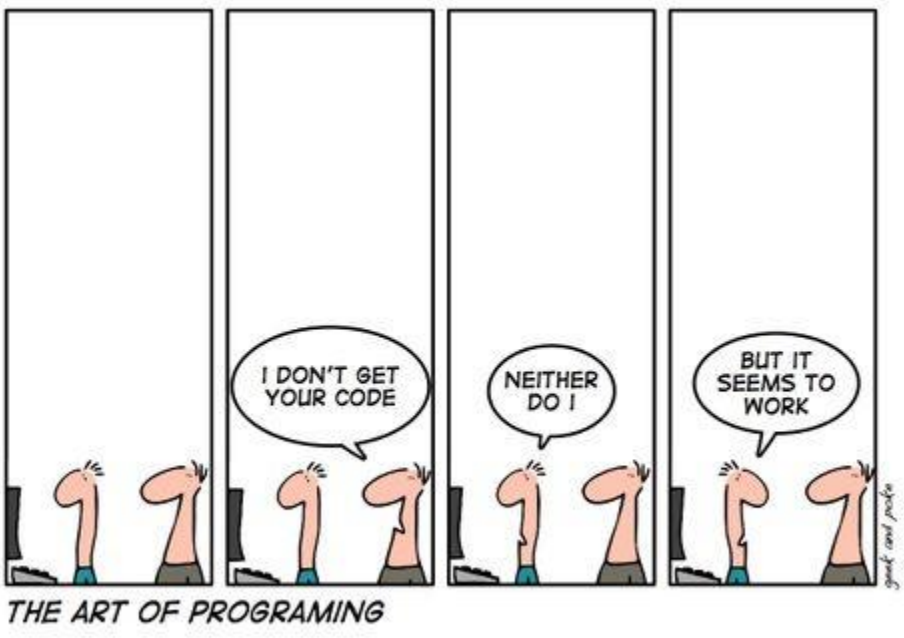


CANDY CRUSH

ΕΥΡΙΠΙΔΗΣ ΜΠΑΛΤΖΗΣ 8196

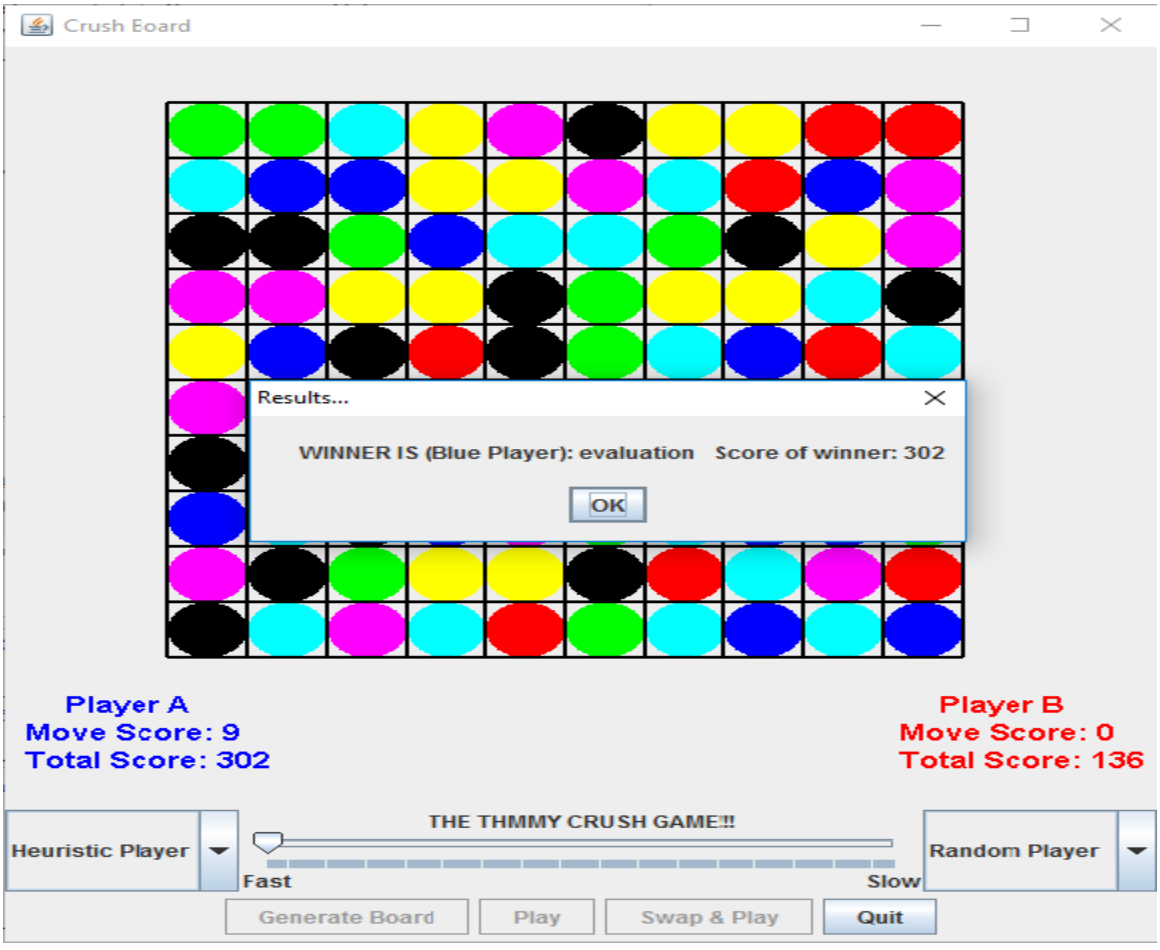
ΒΑΣΙΛΗΣ ΜΠΕΛΛΟΣ 8715

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Στο **δεύτερο** μέρος της εργασίας κληθήκαμε να συμπληρώσουμε την κλάση HeuristicPlayer και συγκεκριμένα τις συναρτήσεις:

- **moveEvaluation**, η οποία θα παρέχει ένα αλγόριθμο αξιολόγησης της προεπιλεγμένης κίνησης ακολουθώντας μια σειρά κριτηρίων ορισμένων από το χρήστη, καθώς και τη
- **findBestMoveIndex**, η οποία θα ελέγχει όλες τις διαθέσιμες κινήσεις, θα τις περνάει από τη συνάρτηση αξιολόγησης και στη συνέχεια θα επιλέγει την καλύτερη.

Μετά την υλοποίση της κλάσης αυτής θα πρέπει να τρέξουμε το παιχνίδι επιλέγοντας τον ένα παίκτη να είναι της μορφής Heuristic Player και τον άλλο να είναι της μορφής Random Player και περιμένουμε ως αποτέλεσμα τη διεξαγωγή ενός παιχνιδιού μεταξύ του παίκτη που δημιουργεί η κλάση που υλοποιήσαμε και του τυχαίου παίκτη, καθώς και τη νίκη του παίκτη που δημιουργήσαμε, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 1: Το περιβάλλον του παιχνιδιού CandyCrush μετά το πέρας ενός επιτυχημένου παιχνιδιού μεταξύ δύο παικτών.

Class HeuristicPlayer

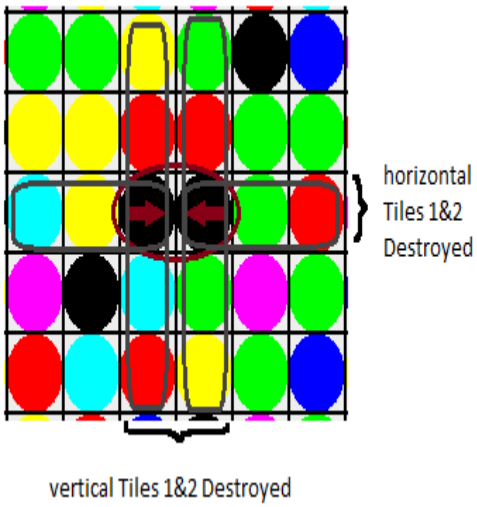
Παρακάτω παρουσιάζεται η υλοποίηση της ζητούμενης κλάσης και περιγράφονται οι μεταβλητές, μέθοδοι και συναρτήσεις που την αποτελούν.

Αρχικά, οι μεταβλητές ενός αντικειμένου της κλάσης αυτές είναι:

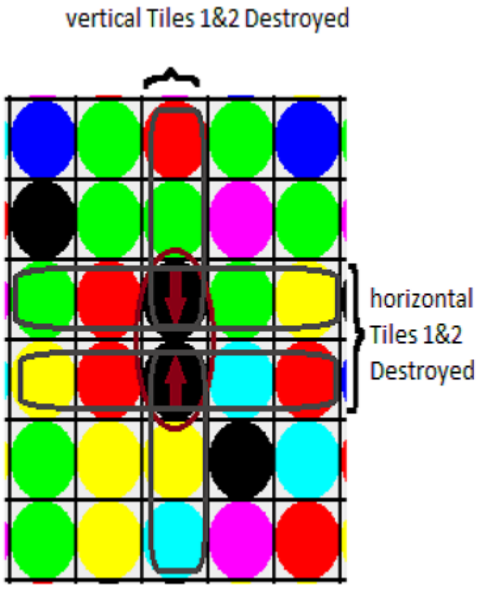
- **int id:** η μεταβλητή αυτή παίρνει τις τιμές 1 ή 2 και καθορίζει αν ο παίκτης που δημιουργήθηκε είναι ο μπλε ή ο κόκκινος.
- **String name:** περιέχει το όνομα του εκάστοτε παίκτη
- **int score:** περιέχει το συνολικό σκορ του παίκτη, το οπολίο προκύπτει από το άθροισμα όλων των τιμών εντός των πλακιδίων που ανήκουν στον παίκτη

Όλες οι συνάρτήσεις **set** και **get** των μεταβλητών αυτών παραμένουν ίδιες με την πρώτη εργασία, όπως και ο αντίστοιχος **constructor HeuristicPlayer(Integer pid)**. Παρακάτω παρουσιάζονται οι προς υλοποίηση συναρτήσεις:

- **Tile[] deletedCandies(Board board, int[] move):** η συνάρτηση αυτή δέχεται σαν όρισμα ένα αντικείμενο της κλάσης Board, δηλ. δέχεται ως όρισμα ολόκληρο το ταμπλό, καθώς και το διάνυσμα (μεγέθους 1x4) move, το οποίο μέσω της κλήσης της συνάρτησης calculateNextMove(move), γεμίζει με τις συντεταγμένες των δύο πλακιδίων που θα αλλάξουν μεταξύ τους θέση κατά τη διάρκεια μιας κίνησης, όπως αυτή επιλέγεται κάθε φορά από τη λίστα διαθέσιμων κινήσεων ArrayList<int[]> availableMoves. Μέσω της στατικής συνάρτησης boardAfterFirstMove της κλάσης CrushUtilities, επιστρέφεται το καινούριο περιεχόμενο του ταμπλό, μετά την αλλαγή των πλακιδίων και πριν την πιθανή διαγραφή πλακιδίων λόγω σχηματισμού συνεχόμενων ν-άδων πλακιδλίων ίδιου χρώματος. Στη συνέχεια, ανάλογα με την κατεύθυνση της αλλαγής, οριζόντιας ή κάθετης, η συνάρτηση υπολογίζει τους γείτονες των πλακιδίων και τους τοποθετεί σε 4 λίστες της μορφής ArrayList<Tile>, ανάλογα με τη θέση τους ως προς τα πλακίδια και όπως φαίνεται στις παρακάτω εικόνες.



Εικόνα 2: Horizontal Swap



Εικόνα 3: Vertival Swap

Στη συνέχεια, ελέγχεται το περιεχόμενο των γειτονικών πλακιδίων ως προς το χρώμα και εάν σχηματίζονται 3άδες, 4άδες ή 5άδες ομόχρωμων πλακιδίων, τότε αυτά θα πρέπει να διαγραφούν από το ταμπλό και κατ' επέκταση τοποθετούνται σε ένα νέο πίνακα τύπου int[numOfDeletedCandies], όπου τοποθετούνται όλα τα προς διαγραφή πλακίδια ανεξαρτήτου θέσης στον πίνακα. Σύμφωνα με τους κανόνες του παιχνιδίου, ο μεγαλύτερος αριθμός πλακιδίων που μπορεί να διαγραφεί με μια κίνηση (εξαιρουμένων των chain moves) είναι 14. Η μεταβλητή numOfDeletedCandies περιέχει το συνολικό αριθμό των πλακιδίων προς διαγραφή μετά από κατάλληλο έλεγχο μέσα στον κώδικα. Τελικά, η συνάρτηση επιστρέφει τον πίνακα αυτόν, ο οποίος είναι ένας πίνακας αντικειμένων της κλάσης Tile.

- **int vertOrHorizOrBoth(int[] move, Tile[] deletedCandies):** η συνάρτηση αυτή δέχεται ως ορίσματα δύο πίνακες, ένα διάνυσμα με integers, το οποίο μέσω της κλήσης της συνάρτησης calculateNextMove(move), γεμίζει με τις συντεταγμένες των δύο πλακιδίων που θα αλλάξουν μεταξύ τους θέση κατά τη διάρκεια μιας κίνησης, καθώς και τον πίνακα deletedCandies με αντικείμενα της κλάσης Tile, όπως αυτός δημιουργήθηκε στην παραπάνω συνάρτηση Tile[] deletedCandies. Στη συνέχεια, υπολογίζεται μέσω 2 μεταβλητών τύπου boolean, εάν κατά την επιλεγμένη κίνηση προέκυψε διαγραφή πλακιδίων κατά τον οριζόντιο ή/και κάθετο άξονα του ταμπλό και επιστρέφεται ένας αριθμός (0=καμία διαγραφή, 1=οριζόντια διαγραφή μόνο, 2=κάθετη διαγραφή μόνο, 3=οριζόντια και κάθετη διαγραφή) στην καλούσα συνάρτηση.

- **Int heightOfMove(Tile[] deletedCandies):** η συνάρτηση αυτή δέχεται ως όρισμα τον πίνακα deletedCandies με αντικείμενα της κλάσης Tile, όπως αυτός δημιουργήθηκε στην παραπάνω συνάρτηση Tile[] deletedCandies. Στη συνέχεια, διασχίζει όλο το length του πίνακα deletedCandies και βρίσκει εκείνο το πλακίδιο (προς διαγραφή) με τη μικρότερη τεταγμένη, την οποία και επιστρέφει στην καλούσα συνάρτηση. Το έυρος των επιστρεφόμενων τιμών κυμαίνεται από 0 έως 9.

- **int moveEvaluation(Board board, int[] move)**: η συνάρτηση αυτή δέχεται σαν όρισμα ένα αντικείμενο της κλάσης Board, δηλ. δέχεται ως όρισμα ολόκληρο το ταμπλό, καθώς και το διάνυσμα (μεγέθους 1x4) move, το οποίο μέσω της κλήσης της συνάρτησης calculateNextMove(move), γεμίζει με τις συντεταγμένες των δύο πλακιδίων που θα αλλάξουν μεταξύ τους θέση κατά τη διάρκεια μιας κίνησης, όπως αυτή επιλέγεται κάθε φορά από τη λίστα διαθέσιμων κινήσεων ArrayList<int[]> availableMoves. Στη συνέχεια, καλούνται οι συναρτήσεις deletedCandies, heightOfMove και vertOrHorizOrBoth και οι επιστρεφόμενες τιμές αποθηκέυονται σε αντίστοιχες μεταβλητές. Όσο πιο χαμηλά στο ταμπλό πραγματοποιείται η κίνηση, τόσο μεγαλύτερη η πιθανότητα δημιουργίας αλυσιδωτής αντίδρασης και κατ' επέκταση τόσο μεγαλύτερη η πιθανότητα διαγραφής περισσότερων πλακιδίων εξαιτίας της. Ωστόσο, σε περιπτωση που δεν υπάρξει αλυσιδωτή αντίδραση, αυξάνεται η πιθανότητα διαυκόλυνσης του αντιπάλου μέσω της μεγαλύτερης ανανέωσης του ταμπλό μέσω κάποιας χαμηλής διαγραφής. Όσον αφορά την κατεύθυνση της διαγραφής, αν πρόκειται για διαγραφή και στις 2 κατευθύνσεις (return 3), τότε η υπεροχή της εκδηλώνεται και από τον αριθμό των πλακιδίων που θα διαγραφούν (μέσω της deletedCandies). Στην περίπτωση οριζόντιας διαγραφής (return 2), τα πλακίδια πάνω από τη γραμμή διαγραφής και εντός των τετμημένων των πλακιδίων που διαγράφονται, θα μετακίνηθουν κατά μια θέση προς τα κάτω. Στην περίπτωση κάθετης διαγραφής (return 1), στη στήλη που θα γίνει η διαγραφή, τα πλακίδια που βρίσκονται πιο ψηλά από εκείνα που πρόκειται να διαγραφούν, θα μετακινηθούν τόσες θέσεις προς τα κάτω, όσος είναι και ο συνολικός αριθμός των πλακιδίων που θα αφαιρεθουν. Έτσι το εύρος των τιμών που επιστρέφει η vertOrHorizOrBoth κυμαίνεται από 0 έως 3. Με δεδομένο ότι όσο χαμηλότερα στο ταμπλό συμβούν οι διαγραφές, τόσο μεγαλύτερες οι πιθανότητες για chain move, ο συντελεστής για τη συνάρτηση heightOfMove θα είναι αρνητικός. Με δεδομένο το εύρος των επιστρεφόμενων τιμών της κάθε συνάρτησης και το ότι στην οι μέγιστες τιμές που μπορούν να πάρουν ταυτόχρτονα τα αξιολογητικά κριτήρια είναι deletedCandies = 14, vertOrHorizOrBoth = 3 και heightOfMove = 4 (για κάθετη διαγραφή 5 πλακιδίων), οι αντλίστοιχοι τους συντελεστές για αποτέλεσμα = 100 είναι (8,8,-9). Τέλος, επιστρέφεται η τιμή της αξιολόγησης που έγινε με βάση την παραπάνω εξίσωση.

- **int findBestMoveIndex(Board board, ArrayList<int[]> availableMoves)**: η συνάρτηση αυτή δέχεται σαν όρισμα ένα αντικείμενο της κλάσης Board, δηλ. δέχεται ως όρισμα ολόκληρο το ταμπλό, και το δυναμικό πίνακα availableMoves, που περιέχει το σύνολο των διαθέσιμων κινήσεων του παίκτη. Στη συνέχεια, σαρώνει ολόκληρο τον πίνακα των διαθέσιμων κινήσεων, περνά την κάθε κίνηση μέσα από την παραπάνω συνάρτηση αξιολόγησης και κρατάει το αποτέλεσμά της. Έπειτα, ο συμπληρωμένος πλέον πίνακας evals σαρώνεται με σκοπό την εύρεση της μέγιστης τιμής του, η οποία είναι και η καλύτερη αξιολόγηση, συνεπώς και η καλύτερη διαθέσιμη κίνηση την εκάστοτε στιγμή. Τελικά, η συνάρτηση επιστρέφει το δείκτη της θέσης του πίνακα που περιέχει αυτή τη μέγιστη αξιολόγηση.