University Of Crete – Computer Science Department

CS252– Object-Oriented programming - Java

Professor: I. Tzitzikas

Fall Semester 2019-2020

Georgios Gerasimos Leventopoulos

AM 4152

05/12/2019

Contents :

• Introduction

In order to make this application I used MVC Model and OOP Principles.
(MVC=Model-View-Controller) . Specifically the model is the “brain” of the game that implements all the required classes fo the game to work as expected. View is the UI (User Interface), what the user see while he is using the game . Also, the Controller is the who controls for the right communication between Model and View . In the next pages, there is an explanation for every package I used in the project: the attributes and the methods of every class and also the UML of every package.

• **Package Model**

-Abstract Class Card

When we create an abstract class Card, we can access all of our data without specifying if a card is simple or specific.

Attributes :

protected String image. // String for the image path of a card.

protected String name. // String for the name of a card.

Methods :

public Card(String name, String image) //Constructor

1) public void setName(String name) // Accessor(Selector) ,Sets the name of the card.

2)public void setImage(String image) //Accessor (Selector), Sets the image of the card.

3)public String getName() //Transformer(Mutative) , Returns the name of a card. 4)public String getImage() //Transformer(Mutative) , Sets the image of a card.

Classes that extend Card : NumberCard , SorryCard

-Abstract Class NumberCard

Attributes

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

Methods

public NumberCard(String name, int value, String image) //Constructor

1)public int getValue(); //Accessor , Returns the value of the card.

2)public void setValue(int number) //Transformer , Sets the value of the card.

-Class SorryCard

Attributes: -

Methods:

SorryCard(String name, String image) //Constructor

1)public boolean (pawn playerPawn, pawn enemyPawn) //Check if the player can switch with his pawn. 2) public String toString() //Returns the description of the card.

Classes that extend Class NumberCard :

NumberOneCard, NumberTwoCard, NumberThreeCard,

NumberFourCard, NumberFiveCard, NumberSevenCard , NumberEightCard,

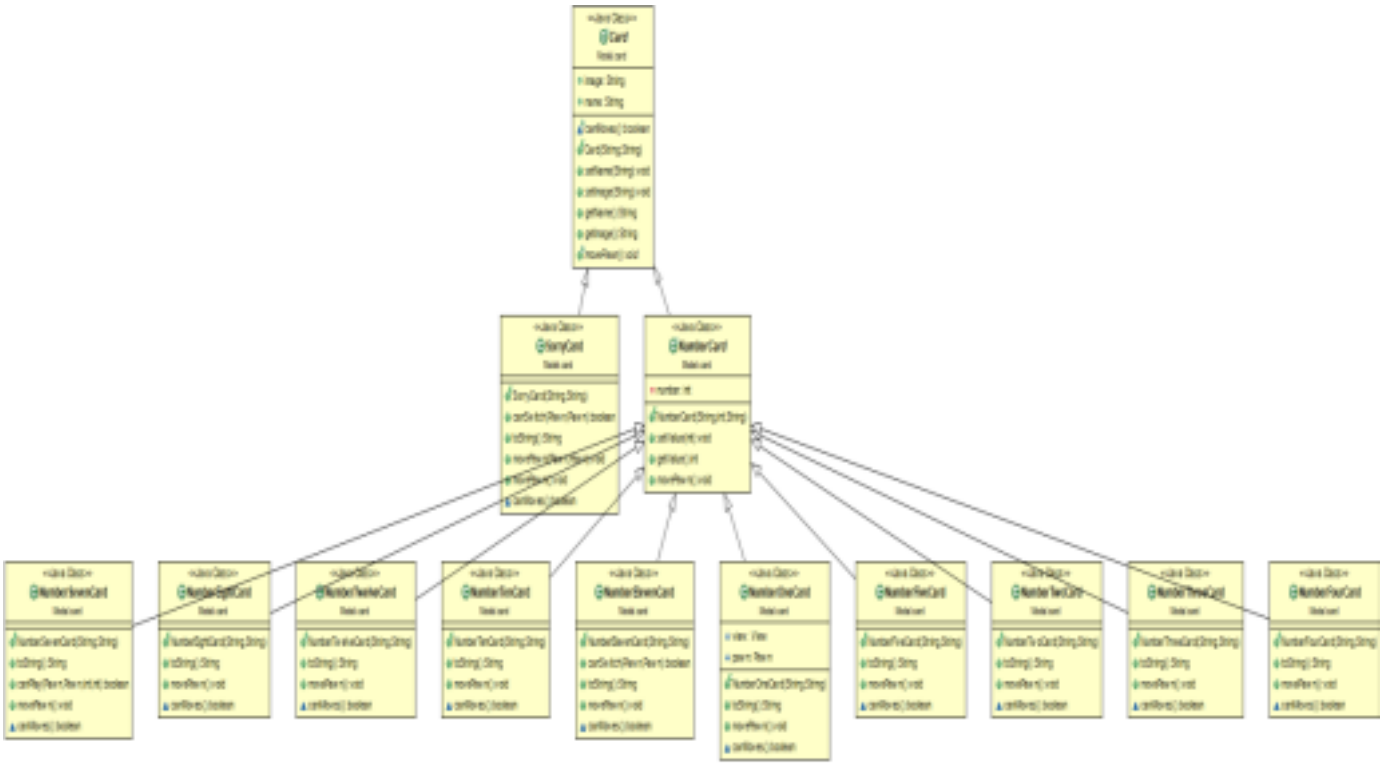
NumberTenCard , NumberElevenCard, NumberTwelveCard

These classes have access to class NumberCard with command super()

and initialize the attributes name, value(depending on the card) , image.

Each of this classes also have a toString() method that holds the string representation of each card.

UML for Package Card



4

-Abstract Class Square

Attributes

1) private int position //Position of the square.

2)private String name; // The name of the square.

Methods

public Square(int position, String name) //Constructor

1) public int getPosition() // Accessor (Selector) , gets the position of the square. 2) public int getName() // Accessor (Selector) , //gets the nameof the square. 3) public void setPosition(int position) // Transformer (Mutative) ,sets the position of the square. 4) public void setName(String name) //Transformer (Mutative) , sets the color of the square.

Classes that extend Square :

SimpleSquare, SafetyZoneSquare, StartSquare, HomeSquare, abstract class SlideSquare

These classes have access to class Square with command super() and initialize the attributes position, name.

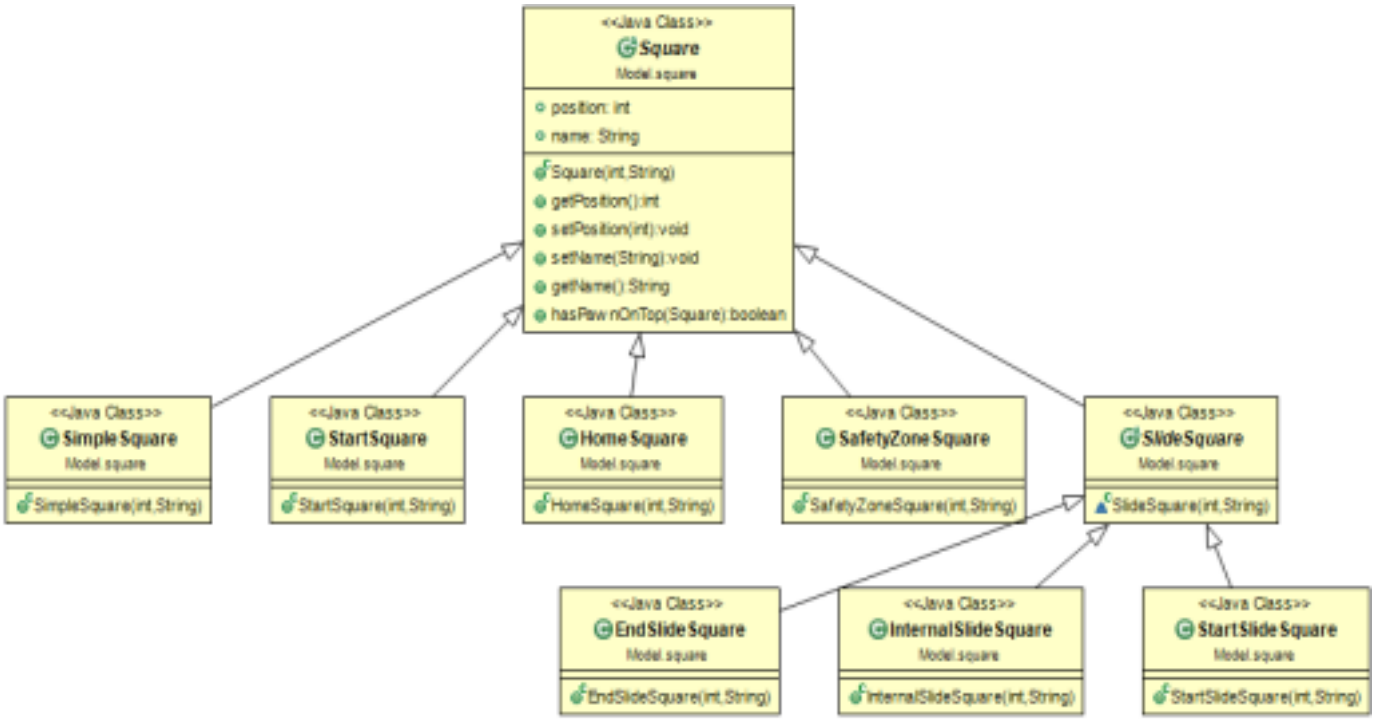
Classes that extend SlideSquare :

StartSlideSquare, InternalSlideSquare, EndSlideSquare

These classes have access to class SlideSquare with command super() and initialize the attributes : position, name.

5

UML for package Square



-Class Player

Attributes

1) private Color color //The color of the player.

2) private String name //The name of the player

3) private pawn pawn1 // pawn1 of the player

4) private pawn pawn2 // pawn2 of the player

Methods

Player(Color color, String name, Pawn pawn1, Pawn pawn2) // Constructor

1) getColor // Accessor(Selector) , gets the color

2) getName // Accessor (Selector) , gets the name

3) getPawn1 //Accessor (Selector) , gets the pawn1

4) getPawn2 // Accessor (Selector) , gets the pawn2

5) setColor // Transformer (Mutative) , sets the color
6) setName // Transformer (Mutative) , sets the name

7) setPawn1 // Transformer (Mutative) , sets the pawn1

8) setPawn2 // Transformer (Mutative) , sets the pawn2

9) public boolean isHisTurn(String name) //Checks if it is players turn, returns true or false.

Class Pawn

Attributes

1) private Color color; // The color of the pawn

2)private int position = -1; // The position of the pawn.

Methods

Pawn (Color color, int position) //Constructor

1)public void setColor(Color color) // Transformer (Mutative) , sets the color of the pawn.

2)public void setPosition(position) // Transformer (Mutative) , sets the position of the pawn..

3)public Color getColor() // Accessor(Selector) , return the color of the pawn. 4)public Square getPosition() //Accessor (Selector) , return the position of the pawn.

Class Deck

Attributes

1) ArrayList<Card> list //For the 52 cards in the deck.

2) int element = -1; //element for cards.

Methods

1)public Deck() //Constructor

2) public void intiCards() //Initialize the cards.

3) public ArrayList<Card> getGameCards() { // return the game cards.

4) public void setGameCards(ArrayList<Card> gameCards) // Transformer (Mutative) Sets the game card.

5) public void getNewCard // (Accessor) Let the user take a new card.

6) public void hasCardsLeft() // Checks if the deck has cards

7)public boolean canFold() //Checks if the player can press fold button

• **Package Controller**

This Class is the mind of the game. It is responsible for the connection between Model and View. It makes every action in order to make the game works in the right way. It is also responsible to show the end of the game and the winner.

-Class Controller

Attributes

1) public View view //a snapshot of class view

2) public Deck deck// a snapshot of class deck

3) public Player player1 //player1 of the game

4) public Player player2//player2 of the game

5) public Pawn[] pawns = new Pawn[4]; //an Array for the 4 pawns

6) public Square[] squares = new squares[73] //Array for the squares of the game.

7)private boolean hasStarted0 = false; //Check if pawns 0 has started.

8)private boolean hasStarted1 = false;//Check if pawns 1 has started.

9)private boolean hasStarted2 = false;//Check if pawns 2 has started

10)private boolean hasStarted3 = false;//Check if pawns 3 has started.

11)int count = 0; // Counter for the cards.

12) int dialogButton = JOptionPane.YES_NO_OPTION; //dialog for card11

13) int dialogResult; //the result of the dialogButton

14)private int turn = 1; //Variable for player turn (Player1 plays 1st by default).

15)private int cardscounter = 44; // counter for cards.

16)private Card card; //A variable to store the data of every card

17)private NumberCard c; //A 2nd variable to store the data of every card

18) card7 =0; //Variables that help us to use cards with number 7.

19) int k0 = -1; //Position of red pawn 1.

20) int k1 = -1; //Position of red pawn 2.

21) int k2 = -1;//Position of yellow pawn 1.

22) int k3 = -1; //Position of yellow pawn 2.

Methods

1) public static void main(String[] args) //Main method that creates a new controller snapshot. 2)public void initialize() // Initialize what is needed. (deck, view, pawns, squares, listeners). 3)initListeners() // Initialize all the listeners.

4) canStart0() //Check if red pawn1 can start

5) canStart1() //Check if red pawn2 can start

6) canStart2() //Check if yellow pawn1 can start

7) canStart3() //Check if yellow pawn2 can start

8) canMove0() //Check if the red pawn1 can move.

9) canMove1() //Check if the red pawn2 can move.

10) canMove2() //Check if the yellow pawn1 can move.

11) canMove3()//Check if the yellow pawn2 can move.

12) public void initPawns() //Initialize the pawns of the player.

13) public void initSquares() // Initialize the squares in the game.

14)public void removeAndReplaceCard() //Removes and replaces a card in the game.

• **Package view**

The package view includes all the graphics of the game. Specifically the view of the game includes the following :

-**Class View**

Attributes

1) private JFrame frame; // The frame of the game.

2)private JLayeredPane back; //The background of the game.

//For the right side

3)private JMenuBar menu; // The menu bar .

4)private JButton foldButton , Card1; // Fold Button and the button for receiving cards

5)private JButton Card2 //Button for current card.

6)private JTextArea infobox; // A box with information for every card.

7)private JLabel ReceiveCard,CurrentCard; // Descriptions that write “ReceiveCard” && “Current card”.

//For the left side

8)private JLabel logo //Sorry logo at the centre of the board

9)private JLabel redHome , yellowHome; // A label for red home and yellow home boxes.

10)private JLabel redStart, yellowStart // A label for red start and yellow start boxes. 11)

private JLabel background // a label for the background of the board

12) private JLabel txt1,txt2,txt3,txt4; //some texts inside the game

13)private JLabel[] square = new JLabel[100]; //the array for the squares in the game

14) private JButton[] pawns = new JButton[4] // An array of buttons for the pawns.

15)private JButton exit; //button to exit the game

16) private int index = 0; //variable that keeps the squares in the view.

17) private int i = 0; //temporary variable.

18) private int dialogButton = JOptionPane.YES_NO_OPTION;

//IMAGES

19) ImageIcon card1 = new ImageIcon("images/cards/backCard.png");

20) ImageIcon pawn0 = new ImageIcon("images/pawns/redPawn1.png");

21) ImageIcon pawn1 = new ImageIcon("images/pawns/redPawn2.png");

22) ImageIcon pawn2 = new ImageIcon("images/pawns/yellowpawn1.png");

23) ImageIcon pawn3 = new ImageIcon("images/pawns/yellowPawn2.png");

24) ImageIcon logos = new ImageIcon("images/sorryImage.jpg");

25) ImageIcon greenback = new ImageIcon("images/background.png");

Methods

1) public View() // Constructor of the game

2) public void initComponents() // Initialize all the components

3) public void initButtons() // Initialize all the buttons

4) public void updateCard(Card card) //Updates the card

5) public void updatePawn(int position, int p) //Update the pawn position depending on the player's action.

6) public void updateInfobox() //Updates the infobox

7) public JButton getCardButton() // Returns the Button that the player press.

8) public JLayeredPane getBack() // Returns the back

9) public JTextArea getInfobox() // Returns the infobox

10) public JButton getPawn0() //Returns the 1st red pawn.

11) public JButton getPawn1()// Returns the 2nd red pawn.

12) public JButton getPawn2()// Returns the 1st yellow pawn.

13) public JButton getPawn3()// Returns the 2nd yellow pawn.

14) public JButton getFoldbutton() // Returns fold button.

15) public JButton getExitButton() //Returns the exit button.

16) public void showMessage() //shows messages in the game.

17) public void resizeImages() //resize the images

18) public int returnMessage() //Returns the result of the dialog.

• Interaction between Classes – **UML Diagrams**

The class Card is the superclass for all the card classes that extend the class card.

In class **Square** all the classes that extend Square are connected.

In **Deck** are initialized all the cards in the deckÓ (so Deck talks with Card and with all of his subclasses).

In the Controller **class** all the squares are initialized (using class Square) , the Players (using class Player), the Pawns(using class Pawn). When the deck is empty, then all the cards are shuffled and initialized again using class Deck . Also every time a player makes a move, then the class View is responsible for moving the pawns on the screen. We can also see every time which card is showed in the screen and also the number and the description of that card (using Card or NumberCard classes).

• **Functionality/Bugs (Phase B)**

Fold doesn't work. Also in slides, only if the enemy's pawn is located in the beginning of in the end of the slide, it will go to the start position (while it passes an opponent pawn). So if an enemy's pawn is inside the slide and another pawn passes another pawn, it will not go to the start position.

Conclusion:

As a conclusion, it was an interesting project and despite the difficulties, it helped me learn new things and grow as a developer using OOP in Java.

George Gerasimos Leventopoulos **AM 4152**

THE END