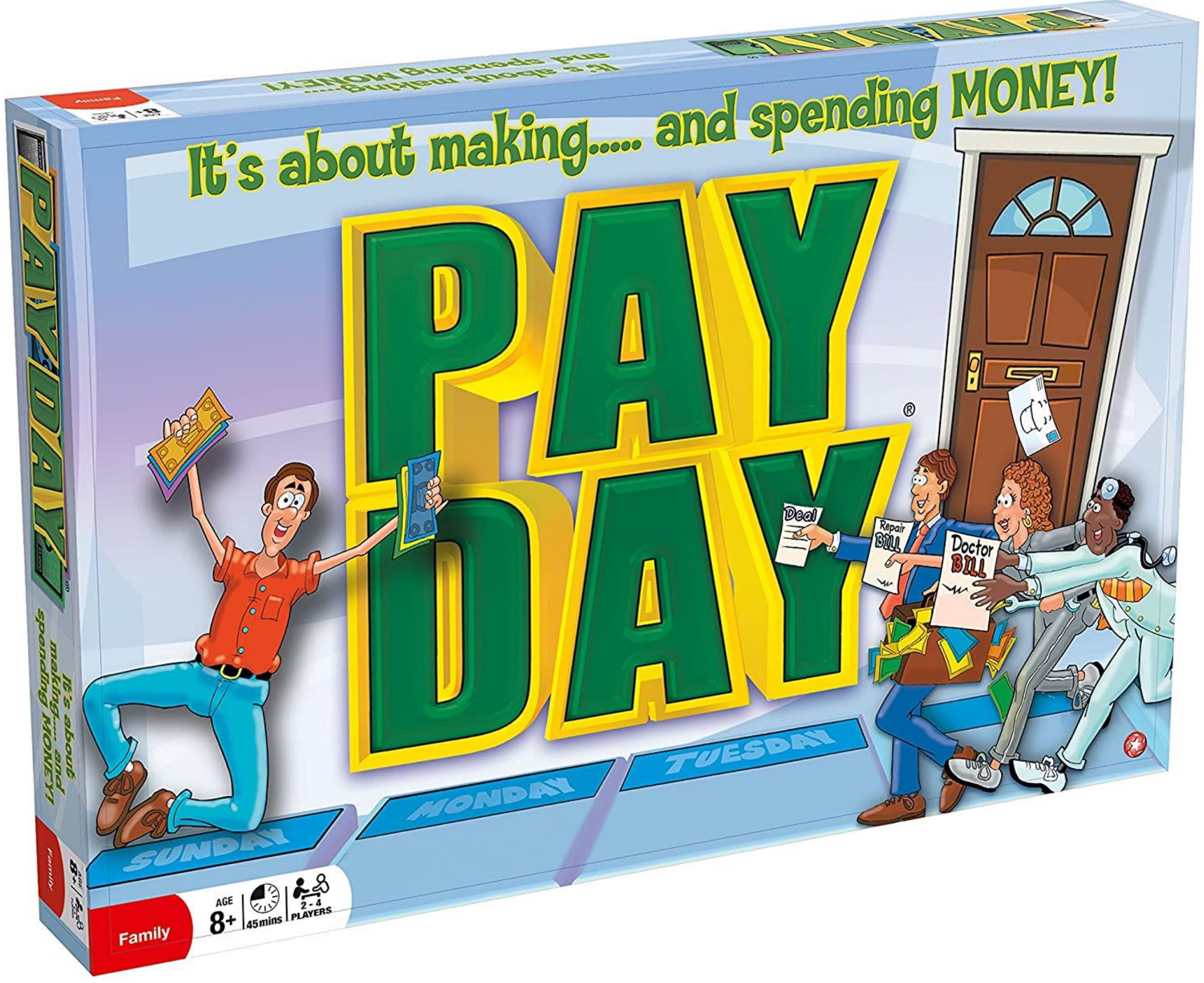


PAYDAY

Author : Trypidakis Orestis csd4235

Course: Object Oriented Programming

Proffessor: Tzitzikas Ioannis

Contents

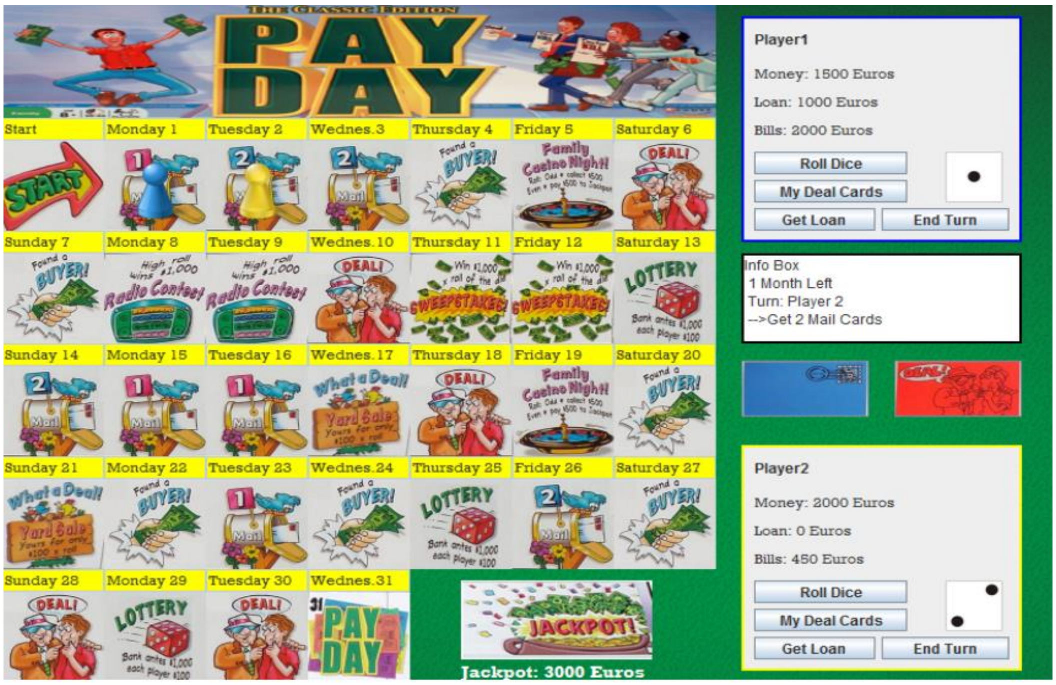
Introduction	2
Package Model • CardPackage • CardDecksPackage • DicePackage • JackpotPackage • TilesPackage • PlayerPackage	3
Package Controller Class Controller	16
Package View Class GraphicUI	18
UML Diagrams	20
Functionality	21
Conclusion	21

Introduction

In this Project, I am going to create the famous board game Payday on Java.For the project implementation I will be using the MVC(Model,View,Controller) architecture.The goal is to use Controller as the link between Model and View. On this report ,I will mention the parts of Model and Controller which are usefull for the first phase of the project,also some parts of the View class will be mentioned.Lastly,some UML photos from each package will be included in the report.

The goal and the finished programm will resemble this reference photo.

First photo is the expected.Second photo is the finished product.



Player1

Money: 1500 Euros

Loan: 1000 Euros

Bills: 2000 Euros

Info Box
1 Month Left
Turn: Player 2
--->Get 2 Mail Cards

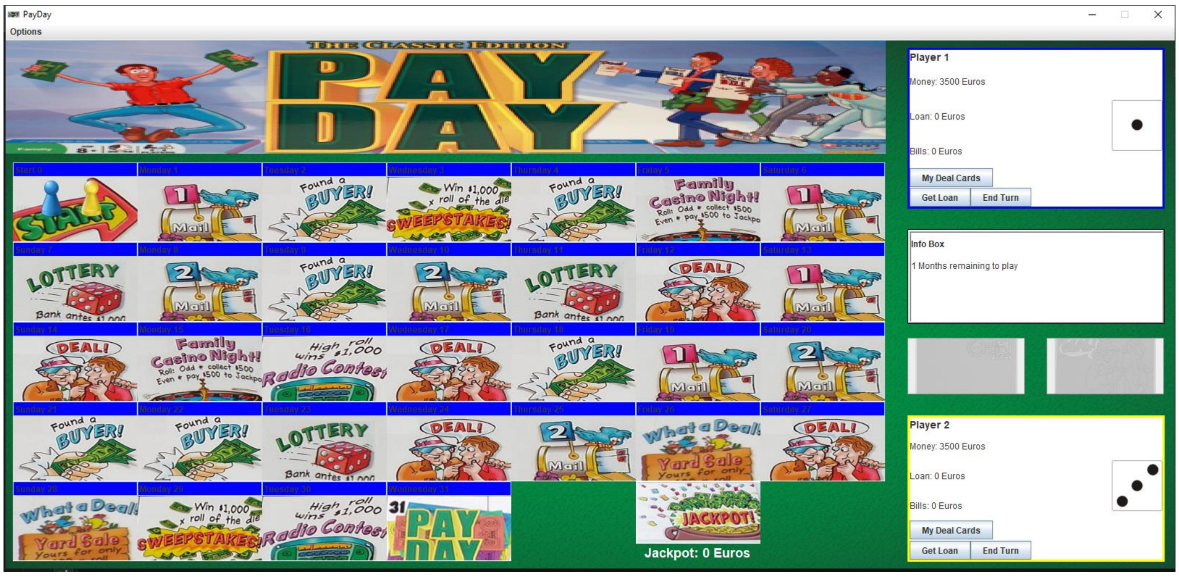
Player2

Money: 2000 Euros

Loan: 0 Euros

Bills: 450 Euros

Jackpot: 3000 Euros



Player 1

Money: 3500 Euros

Loan: 0 Euros

Bills: 0 Euros

Info Box

1 Months remaining to play

Player 2

Money: 3500 Euros

Loan: 0 Euros

Bills: 0 Euros

Jackpot: 0 Euros

Package Model

On model package there are 6 packages.

- Card
- CardDecks
- Dice
- Jackpot
- Player
- Tiles

Card Package

Card abstract class

Card class is an abstract class that contains the basic function of a simple card, also there is a message card class that exists for special cards.Card contains 8 subclases.

Included attributes in Card

```
private final String text; //The message that the card prints
private final int money; // The money that the card costs
private final String imgURL; // The pathof the card's image
```

Included methods in Card

- `public Card(int money,String text,String imgURL){}`

```
//Constructor initializes money text and imgURL
```

- `public String getText(){} //Accessor returns the card’s text`
- `public String getMoney(){} //Accessor returns the card’s money`
- `public String getimgURL(){} //Accessor returns the card’s path`
- `public abstract void CardAction(Player p); // abstract class that all subclases will inherit`

MessageCard class

Class that creates messagecard instances.Contains a constructor with super(money,text,imgURL) because, it inherits Card class,also contains the CardAction function.

Advertisment class

Initializes an advertisment card

This card class contains an add sell it for X euros.

Bill class

Initializes an bill card

This card class contains a bill,forcing you to pay the bank X euros at the end of the month.

Charity class

Initializes an charity card

This card class contains a charity funding,you to pay the charity X ,if the player doesnt have the required money,he is forced to take a loan,

Charity money is added to the jackpot

DealCard class

Initializes an Deal card

This card class contains a deal card,you have the choice to buy the card ,or decline the deal.

Included attributes in Card

```
private int sellPrice; // Cards sell price
```

Included methods in Card

```
public int getSellValue(){}
```

```
Accessor //returns card's sell price
```

Moveto class

Initializes an Moveto card

This card class contains a Moveto card,that moves you to the closest deal/buyer tile

Included methods in Card

```
public int CardAction(Player p,Tiles table[]){}
```

Accessor, Returns a number which is how many steps does to player have to make to go to a deal/buyer tile

PayTheNeighbor class

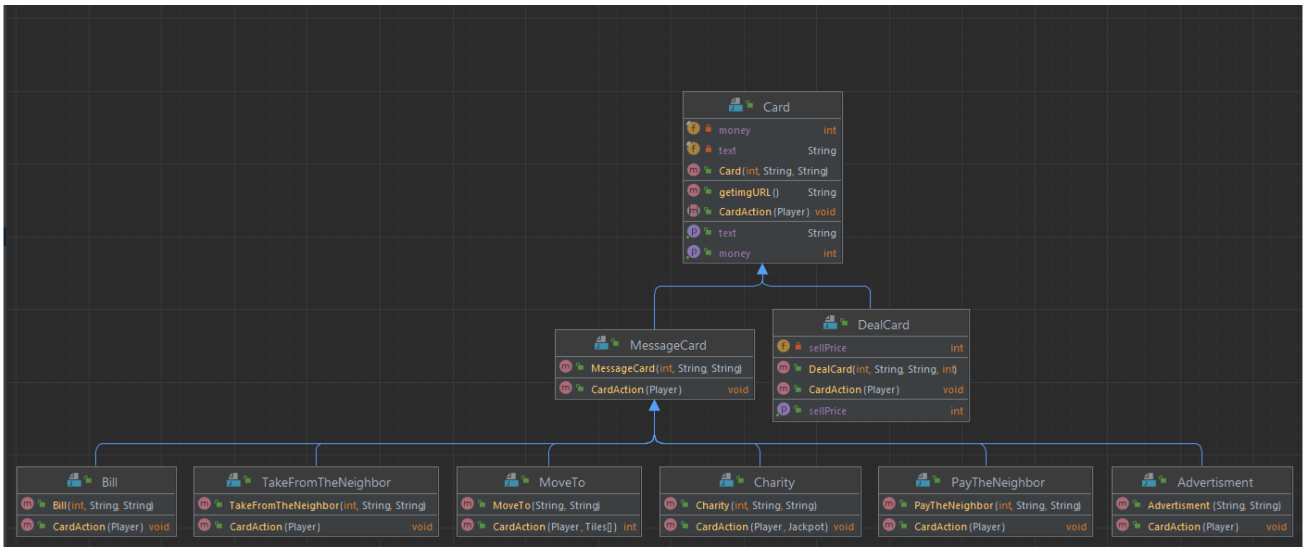
Initializes an PayTheNeighbor card

This card class contains a PayTheNeighbor card,you have to pay the opossing player X euros

TakeFromTheNeighbor class

Initializes an TakeFromTheNeighbor card

This card class contains a TakeFromTheNeighbor card,the opponent has to pay you X euros



CardDecks package

Card decks package contains 3 classes:

- DealCardDeck
- MessageCardDeck
- RejectedCards

DealCardDeck class

Class that contains the deal cards deck,is a class that takes all available dealcards and put them in a Stack.

Included attributes in DealCardDeck

```
private Stack<DealCard> Deck;

//A Stack to collect all DealCards
```

Included methods in DealCardDeck

```
public DealCardDeck(){}

//Constructor: Constructs the  Stack containing Dealcards

public void DeckShuffle(){}
```

```
//Transformer : Shuffles dealcard deck

public boolean isEmpty(){}

//Observer: Checks wheter Deck(Dealcard stack) is empty.

public void AddCard_to_Deck(DealCard dCard){}

//Transformer: Adds a new DealCard to Stack

public void getCard(){}

//Transformer: Returns a card from the stack and then removes it
(stack.pop())
```

MessageCardDeck class

Class that contains the message cards deck,is a class that takes all available messagecards and put them in a Stack.

Included attributes in MessageCardDeck

```
private Stack<DealCard> Deck;

//A Stack to collect all messagecards
```

Included methods in MessageCardDeck

```
public DealCardDeck(){}

//Constructor: Constructs the  Stack containing Dealcards

public void DeckShuffle(){}

//Transformer : Shuffles messagecards deck

public boolean isEmpty(){}

//Observer: Checks wheter Deck(messagecards stack) is empty.

public void AddCard_to_Deck(MessageCard mCard){}

//Transformer: Adds a new messagecards to Stack

public void getCard(){}

//Transformer: Returns a card from the stack and then removes it
(stack.pop())
```

RejectedCards class

Class that contains the message cards deck,is a class that takes all available messagecards and put them in a Stack.

Included attributes in RejectedCards

```
private Stack<Card> RejectedCard;

//A Stack to collect all rejected cards
```

Included methods in RejectedCards

```
public RejectedCards (){}

//Constructor: Constructs the  Stack containing RejectedCards

public void rejectedCard_return(MessageCardDeck mCard, DealCardDeck dCard)
{} //Transformer: Function that returns deal and message cards back to
their decks

public void RejectCard(Card rejCard)

//Transformer: Add a new card to rejected card stack

public boolean isEmpty(){}

//Observer: Checks wheter Deck(RejectedCards stack) is empty.
```

Dice package

Dice class

Class that contains the dice,the number it rolls,and whether the player has rolled

Included attributes in Dice

```
public int roll; //Integer: dice roll roll={1,2,3,4,5,6}
private boolean hasrolled; //Boolean :Whether the player has rolled
```

Included methods in Dice

```
public Dice(){}

//Constructor: Constructs dice initialzing roll and hasrolled

public int getRoll(){}

//Accessor: Returns the dice’s roll

public void setRoll(int roll){}
```

```
//Transformer: sets the dices roll

public void setRolled(boolean rolled){}

//Transformer: Sets whether the player has rolled

public boolean getRolled(){}

// Accessor: Returns whether the player has rolled
```

PHASE B changes:

```
public void setRoll(){}

//Transformer: sets the dices roll

public int getRollNumber(){}

//Returns players roll number
```

Jackpot package

Jackpot class

Class that creates and contains the jackpot of the game

Included attributes in Jackpot

```
private int jackpotSize; // Jackpots pool size
```

Included methods in Jackpot

```
public Jackpot(){}

//Constructor: Initializes jackpotSize to 0

public void AddJackpot(int cash){}

//Transformer: Adds cash to jackpotSize ,also checks that cash >0

public int WinBalance() {}

//Transformer: if a player wins the jackpot takes the balance that
 jackpotSize contains and the new jackpotSize equals 0

public int getjackpotSize(){}

//Accessor: Returns jackpotSize
```

```
public String toString(){}

//To string function to print jackpot size on board
```

Pawn package

Pawn class has been removed since it wasnt used and I found an easier way to use pawns.Methods were added to Player class.

Tiles package

Tiles abstract class

```
Tiles abstract class is the parent class of 10 subclases.Tiles creates all
board tiles,creates their position,checks whether its Sunday or Thursday
and sets classes names.
```

Included attributes in Tiles

```
private int Tileposition; //Tile’s position
private String name; //Tile’s name
private  final String imgURL; //Tile’s image path
```

Included method in Tiles

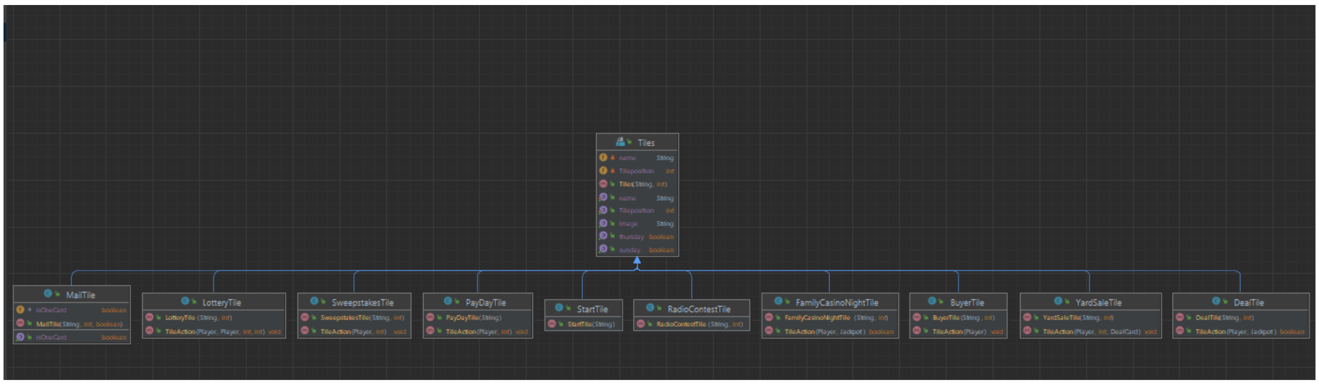
```
public Tiles(String imgURL, int Tileposition){}

//Constructor: Constructs a tile instance

public String getImage(){}

//Accessor: Returns a tiles image

public String getName(){}

//Accessor: Returns a tiles name

public String getTilePosition(){}

//Accessor: Returns a tiles position

public void setName(String name){}

//Transformer: sets tiles name

public void setTileposition(int pos){}
```

```
//Transformer: sets tiles position

public boolean isSunday(){}

//Observer checks whether its Sunday

public boolean isThursday(){}

//Observer :checks whether its Thursday
```

Classes that extend Tiles

- BuyerTile
- DealTile
- FamilyCasinoNightTile
- LotteryTile
- MailTile
- PayDayTile
- RadioContestTile
- StartTile
- SweepstakesTile
- YardSaleTile

Each class contains its individual constructor that takes imgURL and Tilepostion from Tile,and each contain a TileAction function that,work according to the type of Tile.


Tiles
name String
Tileposition int
Tiles(String, int)
name String
Tileposition int
image String
thursday boolean
sunday boolean
MailTile
isOneCard boolean
MailTile(String, int, boolean)
isOneCard boolean
LotteryTile
LotteryTile (String, int)
TileAction(Player, Player, int, int) void
SweepstakesTile
SweepstakesTile(String, int)
TileAction(Player, int) void
PayDayTile
PayDayTile(String)
TileAction(Player, int) void
StartTile
StartTile(String)
RadioContestTile
RadioContestTile (String, int)
FamilyCasinoNightTile
FamilyCasinoNightTile (String, int)
TileAction(Player, Jackpot) boolean
BuyerTile
BuyerTile (String, int)
TileAction(Player) void
YardSaleTile
YardSaleTile(String, int)
TileAction(Player, int, DealCard) void
DealTile
DealTile(String, int)
TileAction(Player, Jackpot) boolean

Player package

Player class

This class represents the Player s of the game,initializes each player and gives acces to information about the player

Included attributes in Player

```
private String name; //Player’s name
private boolean hasPlayed,hasFinished; //If player hasPlayed-hasFinished
private int money,loan,bills; //Player’s money – loan - bills
private int position; // Player’s position
private int AvailableMonths; //Player’s available months
private Stack<DealCard> pDealCards; //Player’s deal card possesion
```

Included methods in Player

```
public Player(String name){}

//Constructor: Constructs a player instance by name setting all the
requirments

public int getMoney(){}

//Accessor: Returns players money

public void setMoney(int money){}

//Transformer: sets players money

public int getLoan(){}

//Accessor: Returns players Loan

public void setLoan (int Loan){}

//Transformer: sets players Loan

public int get Bills (){}

//Accessor: Returns players Bills

public void set Bills (int Bills){}

//Transformer: sets players Bills

public String getName(){}

//Accessor: Returns player name

public void setName(String newName){}

//Transformer: Sets players name

public int getPosition(){}

// Accessor: Returns players postion

public void setPosition(int position)

//Transformer: Sets players position
```

```
public int getAvailableMonths(){}

//Accessor: Returns players available months

public void setAvailableMonths(int availableMonths){}

//Transformer:Sets players available months

public void set_HasPlayed(){}

//Transformer:Sets if player has played

public boolean get_HasPlayed(){}

//Accessor:Returns if player has played

public void set_HasFinished(){}

//Transformer:Sets if player has finished

public boolean get_has_finished(){}

//Accessor:Returns if player has finished

public void HoldDealCard(DealCard dCard){}

//Transformer:Adds a new deal card to players stack

public DealCard SellDealCard(){}

//Transformer:Removes card from players stack

public boolean isEmptyCards(){}

//Observer: Checks whether the player has cards

Phase B changes:


public Dice getDice() {}

//return the dice of the player

public void setName(String newName){}

//setName removed

public Player getOpponent() {   return this.Opponent;}

//returns players opponent

public void setTurn(boolean turn) {}

//Sets players turn
```

```
public boolean getTurn() {return turn;}

//Gets players turn
```

Turn class

Turn class has been removed since it wasnt used and I found an easier way to use turns.Methods were added to Player class.

Package Controller

This class is the practically the brain of the game. The Controller is responsible for controlling the application logic and acts as the coordinator between the View and the Model. The Controller receives an input from the users via the View, then processes the user's data with the help of Model and passes the results back to the View.This class also calculates the score,and declares when the game finishes.

Included attributes in Controller

```
public Player p1;
public Player p2;
```

Included methods in Controller

```
public Controller(){}

//Constructor: Constructs a new controller and sets the game as eligible
to start

public void initialize_game(){}

// Function that initializes the game bu calling all its functions

public void initialize_board(){}

// Function that initializes the board bu setting all tiles

public void randomize_board()

// Function that initializes the board bu setting all tiles

public void initialize_CardDecks(){}

// Function that randomises the cards bu putting them at their decks

public void initialize_PlayMonths(){}

// Function that set how many months the players want to play ,according
to their votes

public void initialize_GoesFirst(){}

// Function that chooses which player goes first according to their rolls

public void GameOver(){}

// Function that calculates the winner according to their money ,loans and
bills

public Player GameWinner(){}
```

```
// Function that calculates the winner according to their money ,loans and
bills

public void CreateSundayMatch(){}

//Makes player choose if he wants to bet on a football match\

public void CreateThursdayCrypto(){}

// Makes player choose if he wants to bet on crypto

public void WinJackpot(Player p) {}

//Function that Adds jackpotmoney to player
```

Package View

GraphicUI class

This package will contain the main class GraphicUI which will create a frame and inside it a JLayredPanel,inside this panel,there will be 10 buttons and some actionlisteners (For players turn,dices,cards).Some texts field for information.Same labels that will contain players names and jackpot.And 32 panels that will contain all the tiles of the board.

Included attributes in View

UPDATE FOR PHASE B

```
private JLayeredPaneExtension BasePanel;
Controller game;
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
private final int width = (int) Math.round(screenSize.getWidth()) -300;
private final int height = (int) Math.round(screenSize.getHeight())-300;
private int counter = 0;
private boolean RadioPos = false;
private boolean LotteryPos = false;
private final ClassLoader cldr;
private URL imgURL;
private Image image;
private final JButton DealCards;
private final JButton MessageCards;
private final JButton DealCardsA;
private final JButton DealCardsB;
private final JButton GetLoanP1;
private final JButton GetLoanP2;
private final JButton EndTurnP1;
private final JButton EndTurnP2;
private final JButton DiceP1;
private final JButton DiceP2;
private final JDesktopPane InfoBox;
private final JDesktopPane table;
private final JDesktopPane P1;
private final JDesktopPane P2;
private final JDesktopPane jackPotPanel;
private final JLabel PayDayImage;
private final JLabel NameP1;
private final JLabel NameP2;
private final JLabel jackPotLabel;
private final JTextField MoneyP1;
private final JTextField MoneyP2;
private final JTextField LoanP1;
private final JTextField LoanP2;
private final JTextField BillsP1;
private final JTextField BillsP2;
private final JTextField jackPotText;
private final JTextField info;
private final JTextField turn;
private final JTextField monthsleft;
private final JTextField command;
private final JDesktopPane[] Tiles;
int LotterychoiceP1, LotterychoiceP2;
JLayeredPaneExtension[] pawn_position;
private JMenu MenuFile;
ImageIcon imgICN;
```

<u>Included methods in View</u>

<u>UPDATE FOR PHASE B</u>

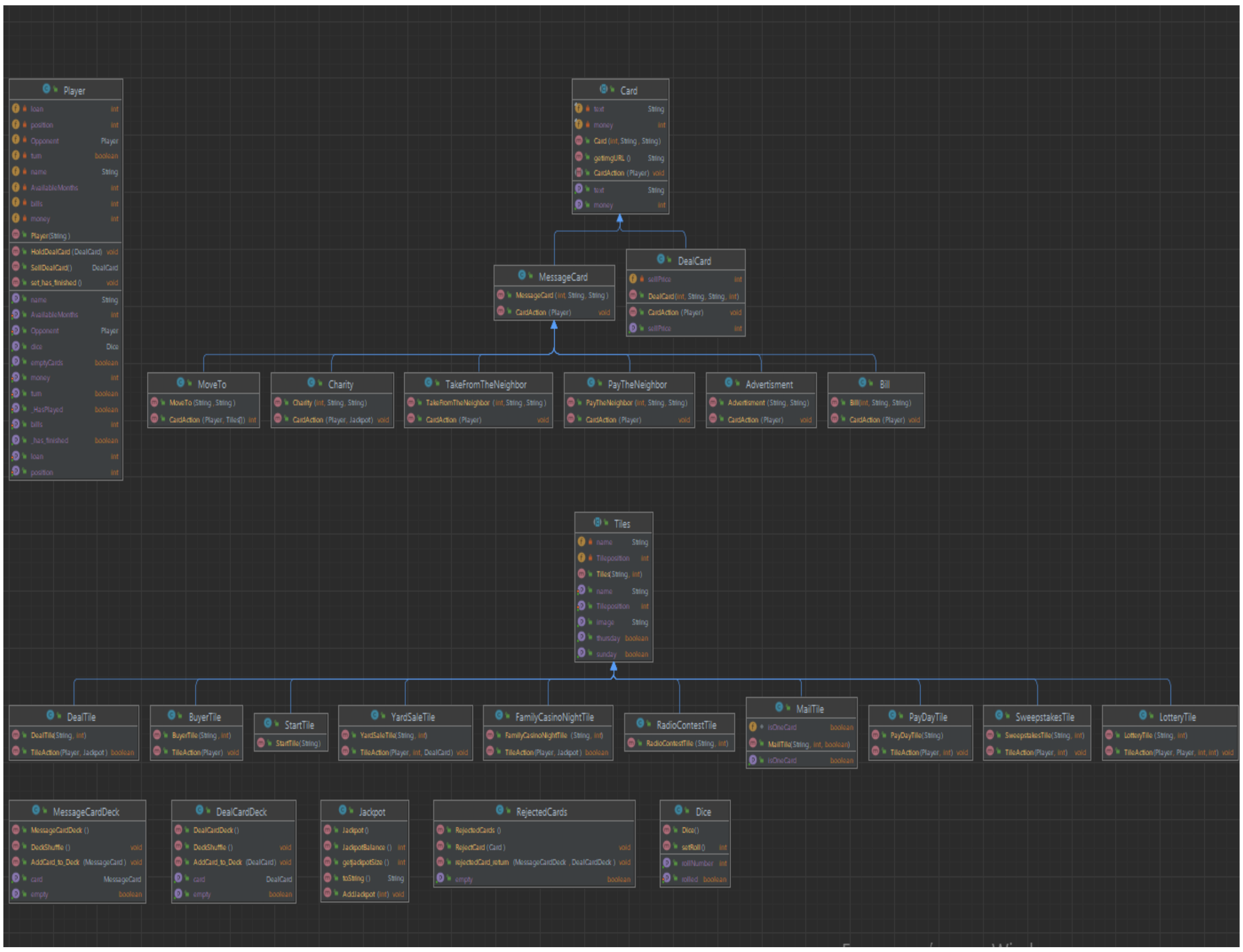
```
public GraphicUI(){}
private void init_Components() {}

private void StartAll() {}

public void NewGame() {}

public void showMailCard(Card MailCard) {}

private class CardListener implements ActionListener

public void initialize_board_graphics()

public void MessageCardAction(Player player, MessageCard C)

private void TileActions(Player p, int position)

private void Thursday(Player p)

private void Sunday(Player p)

private void JackpotInformation()
private void PawnInformation(int position, String name)
private void initialize_Infobox()

private void InfoboxInformation(String commandtxt)

private void PlayerInformation()

private void PlayerPanel()

private class DiceListener implements ActionListener
private void Move_Player(ActionEvent e)

public void Starts_First()

private void DiceInformation(int DiceRoll, String name)

private class ButtonListenerP1 implements ActionListener

public void PlayersDealCards(Player p)

public void GetLoan(Player p)

private class ButtonListenerP2 implements ActionListener

public void CreateMenu()

private class Menulistener implements ActionListener
```

Other classes

These classes are for certain events and will popup when needed

- GamePhasePopUp
- GetLoeanPopUp
- LoanDayPopUp
- LotteryPickPopUp
- MonthChoicePopUp
- SyndayMatchPopUp
- ThursadayCryptoPopUp

UML diagramms



Functionality (Phase B)

I have managed to create all required function and what was requested.

The programm is fully functional,expect an unexplainable bug the creater a null pointer exception on the dice when the dice roll button is pressed too fast.

Conclusion

In conclusion, the game was a huge project that required a lot of time,i have learned to use classes inheritance etc and a lot of graphis.The most important thing i want to note is for the code for the paydaycards that was given to us, I changed it quite a bit ,most importantly instead of taking a random card through the rand function like the code,I used the shuffle method on the card stacks,as a result the program is quite different.