ΛΕΚΚΑ ΣΤΑΥΡΟΥΛΑ 3090108

Τεχνητή Νοημοσύνη 2020–21
1η Εργασία: Reversi
Αναφορά

ΣΗΜΕΙΩΣΗ: η υλοποίηση βασίζεται στους κανόνες του Othello

Τρόπος Χρήσης:

Κατά την εκκίνηση ο χρήστης βρίσκεται στο κεντρικό μενού.

```
[ O T H E L L O ]

0.Exit
1.New Game
2.Show Rules
>1
```

Οι επιλογές του κεντρικού μενού είναι 3:

έξοδος από το πρόγραμμα (επιλογή 0), νέα παρτίδα (επιλογή 1), υπενθύμιση κανόνων παιχνιδιού (επιλογή 2). Οι επιλογές 0 και 2 είναι βασικές και αυτεξήγητες.

```
------------------------------------------------
                R u l e s
------------------------------------------------
1. Player chooses color for their disks between
dark(x) and light(o).
2. Dark always plays first.
3. Each player must place a piece on the board,
in such a position that there exists at least
one straight occupied line (horizontal, vertical,
or diagonal) of one or more contiguous opponent's
pieces between the new piece and another piece
of the player.
4. The pieces in between are captured and change
sides (color).
5. If one player has no available valid moves,
play passes back to the other player.
6. When neither player can move and/or the board
is full, the game ends.
7. The player with the most pieces on the board
at the end of the game wins.
------------------------------------------------
```

Στην επιλογή 1 (νέο παιχνίδι) ο χρήστης καλείται να πάρει περαιτέρω αποφάσεις. Πρώτα επιλέγει το μέγιστο βάθος αναζήτησης του αλγορίθμου MiniMax πληκτρολογώντας έναν θετικό ακέραιο. Όσο μεγαλύτερο το μέγιστο βάθος τόσο μεγαλύτερος ο βαθμός δυσκολίας. Εμφανίζονται ενδεικτικά μέγιστα βάθη για την ευκολία του χρήστη ενώ ανάλογα με την επιλογή του εμφανίζεται μήνυμα με την επιλεγμένη δυσκολία.

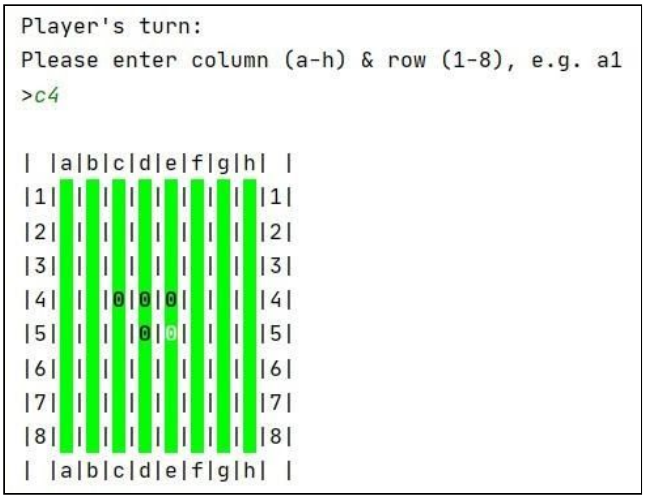
```
[ O T H E L L O ]

0.Exit
1.New Game
2.Show Rules
>1

Insert maximum depth.
Recommended values:
1 for beginners
2 for an easy game
4 for normal gameplay
5 for a challenge
>1
Value set to 1 (Beginner Difficulty)

Select disk color:
1.Dark
2.Light
>1
```

Έπειτα καλείται να επιλέξει το χρώμα των δίσκων του (1 για μαύρα και 2 για λευκά) και άρα κατ' επέκταση το αν θα παίξει πρώτος ή δεύτερος (τα μαύρα παίζουν πάντα πρώτα).

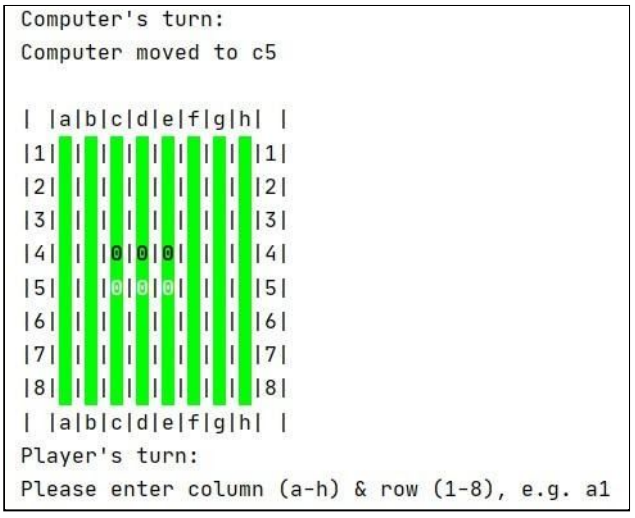
Ακολουθεί η εκκίνηση του παιχνιδιού με εκτύπωση της αρχικής κατάστασης του ταμπλό.

```
-------------------------------------------------
                New Game
-------------------------------------------------

| |a|b|c|d|e|f|g|h| |
|1|  |  |  |  |  |  |  |  |1|
|2|  |  |  |  |  |  |  |  |2|
|3|  |  |  |  |  |  |  |  |3|
|4|  |  |  |O|0|  |  |  |4|
|5|  |  |  |0|O|  |  |  |5|
|6|  |  |  |  |  |  |  |  |6|
|7|  |  |  |  |  |  |  |  |7|
|8|  |  |  |  |  |  |  |  |8|
| |a|b|c|d|e|f|g|h| |

Player's turn:
Please enter column (a-h) & row (1-8), e.g. a1
```

Ανάλογα με την επιλογή χρώματος του παίκτη ξεκινάει ο πρώτος γύρος. Στο παράδειγμα μας ο χρήστης έχει επιλέξει τα μαύρα. Στη σειρά του ο παίκτης καλείται να επιλέξει θέση για το δίσκο του βάζοντας τις συντεταγμένες του πεδίου (γράμμα για στήλη και αριθμό για γραμμή). Εφόσον η θέση είναι έγκυρη εκτυπώνεται η νέα κατάσταση του ταμπλό και είναι πλέον η σειρά του υπολογιστή.



Ο υπολογιστής με τη χρήση του αλγορίθμου MiniMax επιλέγει τη βέλτιστη για εκείνον κίνηση (βλέποντας μπροστά τόσες κινήσεις όσες το μέγιστο βάθος που έχει εισάγει ο χρήστης), εμφανίζεται μήνυμα με την επιλεγμένη κίνηση για ευκολία του χρήστη και τέλος εκτυπώνεται το νέο ταμπλό. Πιο συγκεκριμένα ο αλγόριθμος MiniMax καλεί τη miniMAX για τα μαύρα κομμάτια και τη MINImax για τα λευκά κομμάτια όποιος παίκτης κι αν έχει το κάθε χρώμα.

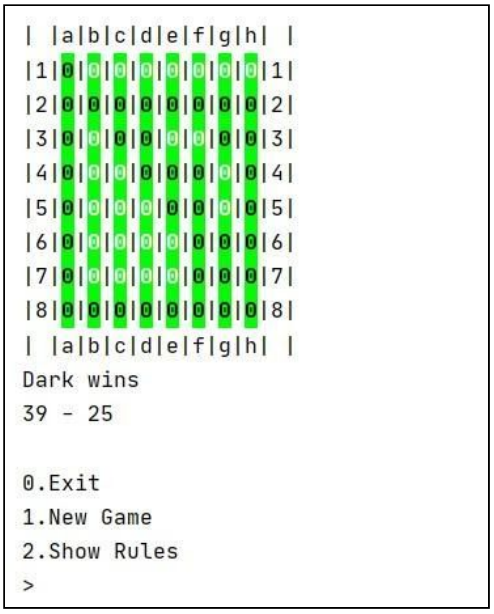


Σε περίπτωση που ο παίκτης επιλέξει μη έγκυρη κίνηση εμφανίζεται αντίστοιχο μήνυμα, εκτυπώνεται ξανά το ταμπλό, αυτή τη φορά με τις δυνατές κινήσεις σημειωμένες με **v** για ευκολία του χρήστη, και το πρόγραμμα περιμένει νέα είσοδο συντεταγμένων, ενώ αν δεν υπάρχουν έγκυρες κινήσεις εμφανίζεται σχετικό μήνυμα και η σειρά περνάει στον αντίθετο παίκτη. Ο έλεγχος για ύπαρξη έγκυρων κινήσεων στο ταμπλό γίνεται μετά από κάθε επιτυχημένη κίνηση.

```
| |a|b|c|d|e|f|g|h| |
|1| | | | | | | | |1|
|2| | | | | | | | |2|
|3| | | | | | | | |3|
|4| | |0|0|0| | | |4|
|5| | |0|0|0|0| | |5|
|6| | | | |0| | | |6|
|7| | | | | | | | |7|
|8| | | | | | | | |8|
| |a|b|c|d|e|f|g|h| |
Player's turn:
Please enter column (a-h) & row (1-8), e.g. a1
>e3
Invalid Move. Please choose another tile.

Hint: Available moves are shown as 'v'
| |a|b|c|d|e|f|g|h| |
|1| | | | | | | | |1|
|2| | | | | | | | |2|
|3| | | | | | | | |3|
|4| | |0|0|0| |v| |4|
|5| | |0|0|0|0| | |5|
|6| |v|v|v|0|v|v| |6|
|7| | | | | | | | |7|
|8| | | | | | | | |8|
| |a|b|c|d|e|f|g|h| |
Please enter column (a-h) & row (1-8), e.g. a1
```

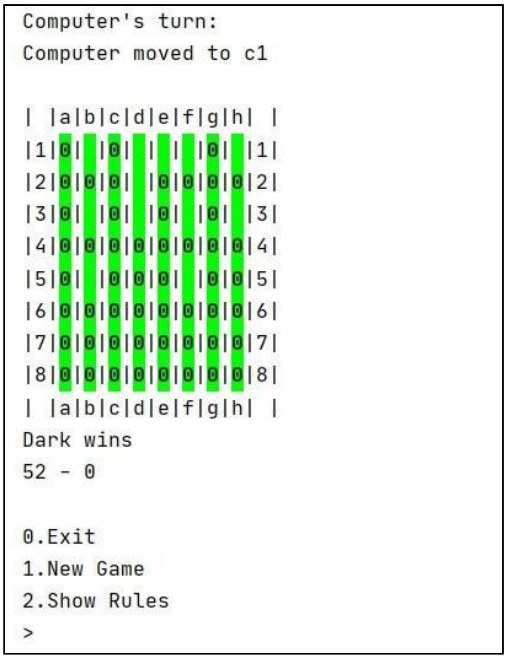
```
Player's turn:
Please enter column (a-h) & row (1-8), e.g. a1
>f8

| |a|b|c|d|e|f|g|h| |
|1| | |0|0|0|0|0|0|1|
|2|0|0|0|0|0|0|0| |2|
|3|0|0|0|0|0|0|0|0|3|
|4|0|0|0|0|0|0|0|0|4|
|5|0|0|0|0|0|0|0|0|5|
|6|0|0|0|0|0|0|0|0|6|
|7|0|0|0|0|0|0|0|0|7|
|8|0|0|0|0|0|0|0|0|8|
| |a|b|c|d|e|f|g|h| |
Computer's turn:
No available moves for computer.

| |a|b|c|d|e|f|g|h| |
|1| | |0|0|0|0|0|0|1|
|2|0|0|0|0|0|0|0| |2|
|3|0|0|0|0|0|0|0|0|3|
|4|0|0|0|0|0|0|0|0|4|
|5|0|0|0|0|0|0|0|0|5|
|6|0|0|0|0|0|0|0|0|6|
|7|0|0|0|0|0|0|0|0|7|
|8|0|0|0|0|0|0|0|0|8|
| |a|b|c|d|e|f|g|h| |
Player's turn:
Please enter column (a-h) & row (1-8), e.g. a1
```

Αν κανείς από τους δύο παίκτες δεν έχει δυνατές κινήσεις ή το ταμπλό είναι γεμάτο το παιχνίδι τελειώνει και ανακηρύσσεται νικητής το χρώμα με το μεγαλύτερο πλήθος κομματιών στο ταμπλό. Αν το πλήθος είναι ίδιο για τα δύο χρώματα το παιχνίδι λήγει σε ισοπαλία. Μετά το τέλος του παιχνιδιού ο χρήστης επιστρέφει στο αρχικό μενού.



Το ταμπλό έχει γεμίσει. Το παιχνίδι λήγει με νικητή τα μαύρα και εμφανίζεται το αρχικό μενού.

Εναλλακτικές καταλήξεις παιχνιδιού:



Υλοποίηση:

Main class

Εδώ βρίσκεται ο κορμός της λειτουργίας του προγράμματος -το βασικό μενού και ο μηχανισμός του παιχνιδιού. Η επιλογή ρυθμίσεων γίνεται με απλή πληκτρολόγηση. Με την επιλογή νέου παιχνιδιού ζητείται η επιλογή μέγιστου βάθους που θα χρησιμοποιηθεί αργότερα στον αλγόριθμο minimax και το χρώμα του χρήστη. Δημιουργείται το ταμπλό και ο παίκτης 'Υπολογιστής' που θα αποφασίζει της κινήσεις του μέσω της minimax. Αν ο χρήστης έχει επιλέξει τα μαύρα ορίζεται ως προηγούμενος παίκτης εικονικά ο υπολογιστής και άρα η πρώτη κίνηση ανήκει στον χρήστη, αλλιώς συμβαίνει το αντίθετο. Όσο το ταμπλό δε βρίσκεται σε τελική κατάσταση, το οποίο ελέγχει η *isTerminal*, εκτελείται η κύρια ροή του παιχνιδιού. Όταν η κατάσταση είναι τελική υπολογίζεται το σκορ, εκτυπώνεται μήνυμα και γίνεται επιστροφή στο κεντρικό μενού.

Στην κεντρική ροή ανάλογα με το είδος του παίκτη εκτελείται διαφορετική διαδικασία. Όταν ο παίκτης που έχει σειρά είναι ο χρήστης κι εφόσον υπάρχουν έγκυρες δυνατές κινήσεις για το χρώμα του, το οποίο ελέγχει η *validMoves*, ζητούνται οι συντεταγμένες της κίνησης που επιθυμεί. Αν η κίνηση που επέλεξε είναι εκτός του ταμπλό ή δεν αποτελεί κάποια από τις προηγουμένως υπολογισμένες δυνατές κινήσεις εκτυπώνεται αντίστοιχο μήνυμα και ζητείται νέα είσοδος μέχρι να δοθεί μια σωστή. Αν οι συντεταγμένες είναι σωστές δημιουργείται η κίνηση και εκτελείται αλλάζοντας την κατάσταση του πίνακα. Όταν ολοκληρωθεί η διαδικασία ή αν ο χρήστης δεν έχει έγκυρες δυνατές κινήσεις δίνεται η σειρά στον αντίπαλο. Όταν ο παίκτης που έχει σειρά είναι ο υπολογιστής και εφόσον υπάρχουν για το χρώμα του δυνατές κινήσεις δημιουργείται μια κίνηση η οποία προκύπτει από την εκτέλεση της *MiniMax και* εκτελείται (*makeMove*). Όταν ολοκληρωθεί η διαδικασία ή αν ο υπολογιστής δεν έχει έγκυρες δυνατές κινήσεις δίνεται η σειρά στον αντίπαλο.

Player class

Εδώ υλοποιείται ο αλγόριθμος minimax που απαιτείται για την επιλογή κίνησης του υπολογιστή. Αρχικά ο *MiniMax* επιλέγει αν θα καλεστεί ο *miniMAX*, που ψάχνει τη μεγαλύτερη αξία και άρα αφορά τα μαύρα κομμάτια, ή ο *MINImax*, που ψάχνει τη μικρότερη αξία και άρα αφορά τα λευκά κομμάτια, βάσει το χρώμα του υπολογιστή. Οι miniMAX και MINImax έχουν ίδια δομή με μόνη διαφορά ότι η πρώτη επιστρέφει το παιδί με τη μεγαλύτερη αξία ενώ η δεύτερη αυτό με τη μικρότερη.

Η δομή έχει ως εξής: Αν η κατάσταση είναι τελική ή έχουμε φτάσει στο μέγιστο βάθος υπολόγισε κι επίστρεψε την κίνηση βάση της ευρετικής. Αν (παρότι δεν είναι τελική κατάσταση) το χρώμα που αντιστοιχεί στη συνάρτηση δεν έχει δυνατές κινήσεις κι άρα όχι παιδιά καλείται η ίδια συνάρτηση (αφού η σειρά ανήκει πάλι στο ίδιο χρώμα). Αλλιώς παράγονται τα παιδιά της κατάστασης και επιλέγεται κι επιστρέφεται το καταλληλότερο αυτών (ανάλογα τη συνάρτηση).

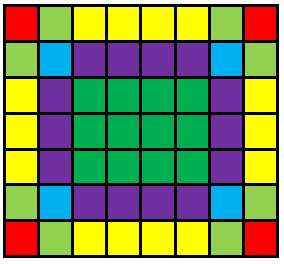
Move class

Εδώ υπάρχουν μόνο constructors, setters και getters που βοηθούν στη διαχείριση της έννοιας των κινήσεων ανάλογα την περίσταση και το σημείο του κώδικά στο οποίο χρειάζονται.

Board class

Εδώ υλοποιείται το μεγαλύτερο μέρος των συναρτήσεων που σχετίζονται με την κατάσταση του ταμπλό ανά πάσα στιγμή. Οι πιο βασικές από αυτές είναι η *validMoves*, που ελέγχει αν υπάρχουν δυνατές κινήσεις για το δοθέν χρώμα με χρήση της *isMoveValid* και δημιουργεί ένα πίνακα-εικόνα του ταμπλό όπου σημαδεύει τα σημεία που βρίσκονται οι κινήσεις αυτές, η *isMoveValid*, που δοθείσης μιας κίνησης ελέγχει αν αυτή υπακούει στους κανόνες του παιχνιδιού, η *getChildren*, που παράγει τα παιδιά της κατάστασης του ταμπλό όπου κάθε παιδί είναι ένα ταμπλό με εκτελεσμένη μια έγκυρη κίνηση, η *evaluate*, που υπολογίζει την αξία του ταμπλό με τρόπο που θα αναλυθεί παρακάτω, η *makeMove*, που τοποθετεί στο ταμπλό το χρώμα του παίκτη που έκανε την κίνηση και εκτελεί όλες τις απαραίτητες αλλαγές που προκύπτουν με μετατροπή των αντίπαλων κομματιών που αιχμαλωτίστηκαν, η *isTerminal*, που ελέγχει αν το ταμπλό βρίσκεται σε τελική κατάσταση όντας γεμάτο ή με μη έγκυρες δυνατές κινήσεις για κανέναν από τους δύο παίκτες, και η hints και print, που εκτυπώνουν το ταμπλό με ή χωρίς βοηθητικές ενδείξεις αντίστοιχα.

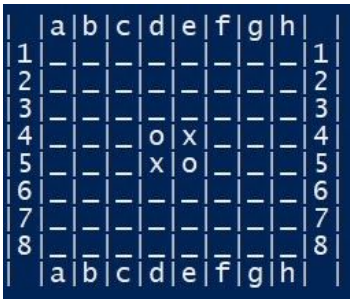
Η *evaluate* υπολογίζει την αξία του ταμπλό προσθέτοντας για κάθε μαύρο κομμάτι και αφαιρώντας για κάθε λευκό. Ανάλογα τη θέση κάθε κομματιού αλλάζει και ο συντελεστής τους καθώς κάποιες θέσεις έχουν μεγαλύτερη στρατηγική σημασία και άρα επηρεάζουν περισσότερο τη ροή του παιχνιδιού. Πιο συγκεκριμένα, σε μια προσπάθεια να γίνει η ευρετική όσο πιο ακριβής γίνεται, οι θέσεις χωρίζονται σε 6 ομάδες. Ο υπολογισμός της αξίας κάθε θέσης προκύπτει από το μέγιστο αριθμό επιθέσεων που μπορεί να κάνει ένας παίκτης τοποθετώντας ένα δίσκο του εκεί (a) συν το διπλάσιο του ελάχιστου αριθμού επιθέσεων από τις οποίες είναι ασφαλής (d) (παίρνοντας την αυθαίρετη σύμβαση ότι η άμυνα είναι σημαντικότερη). Δηλαδή:



	.a=3.d=8.	19
	.a=5.d=6.	17
	.a=3.d=6.	15
	.a=8.d=0.	8
	.a=5.d=0.	5
	.a=3.d=0.	3

Σημείωση:

Τα παραπάνω screenshots τραβήχτηκαν σε IntelliJ με χρήση κωδικών χρωματισμού κειμένου. Λόγω μη αναγνώρισης των κωδικών αυτών από Eclipse και Windows PowerShell η αρχική συνάρτηση χρωματισμού του ταμπλώ μεταποιήθηκε και πλέον αυτό έχει τη μορφή:



Για προβολή της προηγούμενης μορφής σε IntelliJ αρκεί να αλλαχθεί η συνάρτηση colored(char c) στο αρχείο Board.java, line:681, σύμφωνα με τα αντίστοιχα σχόλια.