Γιώργος Δημόπουλος 3150035

Τρύφωνας Μπιτσάκος 3160108

Γιάννης Παπαγεωργίου 3160129

Αρχικά το project μας διαθέτει τέσσερις κλάσεις.

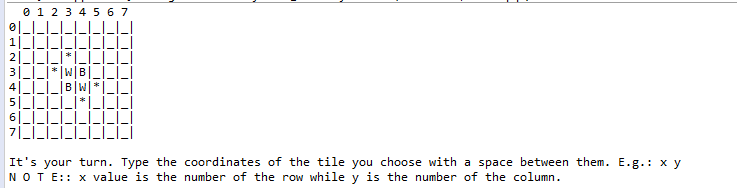
Η πρώτη κλάση Tile, που στην ουσία αντιπροσωπεύει το κάθε πλακάκι,

Έχει κατασκευαστή που αποδίδονται μια απο τις τέσσερις πιθανές τιμές(States) :άδειο space ,άσπρο W ,μαύρο B,δυνατή κίνηση \*.Κλασσικος getter και setter και επιπλέον την μέθοδο equals που συγκρίνει αν δύο πλακάκια έχουν την ίδια κατάσταση.

Η δεύτερη κλάση Board,που αντιπροσωπεύει τον πίνακα του παιχνιδιού.

Περιέχει μια στάτικ μεταβλητή moves, ωστέ να γίνεται σωστή ενημέρωση για τις δυνατές κινήσεις κάθε παίχτη.Έχουμε ορίσει τον πίνακα MY,ο οποίος περιέχει την βαθμολογική αξία κάθε πλακακιού(για την ευρετικη).Μέθοδοι:

public void initBoard (Tile[][] board):εμφάνιζει τον πίνακα στην αρχική κατάσταση του παιχνιδιού



public void copyBoard (Tile[][] target, Tile [][] source): αντιγράφει τον πίνακα target στον πίνακα source

private void updateTiles (States st, int x, int y):αποθηκεύει τις συντεταγμένες των κινήσεων του αλγόριθμου(myTiles) ή του παίκτη(opponent) αντίστοιχα σε πίνακες .

public void printBoard(Tile [][] matrix):κλασσικό τύπωμα,απλά κάνει αντικατάσταση ανάλογα με τις πιθανές καταστάσεις των tiles(-,W,B,\*)και φτιάχνει το σχέδιο του reversi.

public int [] getLastMove(), public void setLastMove(int[] lastMove) κλασσικος getter,setter περί της τελευταίας κίνησης.

public int findPairInList(List<int[]> list, int x, int y) :χρησιμοποιείται στην παρακάτω μέθοδο,καθώς επιστρέφει που βρέθηκε το δοσμένο ζεύγος συντεταγμένων στο δοσμένο πίνακα.

public void readMove():όπως αναφέρθηκε πιο πάνω με την βοήθεια της παραπάνω μεθόδου,αφού διαβάσουμε τις συντεταγμένες της κινήσεως και τις αποθηκεύσουμε στον opponentsMove, έπειτα με βάση την θέση που μας επέστρεψε η findPairInList οτι βρήκε τις δοσμένες συντεταγμένες, κρίνεται αν είναι έγκυρες οι δοσμένες συντεταγμένες ή όχι και σε περίπτωση λάθους εμφάνιζει τα ανάλογα μηνύματα άναλογα το σφάλμα(εγκυρότητας κίνησης ή λάθος στο ινπουτ).

public void findLegalMoves (List<int[]> list, States color, States othercolor): δέχεται ως ορίσματα τα πιόνια (είτε του υπολογιστή είτε του αντιπάλου) το χρώμα του εν λόγω παίκτη και το χρώμα του αντιπάλου του.

Για να βρούμε τις κινήσεις στον πίνακα εργαζόμαστε ως εξής. Κάθε πιόνι μπορεί να έχει κίνηση στις 8 διαφορετικές κατευθύνσεις γύρω του. Για να συμβολίσουμε τις 8 κατευθύνσεις χρησιμοποιούμε τους 2 πίνακες final int[] dx = {-1, -1, 0 ,1 ,1 ,1 0, -1} και final int[] dy = {0, 1, 1, 1, 0, -1, -1, -1}. Με for για τις 8 κατευθύνσεις παίρνουμε μία κατεύθυνση από τους 2 πίνακες για τον συνδυασμό (x,y). Σε μια οποιαδήποτε θέση εάν προσθέσουμε έναν συνδυασμό (x,y) θα κινηθούμε προς την κατεύθυνση που συμβολιζουν τα (x,y).

(-1,0) αριστερά

(-1,1) κάτω αριστερά

(0, 1) κάτω

(1, 1) κάτω δεξιά

(1, 0) δεξιά

(1, -1) πάνω δεξιά

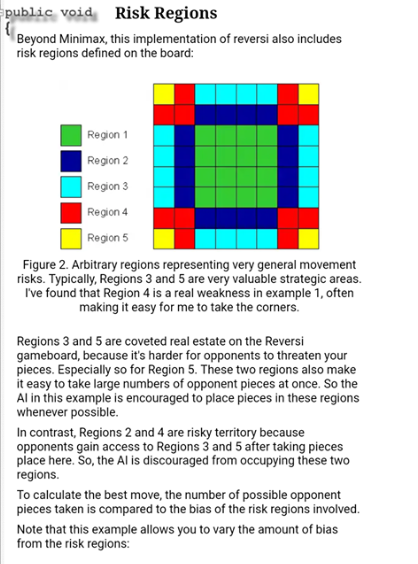
(0,-1) πάνω

(-1,-1) πάνω αριστερά

Έτσι σε κάθε βήμα για κάθε πιόνι ελέγχει τις 8 κατευθύνσεις και εάν σε κάποια κατεύθυνση βρει αμέσως μετά πιόνι του αντιπάλου συνεχίζει προς την ίδια κατεύθυνση μέχρι να βρει κενό το οποίο σημαίνει ότι βρήκε έγγυρη κίνηση.

Τα sx, sy είναι η προσαύξηση κατά ένα βήμα προς την κατεύθυνση που δείχνει το k και οι έλεγχοι (i+sx), (j+sy) κρατάνε την αναζήτηση εντός ορίων. Εάν σε μια συγκεκριμένη κατεύθυνση βρούμε States.LEGALMOVE τότε η κίνηση έχει βρεθεί ήδη ενώ εάν βρούμε color δηλαδή ίδιο χρώμα πιόνι με το δικό μας τότε το πιόνι δεν γίνεται να έχει κίνηση σύμφωνα με τους κανόνες του παιχνιδιού. Τέλος εάν η κίνηση είναι έγκυρη τότε την προσθέτουμε στο moves μαζί με την αξιολόγηση της ευρετικής.

public void makeMove (Tile.States color, int[] move):εφόσον μέσω της findlegalmoves έχουμε βρεί τις πιθανές κινήσεις και διαβάσουμε την κίνηση,τώρα με την συγκεκριμένη μέθοδο χρησιμοποιούμε παρόμοια τεχνική ωστέ να βρεθούν οι αντικαταταστάσεις που πρέπει να γίνουν προς όλες τις κατευθύνσεις μετά την δοσμένη κίνηση(ελέγχονται το πολύ 8 πιθανές κατευθύνσεις) και με την updateTiles γίνεται η ανάλογη ενημέρωση.

public int appreciateMove (int i,int j, States othercolor):με την συγκεκριμένη μέθοδο γίνεται η αξιολόγηγη κινήσεων,ωστέ ο minimax να βρεί την κίνηση που θα μας δώσει το μέγιστο όφελος.Με βάση τον πίνακα MY που εμπεριέχει τις αξιές των θέσεων που δώσαμε στο κάθε πλακίδιο του τραπεζιού του reverse(η παρακάτω εικόνα αποτελεί έμπνευση των αξιών) και με βάση το πόσες κινήσεις θα έχουμε διαθέσιμες,που είναι η καλύτερη στρατηγική κατά κοινή ομολογία,γίνεται η τελική αξιολόγηση της κίνησης.Δίνεται όμοιο βάρος θέσης πλακιδιίου και δυνατών κινήσεων που προσφέρονται μετέπειτα.

Tέλος η pass():ελέγχει αν πρέπει κάποιος παίχτης να πάει πάσο,ελέγχοντας μέσω της στατικής μεταβλητής avmoves(στατική για την σωστή ενημέρωση των τιμών της) που δείχνει πόσες δυνατές κινήσεις έχει ο παίχτης.Αν δεν έχει προφανώς πρέπει να πεί πάσο.

Ο κώδικας που χρησιμοποιείται για τον Minimax είναι βασισμένος στον κώδικα που παρουσιάστηκε στο φροντιστήριο του μαθήματος και βρίσκεται στην κλάση Player.

Πιο συγκεκριμένα η τρίτη κλάση Player περιέχει τις μεταβλητές maxDepth που ορίζει το βάθος της αξιολόγησης του Minimax, το playerColor που αναφέρεται στο ποιος παίκτης παίζει ανάλογα με το χρώμα το opponent που αναφέρεται στο αντικείμενο με τα στοιχεία του αντιπάλου.

Ο Minimax δέχεται ως όρισμα έναν πίνακα και η δουλειά του είναι να χρησιμοποιήσει τις μεθόδους max και min σε εναλλαγή για να επιστρέψει την κίνηση που μεγιστοποιεί την αξία κίνησης του υπολογιστή ενώ ταυτόχρονα ελαχιστοποιεί την αξία της κίνησης του αντιπάλου που στη δική μας περίπτωση είναι ο παίκτης-άνθρωπος. Έτσι σε κάθε βήμα μέχρι ο αλγόριθμος να φτάσει σε τελική κατάσταση (δηλαδή να μην υπάρχουν άλλες κινήσεις στο ταμπλό) ή να βρεθεί στο τελικό βάθος αναζήτησης, βρίσκει όλες τις κινήσεις “παιδιά” με την getChildren(States color) που μπορούν να παιχτούν στη δεδομένη στιγμή και τις κινήσεις-απαντήσεις του υπολογιστή. Ο Minimax καλείται μόνο στην περίπτωση του υπολογιστή όπως φαίνεται από τη main στο switch όπου βλέπουμε ότι ο υπολογιστής ελέγχει ποιός έπαιξε τελευταίος (getLastColorPlayed()) και όταν έρθει η σειρά του εκτελεί τον Minimax. Σε περίπτωση ισοβαθμίας ο Minimax επιλέγει τυχαία μία κίνηση με τη χρήση της Random.

Κατά την εκκίνηση του προγράμματος αφού αποφασιστεί ποιος θα παίξει πρώτος, δημιουργούνται τα αντικείμενα της κλάσης Player, ένα για τον παίκτη και ένα για τον υπολογιστή. Για τη δημιουργία τους χρησιμοποιείται ο κατασκευαστής Player (int maxDepth, Tile.States playerColor) ο οποίος δημιουργεί και τον αντίπαλο παίκτη με το αντίθετο χρώμα από το playerColor. Το playerColor χρησιμοποιείται στον Minimax για να έχει ο υπολογιστής μέγιστη αξία στις κινήσεις του (max και playerColor) και ελάχιστη σε αυτές του αντιπάλου (min και opponent.playerColor).

Έπειτα η turn() απλά μας δίνει την ικανότητα να αποφασίσουμε αν θα παίξουμε πρώτοι ή οχι.Άρα αφου πάντα ο πρώτος παίρνει τα μαύρα, ανάλογα την αποφασή μας ορίζονται και τα χρώματα του κάθε παίχτη.Και με την setDepth, ορίζεται το βάθος κινήσεων που θα βλέπει ο minimax.

Παιχνίδι με αντίπαλη ομάδα

Whites

Αρίστος Δημητρίου 3160208

Γιάννης Κουμήδης 3160210

VS

Blacks

Γιώργος Δημόπουλος 3150035

Τρύφωνας Μπιτσάκος 3160108

Γιάννης Παπαγεωργίου 3160129

Πρώτο παιχνίδι με βάθος :1

Whites:40

Blacks:24

Δεύτερο παιχνίδι με βάθος :2

Whites:38

Blacks:36

Τρίτο παιχνίδι με βάθος :3

Whites:30

Blacks:36

Τέταρτο παιχνίδι με βάθος :4

Whites:20

Blacks:46