



Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζιτζικας

Χειμερινό Εξάμηνο 2020-2021

SORRY!

Εισαγωγή

Βίκτωρας Σφακιανάκης

5085

13/1/2024

Περιεχόμενα

1. Εισαγωγή.....	2
2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model	2
3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller.....	9
4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View.....	11
5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML.....	16
6. Λειτουργικότητα (B Φάση).....	17
7. Συμπεράσματα	18

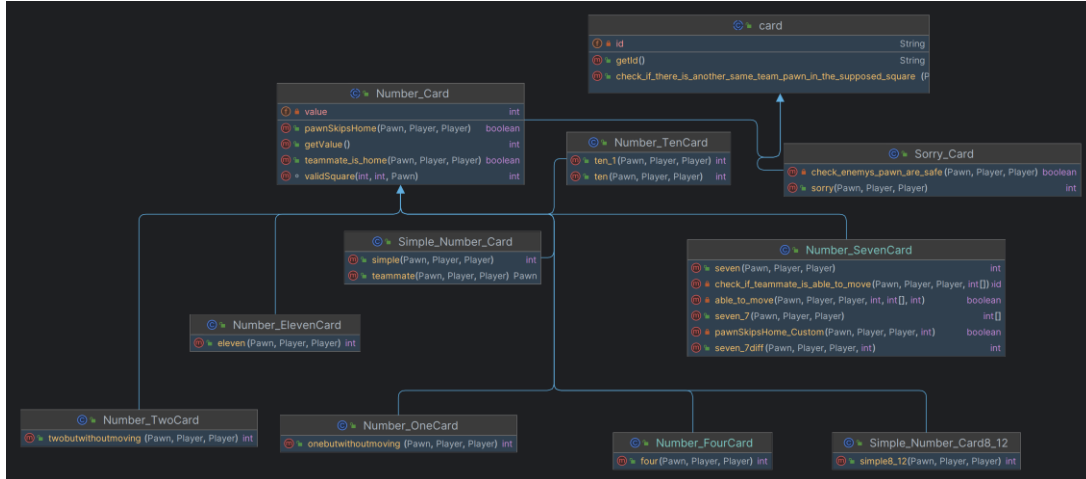
1. Εισαγωγή

Για να λειτουργικότητα του project χρησιμοποίησα την αρχιτεκτονική MVC (Model ,View ,Controller). Το παραπάνω υλοποιεί ένα είδος συνεργασίας μεταξύ του πυρήνα του παιχνιδιού (controller), των μοντέλων όπως τα πιόνια (model), και των γραφικών (view). Το project αυτό υλοποιεί το επιτραπέζιο παιχνίδι Sorry. Το παιχνίδι αυτό κανονικά παίζεται με 4 παίκτες και έχει 4 πιόνια ο κάθε παίκτης ενώ σε αυτή την έκδοση παίζουν 2 παίκτες ο καθένας παίζει με 2 πιόνια. Σκοπός του παιχνιδιού είναι να καταφέρουν οι παίκτες να μεταφέρουν και τα δυο του πιόνια στο Home square , ο πρώτος κερδίζει και το παιχνίδι τελειώνει.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Το πακέτο Model αποτελείται από 4 διαφορετικά πακέτα και την κλάση Board . Τα πακέτα είναι τα εξής

- Card
- Player
- Square
- Turn
- Board



- abstract card class
 - Attribute : private String id
 - Methods : public String getId() ,
public boolean
check_if_there_is_another_same_team_pawn_in_the_supposed_square
- abstract class Number_Card extends card
 - Attribute : private int value

- Methods : public int getValue //every cards has its value (e.g. Card 1 value = 1 , Card 2 value = 2)
 - , public boolean teammate_is_home, public boolean pawnSkipsHome , public int validSquare
- public class Number_OneCard extends Number_Card
 - Method:


```
public int one(Pawn pawn, Player player1, Player player2) // return the position of the pawn one space forward or gets it out of start
```
- public class Number_TwoCard extends Number_Card
 - Method:


```
public int two(Pawn pawn, Player player1, Player player2) // return the position of the pawn two spaces forward or gets it out of start
```
- public class Simple_Number_Card extends Number_Car
 - Method :


```
public int simple(Pawn pawn, Player player1, Player player2) // return the position of the pawn three or five spaces forward depends the value
```
- public class Number_FourCard extends Number_Card
 - Method : public int four(Pawn pawn, Player player1, Player player2)// return the position of the pawn 4 spaces forwards if possible
- public class Number_SevenCard extends Number_Card
 - Methods :
 1. public int seven(Pawn pawn, Player player1, Player player2) // return the position of the pawn seven spaces forward
 2. public int[] seven_7(Pawn pawn, Player player1, Player player2)//return all the possible move that the pawn has in the range [1,7]
 3. private void check_if_teammate_is_able_to_move(Pawn p, Player player1, Player player2, int[] arr) // check if the other pawn of the same player won't be able to move after the first pawn takes the position that is in the array so it will cancel it
 4. private boolean able_to_move(Pawn pawn2, Player player1, Player player2, int k, int[] arr, int i)// helps the previous function
 5. private boolean pawnSkipsHome_Custom(Pawn pawn, Player player1, Player player2, int value) // custom version of pawnSkipsHome that already exist but this one gets the value via parameter and not depends in getvalue()
 6. public int seven_7diff(Pawn pawn, Player player1, Player player2, int value) // return the position of the pawn 7-value spaces forward
- public class Simple_Number_Card8_12 extends Number_Card
 - Method:


```
public int simple8_12(Pawn pawn, Player player1, Player player2) // return the position of the pawn eight or twelve spaces forward
```
- public class Number_TenCard extends Number_Card

- Methods:
 1. `public int ten(Pawn pawn , Player player1 , Player player2)`
 `//return the position of the pawn ten spaces forward`
 2. `public int ten_1(Pawn pawn , Player player1 , Player player2)`
 `//return the position of the pawn one space backward`
- `public class Number_ElevenCard extends Number_Card`
 - Method:
 `public int eleven(Pawn pawn , Player player1 , Player player2)`
 `//return the position of the pawn eleven spaces forward`
- `public final class Sorry_Card extends card`
 - Methods:
 1. `public int sorry(Pawn pawn, Player player1, Player player2)//check in pawn in Start position (if not cancel the move)`
 `+ call the second function to check position of the enemy pawns`
 2. `private boolean check_enemys_pawn_are_safe(Pawn pawn, Player player1, Player player2) //return true if both enemy pawn are safe (if true cancel the card)`

Στο πακέτο card υπάρχουν κλάσεις για το κάθε είδος κάρτας που υπάρχει στο deck μαζί με κάποιες abstract κλάσεις τις card και Number_Card . Οι περισσότερες κάρτες έχουν την δική τους προσωπική κλάση ενώ υπάρχουν 2 και 2 κάρτες που μοιράζονται τις ίδιες κλασεις γιατί έχουν τις ίδιες ιδιότητες. Αυτές είναι η 3 με την 5 που μεταφέρουν και τα δυο πιόνια 3 η 5 θέσεις μπροστά ανάλογα (αν δεν γίνεται και τα δυο τότε μετακινείται μόνο το ένα πιόνι) . Και οι άλλες δυο είναι η 8 με την 12 που είτε δίνουν σε ένα πιόνι την ιδιότητα να μεταπειθεί 8 ή 12 θέσεις μπροστά είτε να τραβήξει ο παίκτης νέα κάρτα . Οι κλασεις σε αυτο το πακέτο προβλέπουν την μελλοντικά ενδεχομένη θέση του πιονιού αλλά δεν του τη δίνουν για ο παίκτης μπορεί να αλλάξει γνώμη για το πιο πιόνι θέλει να διαλέξει . Επιπλέον βλέπουν αν το πιονιού είναι ικανό να μετακινηθεί ώστε να ενεργοποιηθεί το fold button αν κανένα απο τα δυο δεν μπορεί. Η κάθε κλάση έχει τις δίκες τις συναρτήσεις ενώ υπάρχουν 4 συναρτισεις που τις μοιράζονται όλες οι κάρτες και βρίσκονται στις δυο abstract κλασεις .

1. Η μια είναι η boolean `check_if_there_is_another_same_team_pawn_in_the_supposed_square` η οποία ελέγχει αν υπάρχει άλλο πιόνι ιδίου χρώματος σε εκείνη τη θέση που το πιόνι θα πάει , αν ναι τότε επιστέφει true και στη συνέχεια η κλάση της κάρτα κάνει την `moveable` του pawn false και δεν επιστέφει την τωρινή θέση (δηλαδή δεν μπορεί να κινηθεί).
2. Η άλλη συνάρτηση είναι η boolean `teammate_is_home` οπού γίνεται true όταν το ένα από τα πιόνια του παίκτη που παίζει είναι στο Home . Αυτό γίνεται γιατί αυτή είναι η μοναδική περίπτωση που τα δυο πιόνια μπορούν να βρίσκονται στο ίδιο τετράγωνο.
3. Η τρίτη είναι η boolean `pawnSkipsHome` επιστέφει true αν με την κάρτα που δόθηκε στο παίκτη ξεπερνιέται το Home ,τότε το η τιμή του πιονιού `movable` γίνεται false και επιστέφεται η δεδομένη θέση άρα δεν μπορεί να κινηθεί.

4. Η τελευταία είναι η int validSquare που χρησιμοποιείτε για τις κάρτες 10 (στην -1 περίπτωση) και 4 που επιστρέφει 1 ή 4 θέσεις πίσω από το πόνι χρησιμοποιείτε για να δοθεί το σωστό square στην παραπάνω συνάρτηση
check_if_there_is_another_same_team_pawn_in_the_supposed_square

(οι Λειτουργίες των καρτών που λείπουν αναλαμβάνονται στο View)

Player	Pawn
Player(String, int, int, int, Pawn, Pawn)	Pawn(int, String, int)
color String	isHome Boolean
startPosition int	isStart boolean
turn boolean	id int
Pawn1 Pawn	position int
homePosition int	isSafe boolean
SafetyZoneStartPosition int	color String
nextofstartPosition int	moveable boolean
Pawn2 Pawn	
getTurn() boolean	setSafe(boolean) void
getStartPosition() int	isStart() boolean
getPawn2() Pawn	getPosition() int
setNextofstartPosition(int) void	setHome(Boolean) void
getPawn1() Pawn	setMoveable(boolean) void
getColor() String	getId() int
setTurn(boolean) void	isSafe() boolean
getNextafterstartPosition() int	getHome() Boolean
setColor(String) void	isMoveable() boolean
getHomePosition() int	setPosition(int) void
setHomePosition(int) void	setStart(boolean) void
	getColor() String

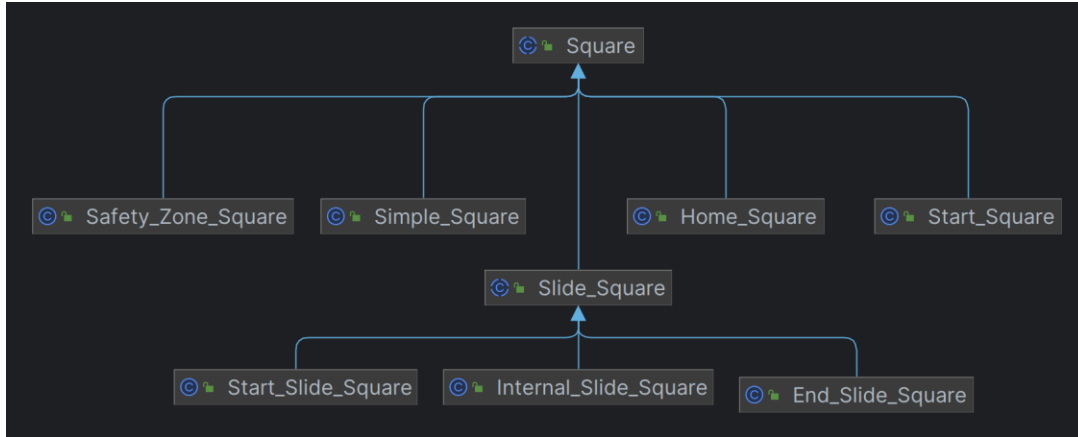
Στο Player υπάρχουν δυο κλάσεις η Player και η Pawn

- Player
 - Attributes:
 1. private String color //color of the player
 2. private int homePosition // home position of the player
 3. private int SafetyZoneStartPosition // the position that safezone starts
 4. private int startPosition // start position of the player
 5. private int nextofstartPosition //the next position after the start
 6. private boolean turn // true if its the turn of the player else false
 7. private Pawn Pawn1 // players pawn1
 8. private Pawn Pawn2 // players pawn2
 - Methods:
 1. public Pawn getPawn1()
 2. public Pawn getPawn2()
 3. public boolean getTurn()
 4. public void setTurn()
 5. public String getColor()
 6. public int getHomePosition()
 7. public int getStartPosition()
 8. public int getNextafterstartPosition()

Η player αποτελείται από χρήσιμες πληροφορίες για τον παίκτη το χρώμα , την θέση του start ,την θέση του home και την θέση μετά το start γιατί ο πίνακας είναι μονοδιάστατος και αρχίζει από πάνω αριστερά άρα για τον κόκκινο η θέση μετά το start είναι 10 ενώ για τον κίτρινο η 46. Περιέχει ακόμα και δυο objects από την κλάση Pawn

- Pawn
 - Attributes:
 1. private int position; //the position of the pawn
 2. private boolean isSafe; // true if the pawn is in safezone
 3. private Boolean isHome; //true if pawn is Home
 4. private boolean isStart; //true if pawn is in Start
 5. private String color; // the color of the pawn
 6. private boolean moveable; // true if the can move else false
 7. private int id; // id of the pawn , pawn1 has id 1 and pawn2 has 2
 - Methods:
 1. public int getId()
 2. public int getPosition()
 3. public void setPosition(int position) // this sets almost every boolean variable that the pawn has (isHome , isSafe ,isStart) and sets the position too
 4. public String getColor()
 5. public boolean isSafe()
 6. public void setSafe(boolean safe)
 7. public Boolean getHome()
 8. public boolean isMoveable()
 9. public boolean isStart()
 10. public void setStart(boolean start)
 11. public void setMoveable(boolean moveable)

Η κλάση Pawn περιέχει το χρώμα του πιονιού την θέση του στο board και 4 Boolean μεταβλητές την isSafe, την isHome , την isStart και την moveable . Οι τρεις πρώτες είναι true αν το πόνι βρίσκεται σε Safezone ή στο Home position ή στο Start position του χρώματος του και η moveable παίρνει την τιμή true όταν μπορεί μετακινηθεί εκεί που του λέει η κάρτα αν δεν μπορεί τότε είναι false.



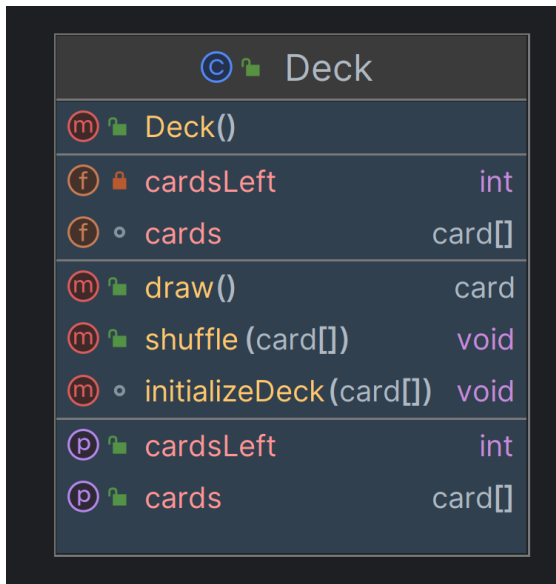
Η κλάση Square περιέχει όλα τα είδη square σε κλάσεις που υπάρχουν στο board του παιχνιδιού μαζί με 2 abstract κλάσεις Square και Slide_Square

Όλες οι κλάσεις είναι οι:

- Square
- Safety_Zone_Square
- Simple_Square
- Home_Square
- Start_Square
- Slide_Square
- Start_Slide_Square
- Internal_Slide_Square
- End_Slide_Square

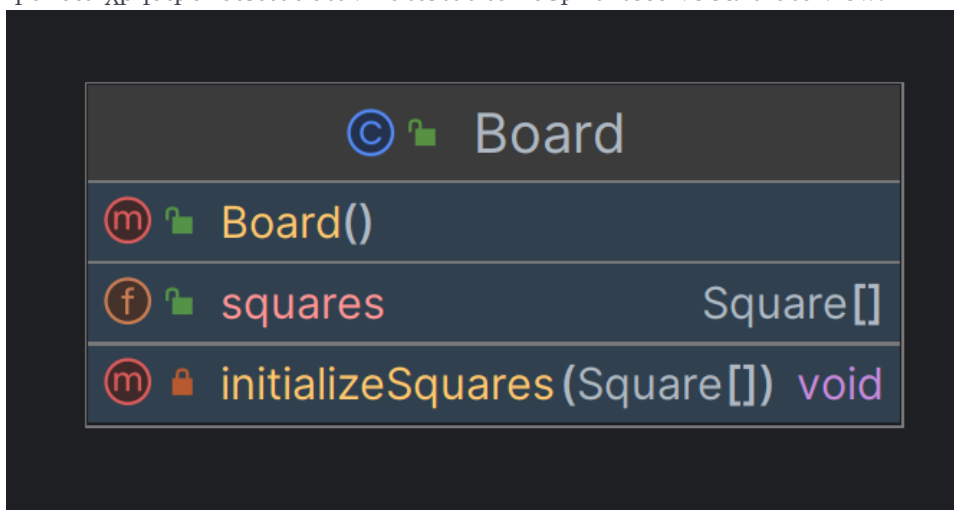
Οι οποίες αποθηκεύονται στον παράλληλο πίνακα board που διαχειρίζεται ο Controller για να ξέρει το Pawn σε τι είδους θέση βρίσκεται πάνω στον πίνακα και να δρα ανάλογα.

Αυτές οι κλάσεις δεν περιλαμβάνουν συγκεκριμένες συναρτήσεις, ωστόσο, η χρήση του instanceof τους είναι χρήσιμη για τη λειτουργία του προγράμματος .



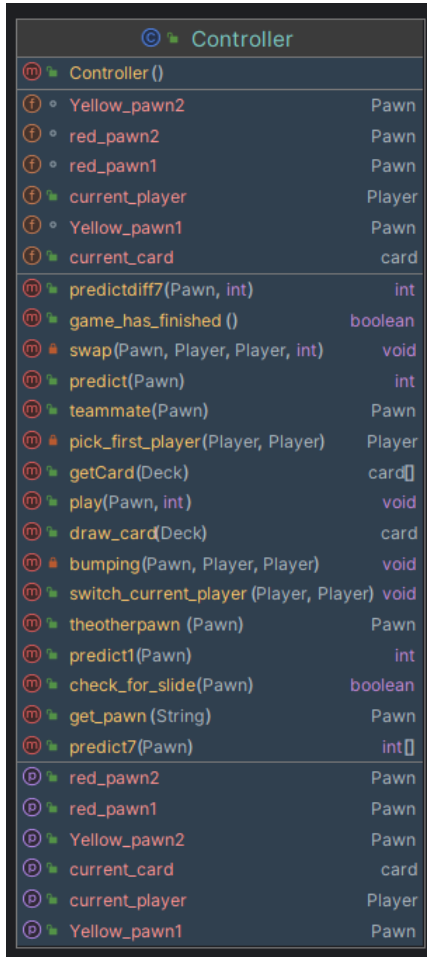
- Deck
 - Attributes:
 1. private int cardsLeft;
 2. card[] cards= new card[44];
 - Methods:
 1. public card[] getCards()
 2. private void initializeDeck(card[] cards)
 3. private void shuffle(card[] cards)
 4. public card draw()
 5. public int getCardsLeft()

Το πακέτο Turn περιέχει μόνο την κλάση Deck η οποία είναι υπεύθυνη για την αρχικοποίηση του πίνακα deck που και την ανάμιξη του ενώ περιέχει και την μέθοδο draw η οποία χρησιμοποιείται όταν πατιέται το κουμπι ReceiveCard στο view.



Η κλάση Board φτιάχνει ένα μονοδιάστατο πίνακα 72 θέσεων που γίνεται initialize στον Constructor του από όλες τις κλάσεις που κάνουν extend την Square. Είναι ο παράλληλος πίνακας που δεν βλέπει ο χρήστης αλλά είναι πολύ χρήσιμος για την ομαλή λειτουργία των πιονιών , χρησιμοποιείται στον Controller .

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller



Controller	
Controller ()	
Yellow_pawn2	Pawn
red_pawn2	Pawn
red_pawn1	Pawn
current_player	Player
Yellow_pawn1	Pawn
current_card	card
predictdiff7(Pawn, int)	int
game_has_finished ()	boolean
swap(Pawn, Player, Player, int)	void
predict(Pawn)	int
teammate(Pawn)	Pawn
pick_first_player(Player, Player)	Player
getCard(Deck)	card[]
play(Pawn, int)	void
draw_card(Deck)	card
bumping(Pawn, Player, Player)	void
switch_current_player (Player, Player)	void
theotherpawn (Pawn)	Pawn
predict1(Pawn)	int
check_for_slide(Pawn)	boolean
get_pawn (String)	Pawn
predict7(Pawn)	int[]
red_pawn2	Pawn
red_pawn1	Pawn
Yellow_pawn2	Pawn
current_card	card
current_player	Player
Yellow_pawn1	Pawn

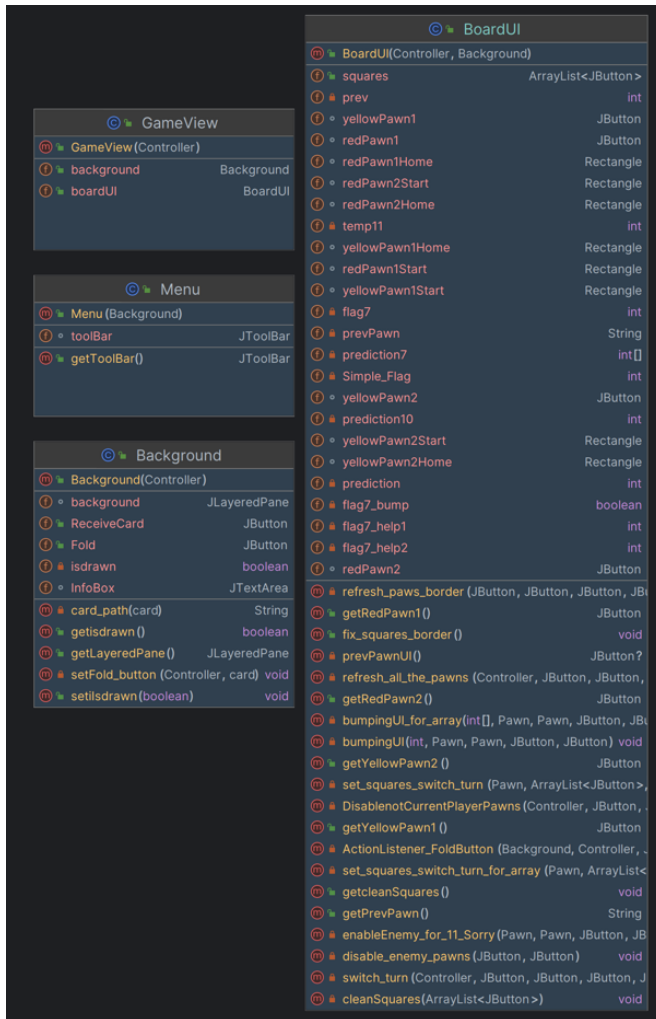
- Controller
 - Attributes:
 1. public Board board ;
 2. public Deck deck ;
 3. public Player current_player ;
 4. Pawn red_pawn1;
 5. Pawn red_pawn2 ;
 6. Pawn Yellow_pawn1;
 7. Pawn Yellow_pawn2 ;
 8. public Player player1 ;
 9. public Player player2 ;
 10. public card current_card ;

- Methods:
 1. public Pawn getRed_pawn1()
 2. public Pawn getRed_pawn2()
 3. public Pawn getYellow_pawn1()
 4. public Pawn getYellow_pawn2()
 5. private Player pick_first_player(Player player1, Player player2)
//pick randomly the first player and return it
 6. public void switch_current_player (Player player1, Player player2)
 7. public boolean game_has_finished() //return true if game has finished
 8. public card draw_card(Deck deck)// call draw from Deck and sets the Current_card too
 9. public int predict1(Pawn pthe) //predict for card 10 – 1 move backward
 10. public int[] predict7(Pawn p) // predict for card 7 – return the array of possible moves
 11. public int predictdiff7(Pawn p, int value) // the predict for the second pawn after the previous predict7
 12. public int predict(Pawn p) //the main predict which get used from all the cards
 13. public void play(Pawn p , int prediction)//sets the position of the pawn p to the prediction and checks for colizations (except card 11)
 14. private void swap(Pawn p, Player player1, Player player2, int mypawnpos)// swap is used in card 11 to swap the position of the pawns
 15. private void bumping(Pawn p, Player player1, Player player2)
//handles the colizations
 16. public boolean check_for_slide(Pawn p) // if p in the start of the opposite color slide it sends it to the end meanwhile sets every pawn that is in the middle their start
 17. public Pawn theotherpawn(Pawn p) // return the other same color pawn
 18. public Pawn get_pawn(String s) // gets a String s with name of the pawn and return the actual pawn
 19. public Pawn teammate(Pawn pawn2) // return the other same color pawn

Αρχικά ο constructor του Controller καλεί τους default constructors των κλάσεων Board και Deck για να αρχικοποιηθεί ο πίνακας και οι τράπουλα στη συνέχεια δημιουργείτε ένα αντικείμενο για κάθε πόνι και ένα για κάθε παίκτη με τις απαραίτητες αρχικές πληροφορίες .Έπειτα κατά τη διάρκεια τις λειτουργίας του προγράμματος μέσω των predic συναρτήσεων ο controller είναι υπεύθυνος να βρει και να καλέσει την συνάρτηση από το πακέτο card . Όλες οι κάρτες αναλαμβάνονται κύριος από το **predict** ενώ αυτές οι οποίες δίνουν στο παίκτη πάνω από μια επιλογές στο που να παίξει χρειάζονται ακόμα μια βιοηθική predic για παράδειγμα στην κάρτα 10 δίνεται η επιλογή στον παίκτη να

κουνήσει το πιόνι του 10 θέσεις μπροστά ή μια θέση πίσω τότε η predict επιστρέφει τις 10 θέσεις μπροστά και η βιοηθική **predict1** επιστρέφει την μια πίσω .Ενώ στην κάρτα 7 υπάρχει η ιδιότητα να μοιραστούν οι 7 κινήσεις στα 2 πιόνια ,αν το ένα δεν μπορεί να κινηθεί το άλλο πάει αναγκαστικά 7 θέσεις μπροστά με την predict αλλά όταν μπορούν και τα δυο πιόνια να κινηθούν τότε η **predict7** επιστρέφει ένα πίνακα 7 θέσεων με όλες τις valid θέσεις που μπορεί το πιόνι να πάει στην συνέχεια το turn δεν αλλάζει και ο παίκτης πρέπει να παίξει τις υπόλοιπες (αν υπάρχουν) κινήσεις στο άλλο πιόνι καλώντας την συνάρτηση **predictdiff7**. Για την 11 χρησιμοποιείται μονο η predict που επιστρέφει τις 11 θέσεις μπροστά .Όσο αναφορά τις ιδιότητες καρτών που έχουν να κάνουν με πιόνια αντίπαλου που είναι η 11 και η sorry την δουλειά του predict την αναλαμβάνει το View στέλνοντας στην συνάρτηση play την θέση του αντίστοιχου αντίπαλου πονιού .Οι predict συναρτήσεις καθορίζουν επίσης αν το πιόνι τελικά μπορεί να κινηθεί αλλάζοντας την boolean τιμή που περιχέει η κλάση Pawn 'movable' η οποία χρησιμοποιείται για την ενεργοποίηση και εμφάνιση του fold button όταν και στα δυο πιόνια είναι false. Στη συνέχεια όταν ο παίκτης αποφασίσει πιο πιόνι θα διαλέξει και σε πια θέση να πάει δηλαδή όταν πατηθεί ένα τετράγωνο στο board η το πιόνι του αντίπαλου αν υπάρχει σύγκρουση τότε καλείται η συνάρτηση **play** η οποία βάζει το πιόνι σε αυτή τη θέση και μέσω της bumping που καλείται στην play ελέγχει για τυχόν συγκρούσεις που επηρεάζουν τα πιόνια του αντίπαλου και δρα ανάλογα . Επίσης ο Controller περιέχει ακόμα την συνάρτηση check_for_slide που είναι υπεύθυνη για την άρτια υλοποίηση των slides στο board σύμφωνα με τους κανόνες του παιχνιδιού . Οι switch_current_player και draw_card (τραβάει μια κάρτα από το deck και αλλάζει την current_card) χρησιμοποιούνται συνήθως όταν αλλάζει η σειρά μεταξύ των παικτών αλλά και σε άλλες περιπτώσεις αναλόγως την κάρτα. Οι υπόλοιπες συναρτήσεις είναι απλά βοηθητικές και χρησιμοποιούνται από κάποιες από τις παραπάνω για παράδειγμα η teammate(pawn) επιστρέφει το άλλο πιόνι του ίδιου player .

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View



Το πακέτο view αρχικά αποτελείται από 4 κλήσεις τις:

- `GameView`
 - Attributes:
 1. `public Background background`
 2. `public BoardUI boardUI`

Η οποία καλεί και ενώνει τις τρεις παρακάτω κλάσεις

- `Background`
 - Attributes:
 1. `JLayeredPane background`
 2. `JTextArea InfoBox`
 3. `public JButton ReceiveCard`
 4. `public JButton Fold`
 5. `private boolean isdrawn ; //true if card in drawn`
 - Methods
 1. Constructor : `Background(Controller controller)`

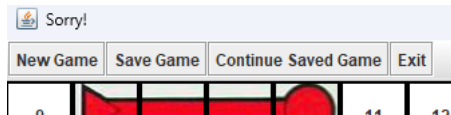
2. `private void setFold_button(Controller controller, card currentCard)`
3. `private String card_path(card s) //returns the path of the image of the card into a string`
4. `public boolean getisdrawn()`
5. `public void setisdrawn(boolean isdrawn)`
6. `public JLayeredPane getLayeredPane()`

Η οποία δημιουργεί το πράσινο background , το κουμπι-κάρτα ReceiveCard μαζί με τον ActionListener της και την εικόνα CurrentCard , τοποθετεί επίσης και τα labels τους ακριβώς κάτω από τις κάρτες, ενώ φτιάχνει και τοποθετεί το Fold button που ο ActionListener του βρίσκεται στη κλάση BoardUI.

Ο ActionListener του ReceiveCard καλεί την συνάρτηση draw_card() του controller ώστε να έρθει καινούρια κάρτα , ενημερώνει το Info_Box και εκτελεί την συνάρτηση setFold_button που αναφέρω αμέσως μετά.

Η συμπεριφορά του fold button εξαρτάτε από της θέσεις των πιονιών του παίκτη που έχει σειρά , και την κάρτα θα έρθει γι' αυτό το status λαμβάνεται όταν πατηθεί το κουμπι ReceiceCard .Αυτό γίνεται καλώντας την private συνάρτηση setFold_button που στη συνέχεια καλεί τις predict συναρτήσεις του controller και για τα δυο πόνια ,αν και τα δυο γίνουν unmovable τότε το Button Fold εμφανίζεται ενεργοποιημένο γιατί ο παίκτης δεν έχει άλλη επιλογή παρά να το πατήσει (εξαιρώντας την κάρτα 11 που σε κάμπες περιπτώσεις μπορεί να παίξει αλλά το fold είναι ανοιχτό) .

- Menu
 - Attribute
 1. `JToolBar toolBar;`
 - Method
 1. `public JToolBar getToolBar()`



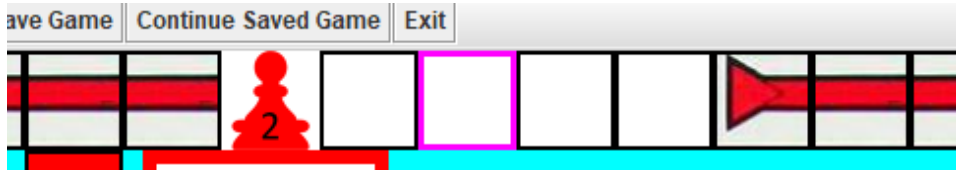
Η οποία περιέχει τα τέσσερα αυτά κουμπιά από τα οποία το New Game κάνει reset το παιχνίδι το δυο επόμενα δεν κάνουν τίποτα και το exit κλίνει την καρτέλα . Το menu τοποθετείτε πάνω από το background στην κλάση GameView.

- BoardUI
 - Attribute
 1. `private int prev // previous position of the pawn`
 2. `private int Simple_Flag = 0`
 3. `private String prevPawn //holds the name of the pawn the has been clicked , useful in bumping`
 4. `private int prediction //hold the prediction that came from predict`
 5. `private int prediction10 = -1 // when card then in drawn this take the 1 move backward prediction`
 6. `private int[] prediction7 = new int[7]`
 7. `private int flag7 = -1`
 8. `private int flag7_help1`

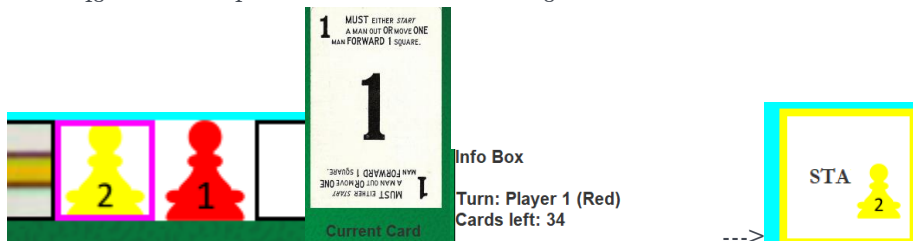
9. private int flag7_help2
10. private int temp11 // used to swap the pawn in card 11
11. private boolean flag7_bump = false
12. public static ArrayList<JButton> squares
13. static JButton redPawn1
14. static JButton redPawn2
15. static JButton yellowPawn1
16. static JButton yellowPawn2
17. Rectangle redPawn1Start;
18. Rectangle redPawn2Start;
19. Rectangle yellowPawn1Start;
20. Rectangle yellowPawn2Start;
21. Rectangle redPawn1Home;
22. Rectangle redPawn2Home;
23. Rectangle yellowPawn1Home;
24. Rectangle yellowPawn2Home;
- Methods
 1. private void disable_enemy_pawns(JButton enemyPawnUI1, JButton enemyPawnUI2)
 2. public static void fix_squares_border()
 3. private JButton prevPawnUI()
 4. private static void cleanSquares
 5. public static void getcleanSquares()
 6. private void bumpingUI_for_array(int[] prediction7, Pawn enemyPawn1, Pawn enemyPawn2, JButton enemyPawnUI1, JButton enemyPawnUI2)
 7. private void enableEnemy_for_11_Sorry(Pawn enemyPawn1, Pawn enemyPawn2, JButton enemyPawnUI1, JButton enemyPawnUI2)
 8. private void bumpingUI(int prediction, Pawn enemyPawn1, Pawn enemyPawn2, JButton enemyPawnUI1, JButton enemyPawnUI2)
 9. private void set_squares_switch_turn_for_array(Pawn pawn, ArrayList<JButton> squares, JButton pawnUI, Controller controller, JButton redPawn1, JButton redPawn2, JButton yellowPawn1, JButton yellowPawn2, Background background, int[] prediction7, int prev, Rectangle Home)
 10. private void set_squares_switch_turn(Pawn pawn, ArrayList<JButton> squares, JButton pawnUI, Controller controller, JButton redPawn1, JButton redPawn2, JButton yellowPawn1, JButton yellowPawn2, Background background, int prediction, int prev, Rectangle Home)
 11. private void refresh_all_the_pawns(Controller controller, JButton redPawn1, JButton redPawn2, JButton yellowPawn1, JButton yellowPawn2, ArrayList<JButton> squares)

12. private void ActionListener_FoldButton(Background background, Controller controller, JButton redPaw1, JButton redPaw2, JButton yellowPaw1, JButton yellowPaw2)
13. private void refresh_paws_border(JButton redPaw1, JButton redPaw2, JButton yellowPaw1, JButton yellowPaw2)
14. private void DisablenotCurrentPlayerPawns(Controller controller, JButton redPaw1, JButton redPaw2, JButton yellowPaw1, JButton yellowPaw2)
15. private void switch_turn(Controller controller, JButton redPaw1, JButton redPaw2, JButton yellowPaw1, JButton yellowPaw2, Background background)

Η κλάση αυτή είναι υπεύθυνη για όλες τις λειτουργίες των πιονιών και του πίνακα που βλέπει ο χρήστης. Περιέχει τέσσερεις όμοιους ActionListeners για τα καθένα πiónι και 47 ActionListeners για κάθε τετράγωνο του πίνακα. Τα πiónια του παίκτη που δεν είναι η σειρά του είναι disabled αλλά ο παίκτης μπορεί να πατήσει μόνο τα δικά του. Μετά που θα τραβήξει την κάρτα, θα πατήσει το πiónι που θέλει να μετακινήσει τότε ο ActionListener του πιονιού θα ενεργοποιηθεί και θα καλέσει την συνάρτηση predict, ανάλογα την κάρτα που ήρθε θα επιστραφεί η θέση του πίνακα που το πiónι μπορεί να πάει, στο τετράγωνο αυτό θα αλλάξει χρώμα το boarder του ώστε να είναι πιο εύκολο για τον χρήστη να το βρει και μην χρειάζεται μετράει κάθε φορά σε πιο τετράγωνο να πάει.



Όλοι οι υπόλοιποι ActionListeners των τετράγωνων εκτός από αυτών που το πiónι μπορεί να πάει πετάνε το μήνυμα "You can't move there". Όταν ο χρήστης πατήσει το έγκυρο τετράγωνο τότε θα καλεστεί η συνάρτηση play του controller και θα αλλάξει η θέση του πιονιού. Στη συνέχεια θα εκτελεστούν οι απαραίτητες εντολές ώστε το πiónι να μετακινηθεί και στα γραφικά. Στην περίπτωση το πiónι θα πέσει πάνω σε άλλο πiónι αλλού παίκτη τότε υπάρχει η συνάρτηση bumpingUI η οποία συγκρίνει την θέση που θα πάει το πiónι με τις θέσεις των αντίπαλων πιονιών και τα ενεργοποιεί ώστε να μπορεί ο παίκτης να τα πατήσει και να πάει πάνω τους.



Αυτό πετυχαίνεται με τη χρήση της εντολής if (ο παίκτης που έχει σειρά είναι ο αντίπαλος) τότε το πiónι πατήθηκε για αυτό το λόγο και βρισκόμαστε σε αυτήν την περίπτωση. Όταν μπαίνει σε αυτήν την if το πiónι πρέπει να συμπεριφερθεί σαν square και επιπλέον να το στείλει και στο start του. Από την στιγμή που το πiónι λειτουργεί σαν square καλεί την play του controller που αναλαμβάνει να τοποθετήσει σε σωστή θέση

και στα δυο πιόνια με την set position για το κόκκινο (στο πάνω παράδειγμα) και την bumping (η bumping βρίσκει το πιόνι και του αλλάζει την θέση του στο start) για το κίτρινο.

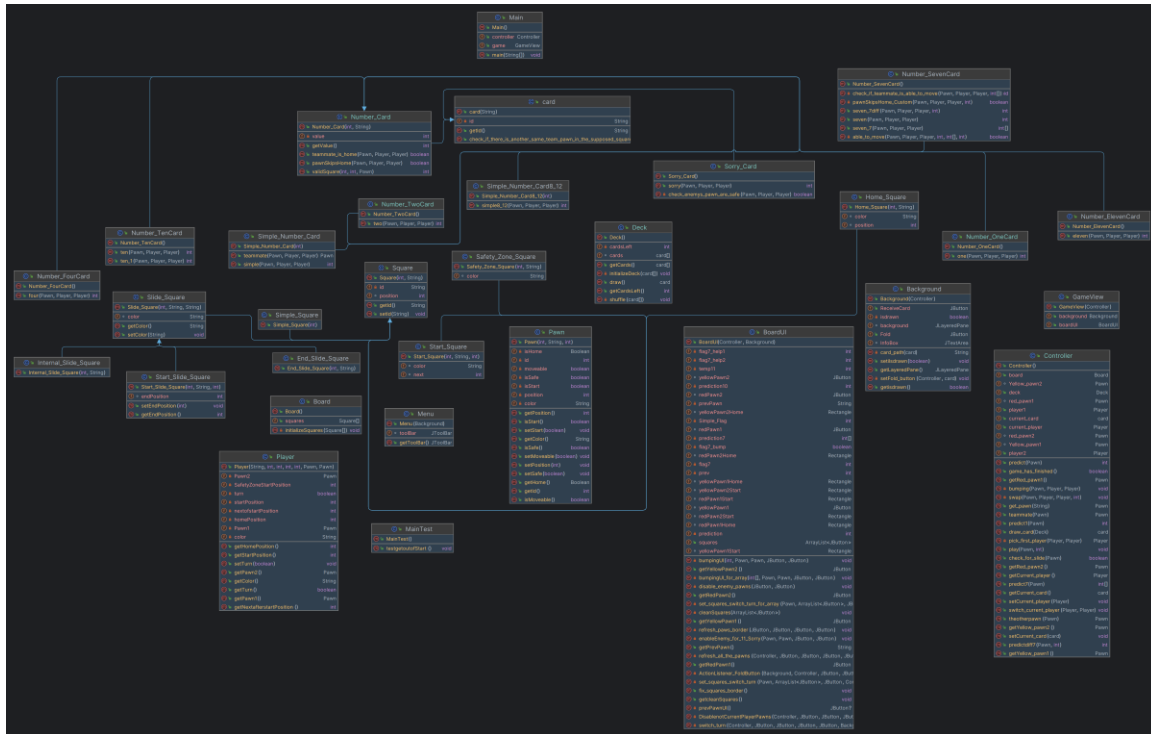
```
public void play(Pawn p,int prediction){
    card temp=getCurrent_card() ;
    if(temp instanceof Number_OneCard){
        p.setPosition(prediction);
        bumping(p,player1,player2);
    }
}
```

(Στις φωτογραφίες παραπάνω το p είναι το κόκκινο και μετά όπιο αντίπαλο πιόνι βρίσκεται στην ίδιο θέση του αλλάζει την θέση στο start)

Το ποια κάρτα θα έχει επίδραση σε διαφορά μέρη την κλάση .

- Η 2 δεν αλλάζει το turn απλά ανοίγει το κουμπι ReceiveCard για να ξανά τραβήξει ο ίδιος παίκτης
- Η 3 και η 5 λειτουργούν με flag που αρχίζει με 0 και όταν γίνει 2 αλλάζει το turn αυτο επιτρέπει και στα δυο πιόνια να κινηθούν.
- Η 7 είναι σαν την 3 και την 5 απλά το flag λειτουργεί διαφορετικά αρχίζει με -1 και στη συνέχεια παίρνει την διαφορά του 7 με τις κινήσεις που έκανε το πρώτο πιόνι για να παίξει το άλλο πιόνι τις υπόλοιπες .
- Η 8 και η 12 ανοίγουν το κουμπι ReceiveCard για να μπορεί να ξανά διαλέξει κάρτα
- Η 10 ανοίγει και την επιλογή να πάει το πιόνι και πίσω πέρα από το +10
- Η 11 και sorry ενεργοποιούν τα αντίπαλα πιόνια ανάλογα την θέση όλων των πιονιών.

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML



Αυτο είναι το UML του src και συνοπτικά ο τρόπος που επικοινωνούν τα τρία πακέτα (MVC) μεταξύ είναι η main καλεί κατευθείαν τον constructor του controller και στη συνέχεια τον περνά σαν παράμετρο σε όλες τις κλάσεις του view εκτός την menu . Στη συνέχεια ανάλογα την κάρτα τα κουμπιά των πιονιών συμπεριφέρονται ανάλογα και καλούν την συνάρτηση του controller αναλόγως τη χρειάζονται , και με την σειρά του ο controller ανάλογα την συνάρτηση είτε καλεί μια συνάρτηση από το model είτε αναλαμβάνει μόνος του.

6. Λειτουργικότητα (Β Φάση)

Έκανα όλα τα ερώτημα χωρίς τα Bonus , προσπάθησα να το κάνω όσο πιο user friendly γίνετε αφαιρώντας το fold button όποτε δεν γίνετε να το πατήσει και δείχνοντας του όλες τις πιθανές κινήσεις που μπορεί να κάνει , θεώρησα πιο λογικό στα slides να στέλνει το rawn κατευθείαν στο τέλος χωρίς να χρειάζεται δεύτερο click από το χρήστη στο τέλος του slide. Ο τρόπος που δουλεύει το παιχνίδι είναι ο χρήστης πατάει την κάρτα μετά πατάει το πόνι και μετά πατάει το τετράγωνο στο board για να μετακινηθεί. Βέβαια επειδή τα edge cases είναι άπειρα μπορεί να μην έχω καταφέρει να τα περιορίσω όλα αλλά όσα βρήκα παίζοντας διορθώθηκαν αρκετά εύκολα.

7. Συμπεράσματα

Δεν συνάντησα κάποιο πρόβλημα που δεν μπορούσα να λύσω, αλλά επειδή κάνω πρώτη φορά άσκηση χωρίς αναλυτικές οδηγίες βήμα - βήμα τη πρέπει να κάνω, έκανα κάποια 'εμπειρικά' λάθη τα οποία με οδήγησαν σε παραπάνω προβλήματα και έλεγχους τα οποία αντιμετωπίστηκαν, ένα από αυτά ήταν ότι έκανα πρώτα το πινάκα στα γραφικά manually (βάση τους αριθμούς του μονοδιάστατου πινάκα `ArrayList<JButton> squares` μετά όταν τον έκανα στην κλάση `board` τον ξανά έκανα με τον ίδιο τρόπο, θα μπορούσα να τον κάνω πρώτα στη κλάση `board` και μετά να τον χρησιμοποιήσω για τα γραφικά ώστε αν γίνει κάποιο λάθος (που έγινε) να χρειαστεί να τον αλλάξω μόνο στο `board`. Το άλλο ήταν ότι άρχισα τον πινάκα από πάνω αριστερά ενώ θα ήταν προτιμότερο να αρχίσει από κάτω αριστερά η πάνω δεξιά μακριά από τα Homes για να μην χρειάζεται για να γίνονται ταυτόχρονα ελέγχει για `out of bounds exception` και τους έλεγχους που ήθελε το παιχνίδι από μόνο του. Θεωρώ πως αυτό με βελτίωσε ώστε να επόμενη φορά να προβλέπω πιο εύκολα τέτοιες λεπτομέρειες. Η κάρτα 7 ήταν πολύ πιο δύσκολη από όλες και θα έπρεπε να βαθμολογείται παραπάνω.