Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2019-2020

[Sorry!Board Game]

[Θεοδώρα Σαμπάλου]

[4306]

[15/1/2019]

Περιεχόμενα

1. Εισαγωγή 1
2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model 1
3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller 1
4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View 2
5. Λειτουργικότητα (Β Φάση) 2
6. Συμπεράσματα 2

- ## Εισαγωγή

Η υλοποίηση της εργασίας θα βασιστεί πάνω στο μοντέλο MVC (Model View Controller).Έτσι , σκοπός μας είναι ο Controller να είναι ο συνδετικός κρίκος των Model και view. Οπότε στη συνέχεια της αναφοράς μας θα αναλύσουμε λίγο ιδιαίτερα τα κομμάτια του Model και Controller που είναι σημαντικά για αυτή τη φάση και τέλος θα αναφερθούμε και λίγο στο view.

- ## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Σε αυτό το πακέτο θα περιέχονται διεπαφή Card,οι κλάσεις NumberCard, SorryCard και κλάσεις που κληρονομούν την NumberCard, οι SimpleNumberCard,NumberOneCard,NumberTwoCard,NumberFourCard,NumberSevenCard,NumberTenCard,NumberElevenCard.Επίσης έχουμε την κλάση Square και τις υποκλάσεις της, StartSquare,HomeSquare, SafetyZoneSquare, SimpleSquare και η SlideSquare.Η SlideSquare χωρίζεται σε υποκλάσεις,η StartSlideSquare, η InternalSlideSquare και η EndSlideSquare.Τέλος υπάρχουν και οι κλάσεις Player,Pawn,Deck,Dashboard.

Card Interface and Other Classes for Cards

Αρχικά φτιάχνοντας τη διεπαφή Card μας δίνεται η δυνατότητα να προσπελάσουμε τα δεδομένα χωρίς να πρέπει να ορίσουμε αν μία κάρτα είναι απλή η ειδική.

Το interface αυτό μας παρέχει τις εξής μεθόδους:

```
public Card(boolean isPlayed, String description, String image,int
value);//Constructor
```

```
sets if it's played,the description,image path and value of vard
```

```
1.private void setPlayed(boolean played);Transformer (Mutative)
```

```
sets the cards if are played.
```

```
2.public boolean getPlayed();Accessor (Selector)
```

```
returns the value if is played.
```

```
3.private void setDescription(String description);Transformer (Mutative)
```

sets the value.

4.**public** String getDescription(); Accessor (Selector)

returns the description.

5.**public** String getImage();Accessor (Selector)

returns image path.

6.**public int** getValue();Accessor (Selector)

returns value of card.

Στη συνέχεια έχουμε την NumberCard και την SorryCard που υλοποιούν την Card.

Class NumberCard

Εδώ θα αναφέρουμε τα attributes και τις υπόλοιπες μεθόδους που έχει η

κλάση αυτή (εκτός από αυτές που υλοποιεί μέσω της διεπαφής Card).

Τα attributes:

1) **private int** number; //The number of the card

2)**private boolean** isPlayed; //check if the card has played

Οι υπόλοιπες μέθοδοι :

public NumberCard(**boolean** isPlayed, String description, String image, **int** value);//Constructor

inherits super from Card

1.**public int** getNumber(); Accessor (Selector)

return the number of card

2.**public int** getisPlayed(); Accessor (Selector)

returns if it's played

Class SorryCard

Οι υπόλοιπες μέθοδοι :

public SorryCard(String description,**int** value); //Constructor

inherits super from Card

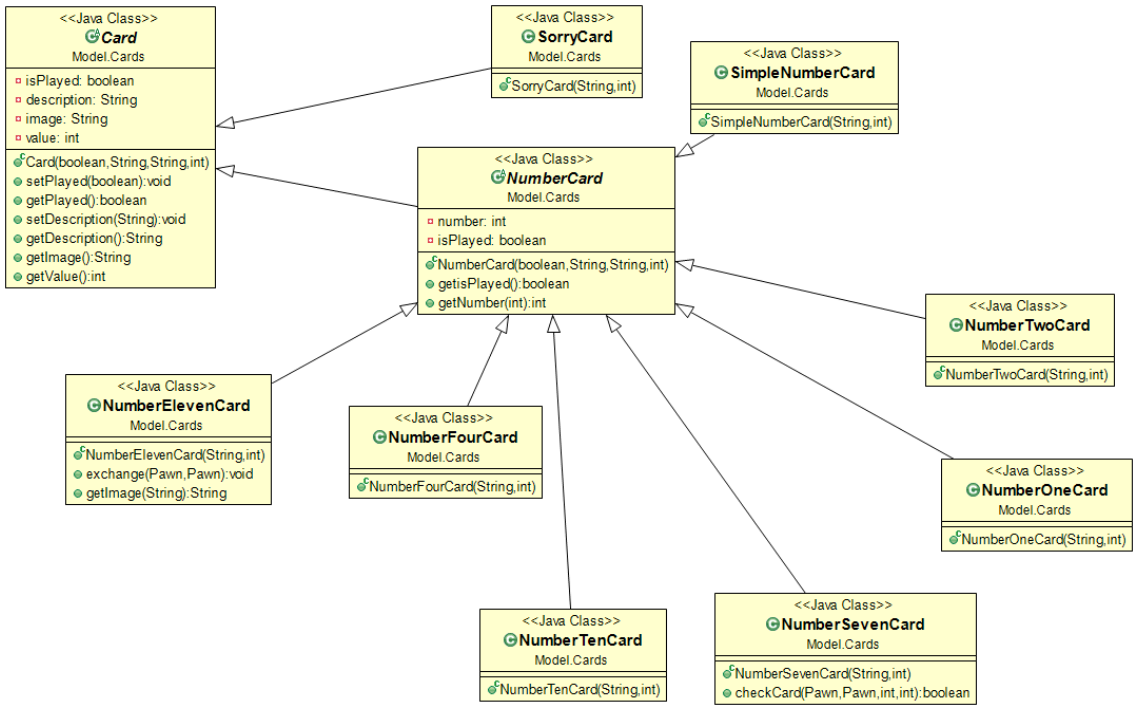
Classes SimpleNumberCard,NumberOneCard,NumberTwoCard,NumberFourCard,NumberSevenCard,NumberTenCard,NumberElevenCard

Αυτές οι κλάσεις κάνουν extend την NumberCard και μέσω της εντολής super αποκτούν πρόσβαση στη κλάση NumberCard .

Και η NumberElevenCard έχει επίσης: `public void exchange(Pawn pawn,Pawn pawn1);`

Και η NumberSevenCard έχει επίσης: `public boolean checkCard(Pawn pawn,Pawn pawn1,int x,int y)`

Τέλος, εδώ θα δείξουμε μια αναπαράσταση των κλάσεων που έχουν σχέση με τις κάρτες μέσω UML.



Class Dashboard

Αυτή η κλαση αρχικοποίει ένα ταμπλό με 74 θέσεις.

Τα attributes:

```
Square[][] dashboard;
```

Οι υπόλοιπες μέθοδοι:

```
1.public void initializeHomeSquares();Transformer (Mutative)
```

initialises home squares

2.`public void` initializeStartSquares();Transformer (Mutative)

intialises start squares

3.`public void` initializeSafetyZoneSquares();Transformer (Mutative)

initialises safety zone squares

4.`public void` initializeSimpleSquares();Transformer (Mutative)

intialises simple squares

5.`public void` initializeSlideSquares();Transformer (Mutative)

initialises slide squares

6.`public` Square getPosition(`int` i);Accessor (Selector)

returns posiiton of dashboard

Class Deck

Αυτή η κλάση αρχικοποιεί δυο παίκτες τον κόκκινο και τον κίτρινο.

Τα attributes:

```
1. private Player redPlayer, yellowPlayer; //yellow and red player
2. private int red1position=3; //3 is the number of start position
3. private int red2position=3; //3 is the number of start position
4. private int yellow1position=33; //33 is the number of start position
5. private int yellow2position=33; //33 is the number of start position
6. private static ArrayList<Card>cards=new ArrayList<Card>(); //list of cards
7. private static ArrayList<Card> cardsremoved = new ArrayList<Card>();//list of
   cards we remove
8. private boolean nomore = false; // variable for no more cards
9. Card selectedCard;//selected card
10.private Dashboard dash;//dasboard
```

Οι υπόλοιπες μέθοδοι :

1)`public` Dashboard getDash();Accessor (Selector)

returns the board

2)`public` Player getredPlayer();Accessor (Selector)

returns the red player

3)`public` Player getyellowPlayer();Accessor (Selector)

returns the yellow player

4)`public` Card keepCard();Accessor (Selector)

returns the card we removed

5)`public` Card removeCard(`int` index);Accessor (Selector)

return removed card

6)`public void` initializeCards();Observer

initializes the cards

7)`public` Card setSelectedCard(Card selected);Accessor (Selector)

returns the card we selected

8)`public void` initializeDashboard();Observer

intializes board

9)`public void` drawCard(Player p);Observer

draws a card

10)`public boolean` checkNoMore();Observer

checks if there are more cards

11)`public void` suffleCards(); Observer

suffles cards

12)`public` ArrayList<Card> getCards();Accessor (Selector)

gets list of cards

13)`public boolean` checkFold();Observer

checks if a player pressed fold

14)`public int` moveYellowPawn2(`int` x);Observer

moves yellow pawn 2

15)`public int` moveYellowPawn1(`int` x);Observer

moves yellow pawn 1

16)`public int moveRedPawn2(int x);`Observer

moves red pawn 2

17)`public int moveRedPawn1(int x);`Observer

Class Pawn

Αυτή η κλάση αρχικοποιεί τις συντεταγμένες του πιονιου και το χρώμα του.

Τα attributes:

```
1.private String color; //Color of pawn
2.private Square square; //square where pawn is placed
3.private boolean on,end; //if pawn is on r on the end
4.private int x, y, position;//x and y and position of pawn
```

Οι υπόλοιπες μέθοδοι :

1.`private void setX();`Transformer (Mutative)

sets x

2.`public String getX();`Accessor (Selector)

gets x

3.`private void setsquare();`Transformer (Mutative)

sets square

4.`public Square  getPositionOfArray();`Accessor (Selector)

gets position at dashboard

5.`public String getColor();`Accessor (Selector)

gets the color of pawn

7.`private void setY();`Transformer (Mutative)

sets y

8.`public String getY();`Accessor (Selector)

gets y

Class Player

Αυτή η συνάρτηση ελέγχει αν το χρώμα του πιονιού είναι κόκκινο ή κίτρινο.

Τα attributes:

1)String color,name;//the color and the name of the player

2)Pawn pawn1,pawn2;//the pawns of the player

3)boolean played;//if he had played

4)private int position;//position of pawn

5)private Deck d;//deck

Οι υπόλοιπες μέθοδοι :

1)public void movePosition(int x) Observer

adds to the position the value of card

2)public String getPosition();Accessor (Selector)

gets the position

3)public String getcolor();Transformer (Mutative)

returns color

4)public String getName();Accessor (Selector)

gets name

5)public boolean getPlayed();Accessor (Selector)

gets if has played

6)public Pawn getPawn1();Accessor (Selector)

gets pawn1

7)public ArrayList<Card> getCards();Accessor (Selector)

gets card list

6)public Pawn getPawn2();Accessor (Selector)

gets pawn2

8)public void toggleTurn();Observer

toggles turn

9)public boolean canPlay();Accessor (Selector)

checks if it's the player's turn

Class Square

Η Square μοντελοποιεί το κάθε τετράγωνο του παιχνιδιού

Τα attributes:

```
private int x,y;//coordinates
private String color;//color of square
private Pawn pawn1,pawn2;//pawns
```

Οι υπόλοιπες μέθοδοι :

1)`private void setX(int x);Transformer (Mutative)`

sets x coordinator

2)`public int getX();Accessor (Selector)`

gets x coordinator

3)`private void setY(int y);Transformer (Mutative)`

sets y coordinator

4)`public int getY();Accessor (Selector)`

gets y

5)`private void setColor(String color);Transformer (Mutative)`

sets color

6)`public String getColor();Accessor (Selector)`

gets color

7)`private void setEmpty(boolean empty);Transformer (Mutative)`

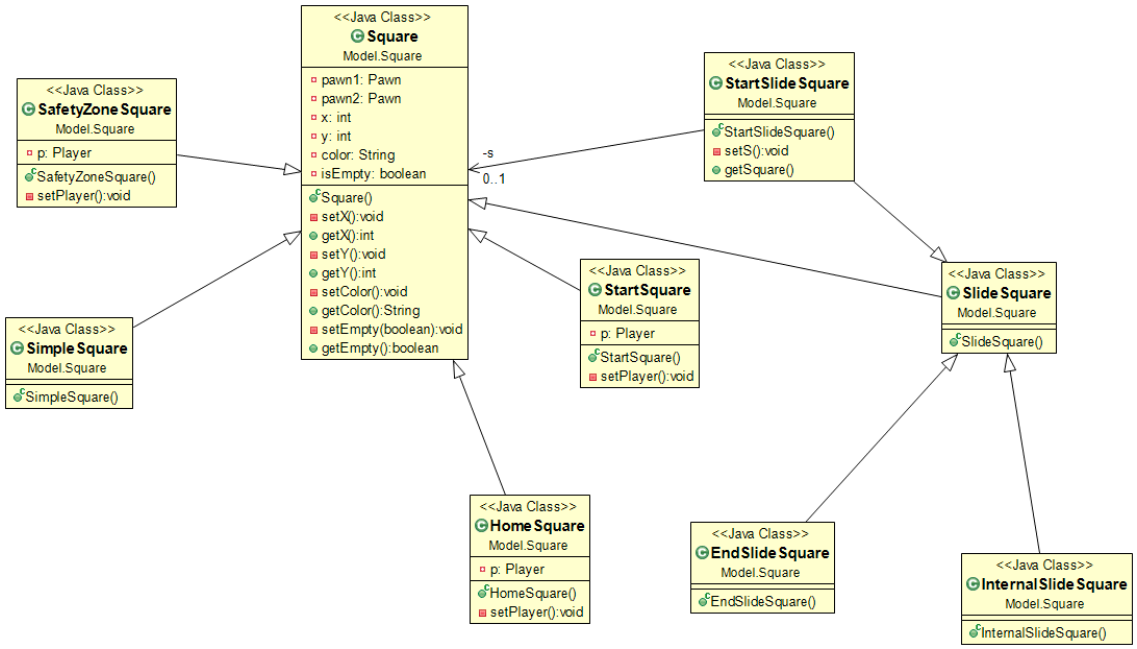
sets empty

8)`public boolean getEmpty();Accessor (Selector)`

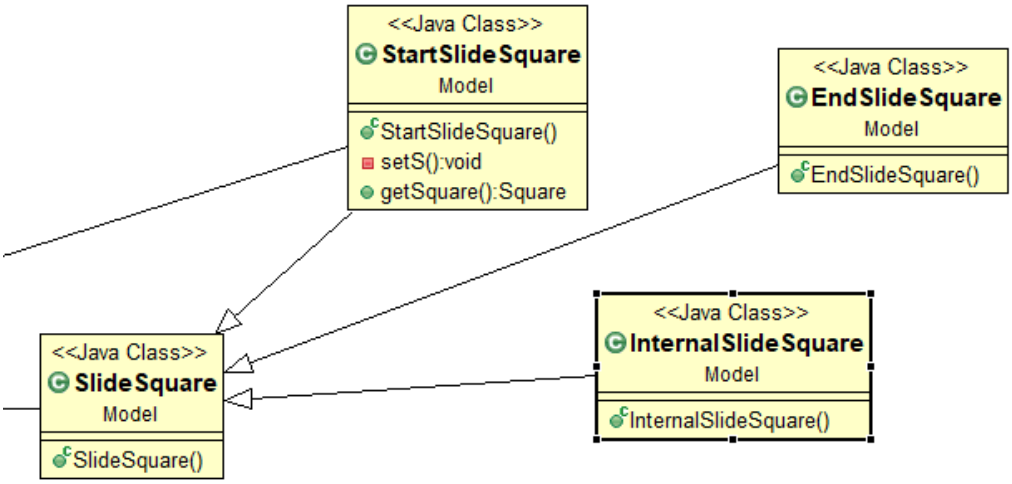
gets empty

Classes StartSquare,HomeSquare, SafetyZoneSquare, SimpleSquare,SlideSquare.

Αυτές οι κλάσεις κάνουν extend την Square και μέσω της εντολής super αποκτούν πρόσβαση στη κλάση Square και όλες υλοποιούν τις setPlayer,getPlayer.



Η SlideSquare αναπαριστά ένα τετράγωνο μίας τσουλήθρας και μπορεί να χωριστεί σε υποκλάσεις,όπως η StartSlideSquare, η InternalSlideSquare και η EndSlideSquare.Αυτές υλοποιούν κάθε τετράγωνο και με την super υλοποιούν και τις μεθόδους της SlideSquare



- ## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Class Controller

Αυτή η κλάση είναι ουσιαστικά το μυαλό του παιχνιδιού. Είναι υπεύθυνη για τη δημιουργία ενός νέου παιχνιδιού, μιας νέας παρτίδας ,τη δημιουργία στιγμιοτύπων παικτών, ομάδων σειράς και γύρων και φυσικά τη σύνδεση μεταξύ των γραφικών και του Model.Αυτό που κάνει η κλάση αυτή είναι να παίρνει τις επιλογές του χρήστη μέσω των γραφικών και να πραγματοποιεί οποιαδήποτε ενέργεια χρειάζεται έτσι ώστε το παιχνίδι να παίζεται σωστά. Φυσικά είναι υπεύθυνη αυτή η κλάση για να υπολογίζει το σκορ και να ενημερώνει πότε τελειώνει το παιχνίδι.

Τα attributes της κλάσης:

```
1)private View view;//access to view

2)private Deck deck;//access to deck

4)int red1counter = 0;//counter if red1 has started

5)int red2counter = 0;//counter if red2 has started

6)int yellow1counter = 0;//counter if yellow1 has started

7)int yellow2counter = 0;//counter if yellow2 has started

8)String red = "red";//setting red as red

9)String yellow = "yellow";//setting yellow as yellow

10)Card c;//access to a card
```

Οι υπόλοιπες μέθοδοι :

```
1)public void initialize();Observer

intializes deck

2)public boolean movePlayer(Player player, Card c);Accessor(Selector)

returns if the player has moved position

4)public Player playersTurn();Accessor(Selector)

returns which player's turn is

5)public Deck getDeck();Accessor(Selector)
```

gets deck

```
6)public void setListeners();//Observer
```

sets listeners

Οι παρακάτω κλάσεις βρίσκονται μέσα στον Controller:

```
private class CardListener implements MouseListener
```

Αυτή η κλάση υλοποιεί το πάτημα της κάρτας ωστε να την αλλάξουμε και να σούμε ποια κάρτα διαλέξαμε.

```
class PawnMovementListener implements ActionListener
```

Αυτή η κλάση υλοποιεί οτιδήποτε κανει το πιονι όταν το πατάμε ή όταν πατάμε το κουμπί fold.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Στην γραφική απεικόνιση θα έχουμε μεθόδους που υλοποιούν τα γραφικά μέσα στο παιχνίδι.Θα έχουμε attributes για το μενου,το τετράγωνο,τετράγωνο με φωτογραφία,τετράγωνο με κείμενο για το τετράγωνο Home,κουμπιά για τα πιόνια και για τις κάρτες,κείμενο που θα περιγράφει κάρτες ή εντολές,ένα κουμπί για να πατάει ο παίκτης fold και ένα πλαίσιο με πληροφορίες.Θα υπάρχουν μέθοδοι που θα μπορώ να πατάω το πιόνι που θα διαλέγω και το τετράγωνο που θέλω να πάει,που θα τραβάω την κάρτα και θα ξέρει ο Controller ποιά κάρτα πατήθηκε για να εκτελέσει τις εντολές του και το μενου που ο παίκτης θα επιλέγει αν θέλει να ξεκινήσει νέο παιχνίδι,να αποθηκεύσει το παιχνίδι,να συνεχίσει αποθηκευμένο παιχνίδι ή να φύγει απο το παιχνίδι.

Τα attributes της κλάσης:

```
JButton fold;//fold button
private JTextArea area;//text area
public JButton cardsb = new JButton();//button with cards we havent drew
private JButton[] redpawns = new JButton[2];//red pawns
private JButton[] yellowpawns = new JButton[2];//yellow pawns
private JButton[] pawns = new JButton[5];//all the pawns
private JLabel startred;//start area for red pawn
JLabel startyellow;//start area for yellow
private JLabel[] array = new JLabel[80];//squares of board
static JFrame frame;//whole frame
JPanel panel = new JPanel();//panel
JLayeredPane layeredPane = new JLayeredPane();//layeredpane
JLayeredPane layeredPane2 = new JLayeredPane();second layeredpane
JMenuBar menuBar = new JMenuBar();//menu
private AbstractButton cards1;//cards we can see
```

Οι υπόλοιπες μέθοδοι :

1)`public void updateCard(Card c);` Observer
updates card everytime we draw one

2)`public JLabel createButton(int x, int y);` Accessor
creates squares on board

3)`private void createColumn1();` Observer
creates first column with simple squares

4)`private void createRow1();` Observer
creates first row of simple squares

5)`private void createColumn2();` Observer
creates second column of simple squares

6)`private void createRow2();` Observer
creates second row of simple squares

7)`private void createYellowButtons();` Observer
creates safety zone for yellow pawn

8)`private JLabel createYellowButton(int y);` Accessor
creates yellow square

9)`private void createRedButtons();` Observer
creates safety zone for red pawn

10)`private JLabel createRedButton(int y);` Accessor
creates a red square

11)`public JLabel[] getButtons();` Accessor
returns array of squares

12)`public JButton getCardsb();` Accessor
returns button of the cards we havent chosen yet

13)`public void updateRedPawn(int positionPawn, int i);` Observer
puts red pawns to the right square on the board

14)`public void updateYellowPawn(int positionPawn, int i);` Observer
puts yellow pawns to the right square on the board

15)`public JButton[] getPawns();` Accessor
returns pawns

16)**public** JButton[] getRedPawns(); Accessor
returns red pawns

17)**public** JButton[] getYellowPawns(); Accessor
returns yellow pawns

18)**public** JTextArea getText() Accessor
returns text area

19)**public void** addPawnMovementListener(ActionListener listenForPawnButton)Observer
add listeners to pawns and fold button

Αλλαγές απο την Α στην Β φάση

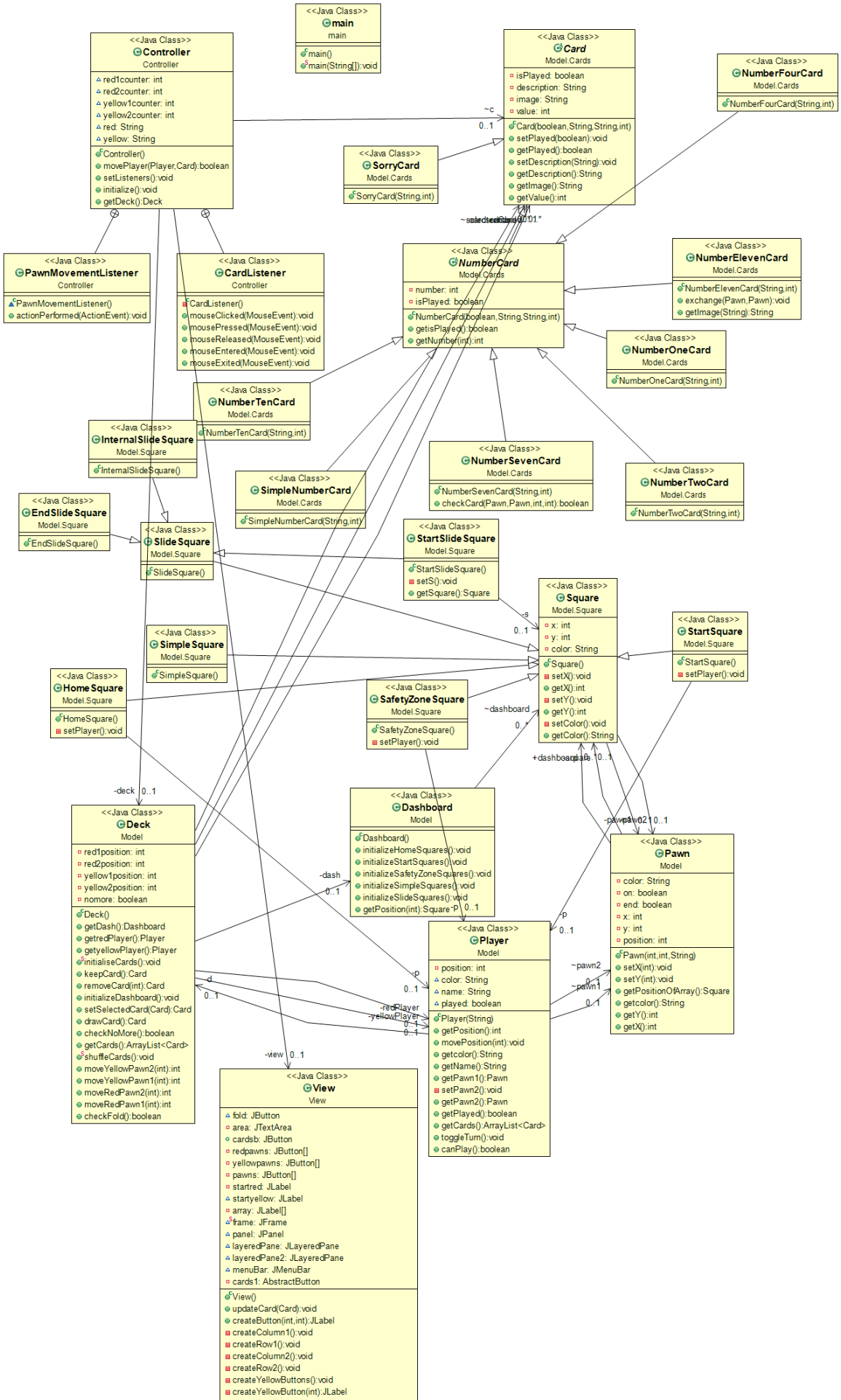
Αφαιρέθηκαν αρκετές μέθοδοι και κλάσεις απο το Model που δεν χρειαζόντουσαν και προστέθηκαν στο View αρκετά πράγματα.

JUNITS

Υπάρχουν JUnits τεστ για της μεθόδους που κουνανε τα πιονια και επιστρεφουν την θεση που πρεπει να παει το καθε πιονι μέσα στο ταμπλό

Πράγματα που δεν υλοποιήθηκαν

Αρχικά δεν υλοποιήθηκαν όλες οι κάρτες,μονο η 4, οι απλές και ξεκινάνε τα πίονια μόνο με 1 ή 2 κάρτα.Επίσης δεν έχει υλοποιηθεί το μενού μόνο τα γραφικά του.Ακόμη, το πιόνι δεν μπορεί να εκτελέσει τα χαρακτηριστικά των slides και να φάει το αντίπαλο πιόνι όταν πρέπει.Τέλος δεν έχει ολοκληρωθεί το παιχνίδι έτσι ώστε καθε πιόνι να φτάσει στην τελική θέση Home.Ωστόσο κάποιες μέθοδοι υπάρχουν μέσα στις κλάσεις και κάποιες μεταβλητές που θα μπορούσαν να υλοποιηθούν ώστε να ολοκληρωθεί το παιχνίδι.


<<Java Class>>
main
main
main()
main(String[]):void
<<Java Class>>
Controller
Controller
red1counter: int
red2counter: int
yellow1counter: int
yellow2counter: int
red: String
yellow: String
Controller()
movePlayer(Player,Card):boolean
setListeners():void
initialize():void
getDeck():Deck
<<Java Class>>
Card
Model.Cards
isPlayed: boolean
description: String
image: String
value: int
Card(boolean,String,String,int)
setPlayed(boolean):void
getPlayed():boolean
setDescription(String):void
getDescription():String
getImage():String
getValue():int
<<Java Class>>
NumberFourCard
Model.Cards
NumberFourCard(String,int)
<<Java Class>>
SorryCard
Model.Cards
SorryCard(String,int)
<<Java Class>>
PawnMovementListener
Controller
PawnMovementListener()
actionPerformed(ActionEvent):void
<<Java Class>>
CardListener
Controller
CardListener()
mouseClicked(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
<<Java Class>>
NumberCard
Model.Cards
number: int
isPlayed: boolean
NumberCard(boolean,String,String,int)
getisPlayed():boolean
getNumber(int):int
<<Java Class>>
NumberElevenCard
Model.Cards
NumberElevenCard(String,int)
exchange(Pawn,Pawn):void
getImage(String):String
<<Java Class>>
NumberOneCard
Model.Cards
NumberOneCard(String,int)
<<Java Class>>
NumberTenCard
Model.Cards
NumberTenCard(String,int)
<<Java Class>>
InternalSlideSquare
Model.Square
InternalSlideSquare()
<<Java Class>>
SimpleNumberCard
Model.Cards
SimpleNumberCard(String,int)
<<Java Class>>
NumberSevenCard
Model.Cards
NumberSevenCard(String,int)
checkCard(Pawn,Pawn,int,int):boolean
<<Java Class>>
NumberTwoCard
Model.Cards
NumberTwoCard(String,int)
<<Java Class>>
EndSlideSquare
Model.Square
EndSlideSquare()
<<Java Class>>
SlideSquare
Model.Square
SlideSquare()
<<Java Class>>
StartSlideSquare
Model.Square
StartSlideSquare()
setS():void
getSquare():Square
<<Java Class>>
Square
Model.Square
x: int
y: int
color: String
Square()
setX():void
getX():int
setY():void
getY():int
setColor():void
getColor():String
<<Java Class>>
StartSquare
Model.Square
StartSquare()
setPlayer():void
<<Java Class>>
Simple Square
Model.Square
SimpleSquare()
<<Java Class>>
Home Square
Model.Square
HomeSquare()
setPlayer():void
<<Java Class>>
SafetyZone Square
Model.Square
SafetyZoneSquare()
setPlayer():void
<<Java Class>>
Deck
Model
red1position: int
red2position: int
yellow1position: int
yellow2position: int
nomore: boolean
Deck()
getDash():Dashboard
getredPlayer():Player
getyellowPlayer():Player
initialiseCards():void
keepCard():Card
removeCard(int):Card
initializeDashboard():void
setSelectedCard(Card):Card
drawCard():Card
checkNoMore():boolean
getCards():ArrayList<Card>
shuffleCards():void
moveYellowPawn2(int):int
moveYellowPawn1(int):int
moveRedPawn2(int):int
moveRedPawn1(int):int
checkFold():boolean
<<Java Class>>
Dashboard
Model
Dashboard()
initializeHomeSquares():void
initializeStartSquares():void
initializeSafetyZoneSquares():void
initializeSimpleSquares():void
initializeSlideSquares():void
getPosition(int):Square
<<Java Class>>
Pawn
Model
color: String
on: boolean
end: boolean
x: int
y: int
position: int
Pawn(int,int,String)
setX(int):void
setY(int):void
getPositionOfArray():Square
getcolor():String
getY():int
getX():int
<<Java Class>>
Player
Model
position: int
color: String
name: String
played: boolean
Player(String)
getPosition():int
movePosition(int):void
getcolor():String
getName():String
getPawn1():Pawn
setPawn2():void
getPawn2():Pawn
getPlayed():boolean
getCards():ArrayList<Card>
toggleTurn():void
canPlay():boolean
<<Java Class>>
View
View
fold: JButton
area: JTextArea
cardsb: JButton
redpawns: JButton[]
yellowpawns: JButton[]
pawns: JButton[]
startred: JLabel
startyellow: JLabel
array: JLabel[]
frame: JFrame
panel: JPanel
layeredPane: JLayeredPane
layeredPane2: JLayeredPane
menuBar: JMenuBar
cards1: AbstractButton
View()
updateCard(Card):void
createButton(int,int):JLabel
createColumn1():void
createRow1():void
createColumn2():void
createRow2():void
createYellowButtons():void
createYellowButton(int):JLabel