Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστραφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2022-2023

Ά ΦΑΣΗ PROJECT

Περιγραφή του δρόμου που θα ακολουθήσουμε στην προγραμματιστική άσκηση της Java, υλοποιώντας το Stratego

Φώτης Πελαντάκης

4988

21 Νοεμβρίου 2022

Περιεχόμενα

1. Εισαγωγή

Στην εργασία αυτή καλούμαστε να υλοποιήσουμε μία παραλλαγή του κλασικού παιχνιδιού Stratego με τίτλο Stratego Ice vs Fire. Το Stratego είναι ένα επιτραπέζιο παιχνίδι στρατηγικής το οποίο έκανε τα πρώτα του βήματα στο μακρινό 1946. Περισσότερες πληροφορίες για αυτό μπορείτε να βρείτε στον παρακάτω σύνδεσμο

Information about Stratego

Στην υλοποίηση του project θα εφαρμόσουμε την τεχνική MVC (Model - View - Controller) επειδή με αυτόν τον τρόπο είναι πιο εύκολη η γραφή του κώδικα, αλλά και την συντήρησή του. Γι' αυτόν τον λόγο η υλοποίηση θα χωριστεί σε τρία μέρη:

- Το "Model", στο οποίο υλοποιούμε την λειτουργικότητα του παιχνιδιού, στην προκειμένη περίπτωση, το πως δουλεύουν τα πιόνια, οι μάχες κ.α.
- Το "View" στο οποίο υλοποιείται το οπτικό μέρος του project, δηλαδη το UI.
- Τέλος το "Controller" το οποίο γεφυρώνει τα "View" και "Model".

Σε αυτήν την αναφορά θα ασχοληθούμε με τον τρόπο που θα χρησιμοποιήσουμε το MVC για την εκπόνηση του project μας.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

- ***Classes για τα πιόνια του παιχνιδιού (Pieces)***

Η παρακάτω κλάση χρησιμοποιείται για να δώσει χαρακτηριστικά στους παίκτες, τα οποία τα μοιράζονται και οι δύο (π.χ. το turn δηλαδή αν είναι η σειρά κάποιου)

public abstract Player

```
private int aixmalotiseis;
private int diasoseis;
private int round;  //γύρος που βρισκόμαστε
private int success_attacks;     //πλήθος επιτυχ. επιθέσεων
private int attacks;         //πλήθος επιθέσεων
private float percent_success_attack;  //ποσοστό επιτυχημένων επιθέσεων
public Player(boolean turn)  //constructor
//παρακάτω είναι setters και getters των αντίστοιχων τιμών
float getPercent_success_attack(int success_attacks, int attacks)
int getSuccess_attack()
void setDiasoseis(int diasoseis)
int getDiasoseis()
void setAixmalotiseis(int aixmalotiseis)
int getAixmalotiseis()
```

Η κλάση Player1 "κληρονομεί" τα χαρακτηριστικά της Player στον 1ο παίκτη

public class Player1 extends Player

```
public Player1(boolean turn)  //καλεί την υπερκλάση Player
```

Η κλάση Player2 "κληρονομεί" τα χαρακτηριστικά της Player στον 2ο παίκτη

public class Player2 extends Player

public Player2(boolean turn) // καλεί την υπερκλάση Player

Moveable pieces

abstract class Piece

Η κλάση αυτή περιέχει σημαντικές πληροφορίες που μοιράζονται όλες οι κάρτες του παιχνιδιού.

Περιέχει τα εξής πεδία:

private static int *power*; //πόντοι κάθε κάρτας (στην σημαία και στην παγίδα έχουμε οι τιμές -2 και -1 αντίστοιχα για διευκόλυνση στον κώδικα

private int diasoseis; //διασώσεις που έχουν πραγματοποιηθεί για το συγκεκριμένο κομμάτι

private int player;   //κάτοχος κομματιού

private int position;   //θέση κομματιού

και τις εξής μεθόδους:

public Piece(int power, int player, int position) //constructor

public void setDiasoseis(int diasoseis) //δίνει τιμή στις διασώσεις

public void setPlayer(int player) //δίνει κάτοχο στο κομμάτι

public int getplayer() //επιστρέφει τον παίκτη που το έχει (1 ή 2)

public void set_position(int position) //θέτει θέση στο κομμάτι

public int get_position() //επιστρέφει την θέση του

Η συγκεκριμένη κλάση όπως και οι υπόλοιπες αντίστοιχες της ενότητας αυτής συμπληρώνει ένα κομμάτι (π.χ. εδώ το BeastRider) με κάποια συγκεκριμένα έξτρα χαρακτηριστικά και μεθόδους που μόνο αυτό το κομμάτι μπορεί να διαθέτει (π.χ. ξεχωριστή συμπεριφορά στην κίνηση)

public class BeastRider extends Piece

public BeastRider(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει το BeastRider είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Dragon extends Piece

public Dragon(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει ο Dragon είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Elf extends Piece

public Elf(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει το Elf είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Knight extends Piece

public Knight(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του Knight είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class LavaBeast extends Piece

public LavaBeast(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του LavaBeast είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Mage extends Piece

public Mage(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(intcurrent_position,int next_position) //εξετάζει να η κίνηση που κάνει του LavaBeast είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Scout extends Piece

public Scout(int player, int position) //καλεί την υπερκλάση Piece.

boolean den_diaperna_tin_apagorevmeni_perioxi (int current_position, int new_position) //εξετάζει ότι κατά την κίνηση δεν διαπερνάται η απαγορευμένη περιοχή

boolean ValidMove (int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του Scout είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Slayer extends Piece

public Slayer(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του Slayer είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Sorceress extends Piece

public Sorceress(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του Sorceress είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

public class Yeti extends Piece

public Yeti(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position, int next_position) //εξετάζει να η κίνηση που κάνει του Yeti είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

Moveable pieces ---> Special Moveable Piece

public class Dwarf extends Piece

public Dwarf(int player, int position) //καλεί την υπερκλάση Piece.

boolean ValidMove(int current_position,int next_position) //εξετάζει να η κίνηση που κάνει του Dwarf είναι έγκυρη και επιστρέφει την αντίστοιχη τιμή.

Immovable pieces

public class Flag extends Piece

private static int *power*; // πόντοι = -2

private int player; //κάτοχος σημαίας

private int position; //θέση σημαίας

public Flag(int player, int position) //καλεί την υπερκλάση Piece.

public class Trap extends Piece

private static int *power*; //πόντοι = -1

private int player;   //κάτοχος σημαίας

private int position; //θέση σημαίας

public Trap(int player, int position)   //καλεί την υπερκλάση Piece.

Η συγκεκριμένη κλάση δημιουργεί έναν πίνακα brd 10x8 ο οποίος δείχνει αν ένα κελί είναι κενό ή όχι. Για παράδειγμα αν μπεί ένα κομμάτι στο [2,3] το brd[2*10 + 3 - 1] θα γίνει true αν παραμείνει κενό τότε θα είναι false

public class Board

public static boolean *brd*[]; //πίνακας boolean

public Board(boolean[] brd) // τον γεμίζει αρχικά με false

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Η συγκεκριμένη κλάση έχει σκοπό να βρίσκει το αποτέλεσμα της επίθεσης

public class Attack

static int result(int power_defender, int power_attacker) //επιστρέφει το αποτέλεσμα της μάχης

Η συγκεκριμένη κλάση δίνει μορφή στα διάσπαρτα τμήματα που φτιάχναμε μέχρι τώρα. Συγκεκριμένα είναι υπεύθυνη για το πότε θα λήξει το παιχνίδι, τον τρόπο που θα αρχικοποιθούν όλα τα στοιχεία του προγράμματος κατά την εκκίνηση του παιχνιδιού, όπως και συνδέει το Model με το κομμάτι View.

public class Controller

public static boolean *original_rules*;

public static ArrayList<Piece> *pieces_ingame_p1*;

public static ArrayList<Piece> *pieces_ingame_p2*;

public static ArrayList<Piece> *pieces_out_of_the_game_p1*;

public static ArrayList<Piece> *pieces_out_of_the_game_p2*;

public static Player1 *p1*;

public static Player2 *p2*;

public Controller() //Constructor

void StartGame() //αρχικοποίηση του παιχνιδιού

public static boolean end_game() //ελέγχει εάν ήρθε το τέλος

Σε αυτήν την κλάση φτιάχνουμε την κίνηση των κομματιών, την επίθεση, την διάσωση και τα buttons

class Move implements ActionListener

public Move() //Constructor

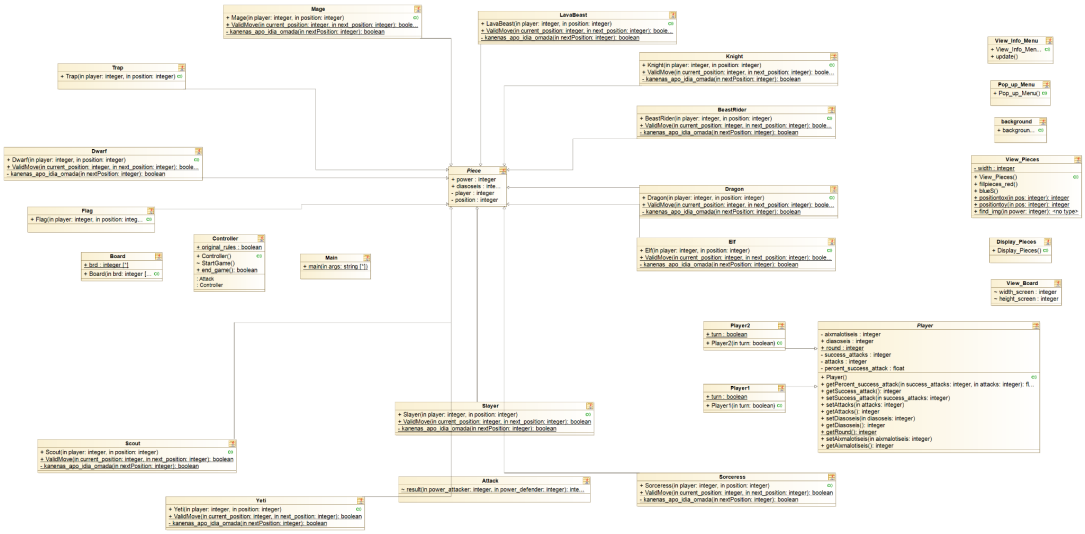
```
void Diasosi()    //ζητάει ποιο κομμάτι θα διασωθεί και την θέση που θα τοποθετηθεί

public void actionPerformed(ActionEvent e) //ελέγχει τα κουμπιά
```

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Για το View τις εξής κλάσεις:

- class Display_Pieces extends JFrame το οποίο θέτει τα όρια του κεντρικού παραθύρου (με το board)και θα τοποθετεί τα κομμάτια στα κατάλληλα κελιά (καλώντας την View_Pieces).
- Την **class Pop_up_Menu** η οποία σχηματίζει το pop up menu στην αρχή κάθε παιχνιδιού που δίνει την δυνατότητα στον χρήστη να επιλέξει αν θα παίξει το παιχνίδι με του πρότυπους κανόνες ή την εναλλακτική εκδοχή (μειωμένος στρατός). Αυτό περιλαμβάνει checkbox. Όταν είναι "τικαρισμένο" θα ενεργοποιείται η αντίστοιχη λειτουργία. Επίσης θα έχει ένα button "Save" που θα οριστικοποιεί την επιλογή του χρήστη και θα κλείνει το menu.
- Την **class View_Board** που θα δίνει μορφή με χρήση GUI στην board. Αυτή θα περιλαμβάνει ένα panel και μέσα σε αυτό θα σχεδιάζει το board.
- Την **class View_Pieces** που θα παίρνει τις εικόνες από τον φάκελο με τις εικόνες των κομματιών και θα τις βάζει στις κατάλληλες θέσεις στον board. Αυτή θα αποτελείται από το panel του board και θα βάζει μέσα τις εικόνες. Επίσης θα έχει μια μέθοδο "refresh" που θα ανανεώνει τις θέσεις των κομματιών μετά από κάθε γύρο.
- Την **class View_Info_Menu** η οποία θα είναι ένα JMenu και θα εκτυπώνει στις κατάλληλες θέσεις το αντίστοιχο κείμενο (π.χ. Αιχμαλωτίσεις) και θα έχει checkboxes τα οποία θα εκφράζουν το αποτέλεσμα του Pop_up_menu και τις εικόνες κάτω από τις αιχμαλωτίσεις αυτών που έχουν αιχμαλωτιστεί. Επίσης θα έχει μια μέθοδο "refresh" που ανανεώνει το μενού.

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML



6. Λειτουργικότητα (Β Φάση)

Η εργασία με βοήθησε να έχω μια πιο ολοκληρωμένη εικόνα όχι μόνο για την Java, αλλά και για την ανάπτυξη μιας εφαρμογής αφού ήταν η πρώτη φορά που ασχοληθήκαμε με την σύνδεση πολλών μερών του προγράμματος για να δημιουργηθεί το ενιαίο. Παρόλα αυτά θα ήθελα να δινόντουσαν περισσότερες πληροφορίες για τα buttons και πως να τα χρησιμοποιήσουμε.

7. Συμπεράσματα

Η εργασία είναι πολύ χρήσιμη και ενδιαφέρουσα αλλά προσωπικά δυσκολεύτηκα αρκετά να μάθω πως να χειρίζομαι στα buttons στο πρόγραμμα. Κατα τα άλλα η εκφώνηση ήταν πολύ καλή, το θέμα ενδιαφέρον και το αποτέλεσμα πολύ εποικοδομητικό.