κενά καθώς αν δεν είναι αυτόματα αποκλείεται από την λίστα. Αν λοιπόν είναι κενή τότε διατρέχει την ανάλογη ευθεία π.χ αν εξετάζεται η κάτω αριστερά διαγώνια θέση εξετάζουμε την πάνω δεξιά διαγώνιο ευθεία

Ξεκινώντας λοιπόν από την κάτω κόκκινη κουκίδα σκανάρει την κόκκινη διαγώνιο εώς ότου συναντήσει κενό ή άσπρο πούλι.



Αν συναντήσει άσπρο πούλι τότε αυτό σημαίνει ότι η θέση είναι κανονική ενώ αν συναντήσει κενό, όπως στην πάνω κόκκινη κουκίδα, τότε δεν είναι.

Όμοια λειτουργεί και η **helpPair** για την αριστερή θέση του πουλιού και την δεξιά ευθεία για παράδειγμα. Αφού ελέγχει όλες τις πιθανές θέσεις για το πούλι αυτο τοτέ συνεχίζει τον έλεγχο μέχρι το επόμενο πούλι του αντιπάλου.

Ακόμα το αρχείο περιέχει την συνάρτηση **makeMove** η οποία δίνει στο Pair το ανάλογο χρώμα του κάθε παίκτη στην θέση που έχει επιλέξει ο παίκτης να τοποθετήσει το πούλι του και το τοποθετεί στο Array. Έτσι τοποθετεί το πούλι και πραγματοποιεί την κίνηση.

Η συνάρτηση **NoMoves** πραγματοποιεί έναν απλό έλεγχο του αν η λίστα με τα διαθέσιμες θέσεις είναι κενή.

Ακόμα η συνάρτηση **updateBoard** ουσιαστικά αντικαθιστά μετά την κίνηση του παίκτη τον πίνακα με τον νέο ανανεωμένο που έχει αλλάξει το χρώμα σε όσα πούλια χρειάζεται. Αυτό επιτυγχάνεται με το να διέρχεται η λίστα Pairs με τις διαθέσιμες συντεταγμένες εώς ότου να βρεθεί οι συντεταγμένες που έπαιξε ο παίκτης. Αφού υπάρχουν στην λίστα λοιπόν εντοπίζουμε στην δεύτερη λίστα την Ogs η οποία περιέχει τις συντεταγμένες του αντιπάλου πουλιού που εξετάστηκε για να προκύψει η διαθέσιμη θέση αυτή και αφαιρούμε αντίστοιχα τα χ και τα ψ των δύο θέσεων. Αν λοιπόν προσθέσουμε τα αποτελέσματα αυτά στις αρχικές συντεταγμένες της πρώτης θέσης παρατηρούμε ότι αρχίζει να κινείται προς την ευθεία που ελεγχουμε. Μέχρι λοιπόν στην ευθεία αυτη να συναντήσουμε πούλι του παίχτη μετατρέπουμε όλα τα πούλια του αντιπάλου σε όμοια με αυτα του παίχτη.

Η συνάρτηση **score** επιστρέφει το σκορ του παιχνιδιού απονέμοντας 1 βαθμό για κάθε πούλι .

Η συνάρτηση **DesideFirstOrNot** δίνει στον παίκτη την δυνατότητα να επιλέξει ποιός θα παίξει πρώτος, αυτός η ο υπολογιστής.

Player.java

Η κλάση αυτή παίρνει ως ορίσματα το **maxDepth** και το **PlayerColor** δηλαδη το μέγιστο βάθος αναζήτησης της Minimax και το χρώμα του κάθε παίκτη.

Η συνάρτηση **getColor** επιστρέφει το χρώμα του παίκτη δηλαδή το **PlayerColor**.

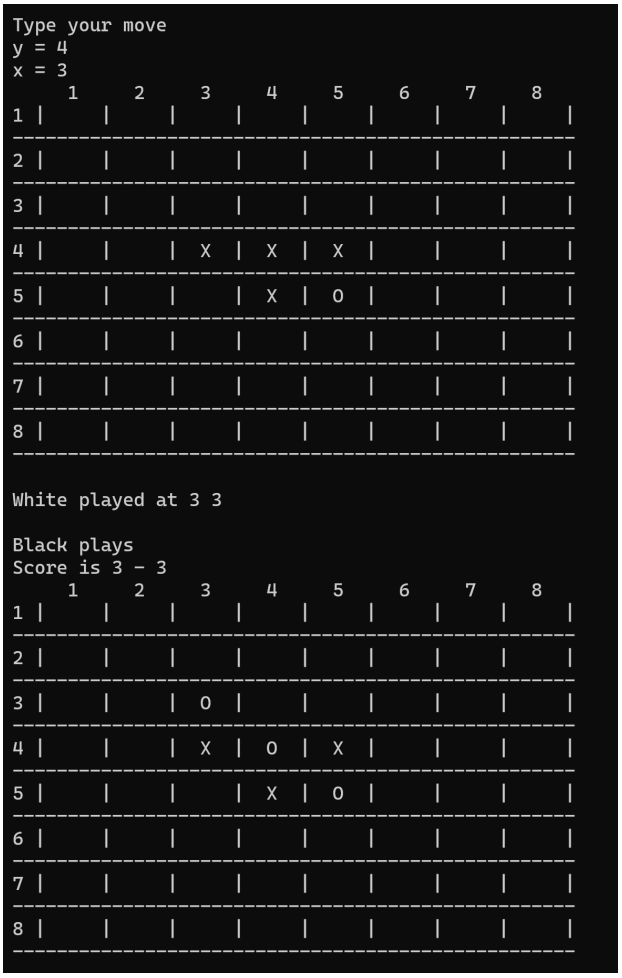
Η συνάρτηση **desideDepth** ζητάει από τον παίκτη να ορίσει το μέγεθος της **maxDepth** και στην συνέχεια το τοποθετεί στην αντίστοιχη μεταβλητή.

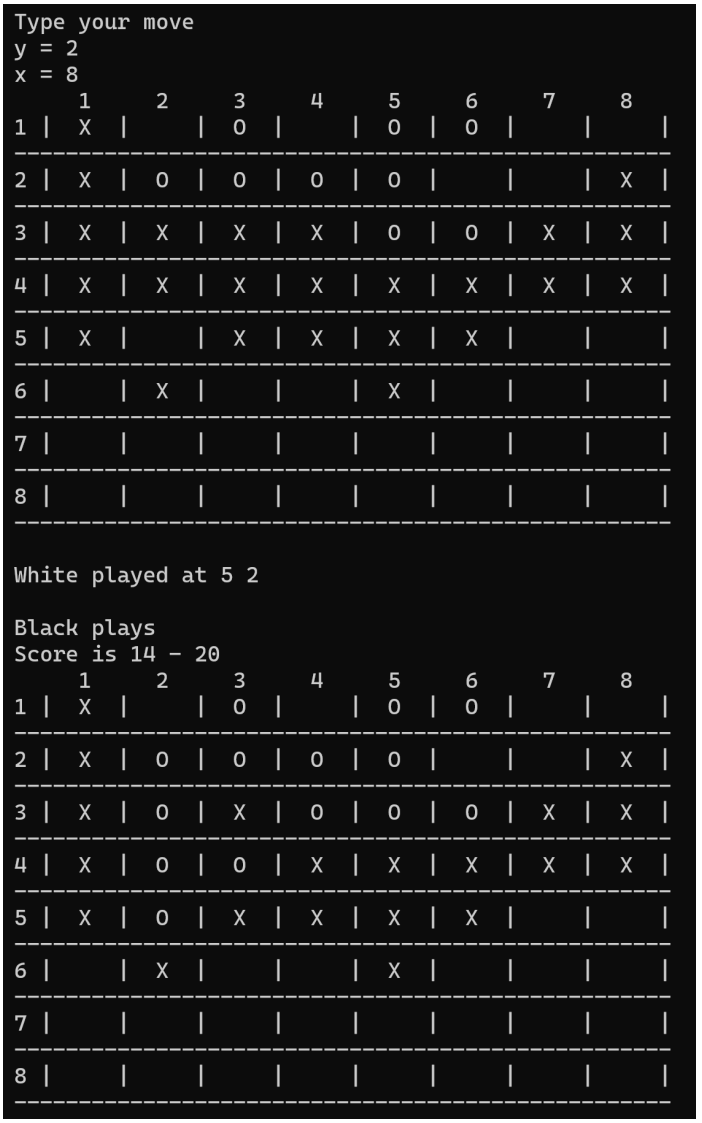
Η συνάρτηση **DesideMove** δημιουργεί ένα Pair με τις συντεταγμένες τις οποίες παίρνει σαν input από τον παίκτη.

Η **MoveMiniMax** είναι η συνάρτηση η οποία παίρνει τις αποφάσεις για το παίκτη AI αρχικά ελέγχει αν το βάθος αναζήτησης είναι 0, αν ναι τότε επιστρέφει το **evaluate*** του κόμβου αυτού, αλλιώς διατρέχει τις διαθέσιμες κινήσεις και πραγματοποιεί κάθε μια από αυτές σε ένα αντίγραφο του αρχικού πίνακα το οποίο δημιουργείται με την συνάρτηση mimic, στην συνέχεια καλεί αναδρομικά πάλι την **MoveMiniMax** για ένα επίπεδο βάθους μικρότερο από αυτό αυτής της κλήσης. Η διαδικασία αυτή επαναλαμβάνεται μέχρι το depth=0. Με κάθε κλήση της η συνάρτηση **MoveMiniMax** επιστρέφει μια τιμή **value** η οποία συγκρίνεται με την μεταβλητή **best**

και αν είναι μεγαλύτερη παίρνει την θέση της, όμοια σύγκρισή πραγματοποιείται μεταξύ των μεταβλητών **alpha** και **best**. Η σύγκρισή αυτή γίνεται για να πραγματοποιηθεί το πριόνισμα το οποίο γίνεται με τον έλεγχο (beta<=alpha) δίνοντας έτσι τις μέχρι τώρα κινήσεις τον βέλτιστο συνδυασμό για τον επόμενο παίκτη καθιστώντας έτσι αχρείαστο καποιον περαιτέρω έλεγχο και έτσι παραλείπεται. Με αυτόν τον τρόπο λοιπόν η συνάρτηση επιστρέφει την βέλτιστη επιλογή για τον παίκτη AI έχοντας ερευνήσει σε βάθος depth κινήσεων το παιχνίδι.

Αρχικό στάδιο





Στην αριστερή πάνω εικόνα είναι το στάδιο ύστερα από την κίνηση (4,3) του παίκτη X και στην αριστερή κάτω εικόνα είναι το στάδιο ύστερα από την κίνηση (3,3) του παίκτη O.

Στην δεξιά πάνω εικόνα είναι ένα στάδιο κοντά στην μέση του παιχνιδιού και στην δεξιά κάτω η αμέσως επόμενη κίνηση.

Σε αυτό το στάδιο το παιχνίδι έχει φτάσει στο τέλος του με νικητή την AI, αφού η βαθμολογία είναι 22 - 42 με το δεύτερο να είναι η βαθμολογία της AI.