ECSI402 - Programming Principles 01

Coursework 02 – Slot Machine

Module Leader: Mr. P. Guganathan

Date of submission: 27th November 2017

First Name :Samararathna

Last Name : Kodikara

IIT Student Number : 2016419

UOW Login : w16545713

Table of Contents

Itroduction

This game is a slot machine game. In the reel it will appear one of 6 different symbols. In the beginning user has 10 credits by default and user has to bet credits and spin the reels by clicking spin button. When all 3 images are same user will get credits by bet credits multiple by image value . every image has a unique value. If two images are equal user also win and get credits. If any of image not equal user has lose then he has to bet again and spin the reels.

If user's credits 0 he can add coins by clicking addcoin button. If he wants to check his details user can show them by clicking statistics . it will show number of wins and number of loses that he achieved. And also from a pie chart . If user needs to save his details by clicking save statistics user can save them in a file.

Requirements

Functional Requirements

- ADD COIN BUTTON

Designed so that user can add credits to continue playing the game as he/ she require.

- CREDIT AREA

Label is used to inform the user the number of credits that he/she has left to play with. Initial when the program starts 10 credits are given out free.

- BET ONE BUTTON

Allows the user to bet a single credit.

- BET MAX BUTTON

Designed to bet a maximum number of credits which is permanently set to 3.

- RESET BUTTON

As required this allows the user to return the amount he/she has bet to the credits that he/she still has.

- SPIN BUTTON

This causes the three Reels to spin.

- BET AREA

Informs the user of the number of credits that he/she is betting on the current game.

- THREE REELS

When the spin button is pressed, all three reels will start spinning. To stop the spinning the user just have to click on one.

- STATISTICS BUTTON

This allows the user to check his/her number of wins, losses, and the average number of credits that she/he has netted per game. If the average is positive, then the player has been, on average, winning more credits than losing. If it is negative, then the player has been, on average, losing more credits than winning.

- SAVE STATISTICS

  This will save user's statistics in a file.

Non-Functional Requirements:

- User-Friendliness

Java GUI application is designed with attractive buttons and proper layout which enables user in understanding each function of the Slot Machine Application effectively.

- Accuracy

The accuracy of the output data is so much precise.

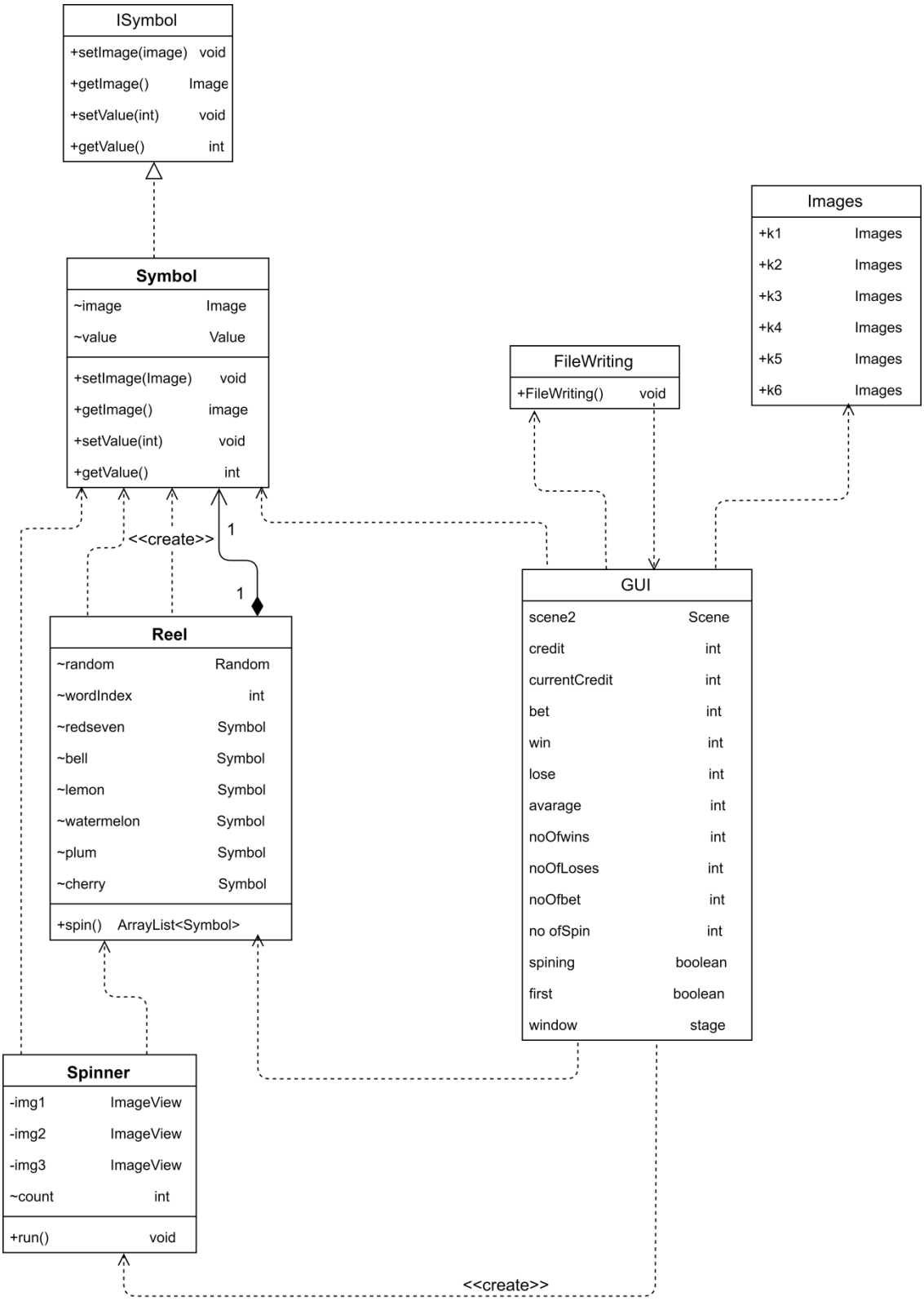
- Recoverability

Personal performance in the Slot Machine Application is recorded in a text file in users' desktop. As the Statistics are saved on the text file user can easily view on the data for further calculation or so.

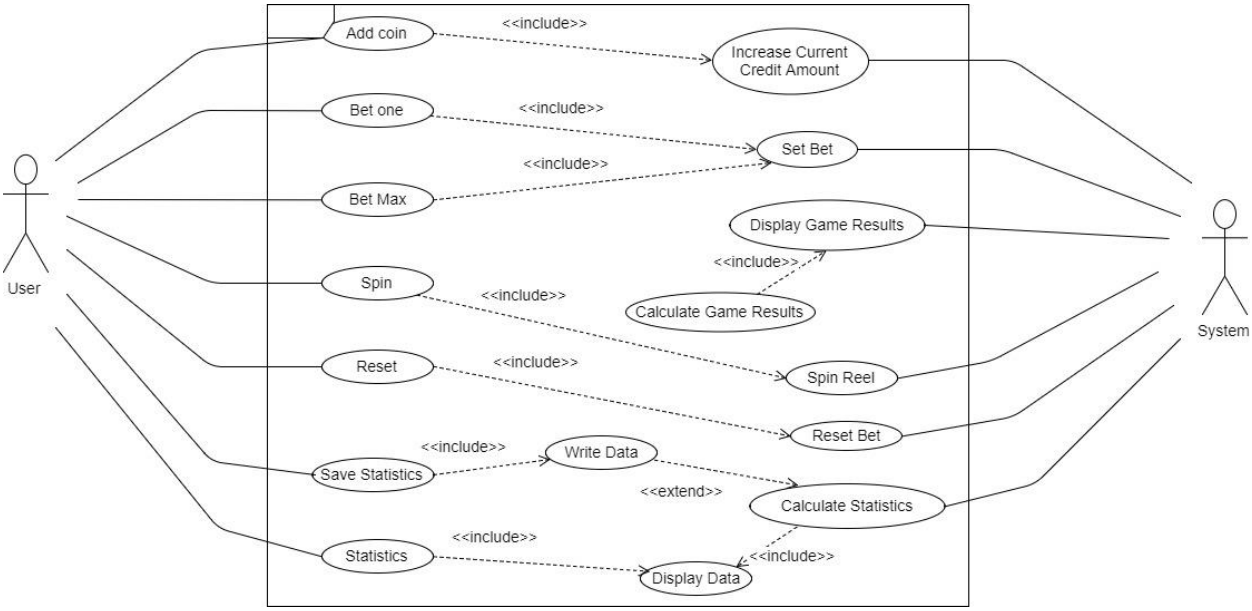
- Performance

Program is running under Java 8 which is faster. And the coding format enables the program to run smoothly and effectively.

UML Diagram :


ISymbol
+setImage(image) void
+getImage() Image
+setValue(int) void
+getValue() int
Symbol
~image Image
~value Value
+setImage(Image) void
+getImage() image
+setValue(int) void
+getValue() int
Images
+k1 Images
+k2 Images
+k3 Images
+k4 Images
+k5 Images
+k6 Images
FileWriting
+FileWriting() void
<<create>>
1
1
Reel
~random Random
~wordIndex int
~redseven Symbol
~bell Symbol
~lemon Symbol
~watermelon Symbol
~plum Symbol
~cherry Symbol
+spin() ArrayList<Symbol>
GUI
scene2 Scene
credit int
currentCredit int
bet int
win int
lose int
avarage int
noOfwins int
noOfLoses int
noOfbet int
no ofSpin int
spining boolean
first boolean
window stage
Spinner
-img1 ImageView
-img2 ImageView
-img3 ImageView
~count int
+run() void
<<create>>

Usecase Diagram:



Code

GUI class

```
package sample;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
```

```
import javafx.scene.layout.*;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import java.util.ArrayList;
import java.io.IOException;


public class GUI extends Application {

   Scene  scene2;
   static int credit = 10;
   static int currentCredit=0;
   int bet = 0;
   static int win;
   static int lose;
   static double avarage;
   static int noOfwins =0;
   static int  noOfLoses = 0;
   int noOfbet = 0;
   int noOfSpin = 1;

   static boolean spining = false;
   static boolean first = true;
   Stage window;
   public static final int IMAGESIZE=100;


   public static void main(String[] args) {
      launch(args);
   }

   @Override
   public void start(Stage primaryStage) throws Exception {
      window = primaryStage;
      window.setTitle("Slot Machine");

      //GridPane with 10px padding around edge
      GridPane grid = new GridPane();
      grid.setAlignment(Pos.CENTER);
      grid.setPadding(new Insets(10, 10, 10, 10));
      grid.setVgap(10);
```

```
grid.setHgap(10);

ColumnConstraints colum1 = new ColumnConstraints(100,150,100);
ColumnConstraints colum2 = new ColumnConstraints(200,300,Double.MAX_VALUE);
RowConstraints rowEmpty = new RowConstraints();
RowConstraints row5 = new RowConstraints(75,100,200);

//vertical & horizontal grow priority
colum1.setHgrow(Priority.ALWAYS);
colum2.setHgrow(Priority.ALWAYS);
row5.setVgrow(Priority.ALWAYS);

//Image1 imageview1 - constrains use (child, column, row)
ImageView imageView1 = new ImageView(Images.k7);
imageView1.setFitHeight(IMAGESIZE);
imageView1.setFitWidth(IMAGESIZE);
GridPane.setConstraints(imageView1, 1, 0);

//Image2 imageview2
ImageView imageView2 = new ImageView(Images.k7);
imageView2.setFitWidth(IMAGESIZE);
imageView2.setFitHeight(IMAGESIZE);

GridPane.setConstraints(imageView2, 2, 0);

//Image3 imageview3
ImageView imageView3 = new ImageView(Images.k7);
imageView3.setFitHeight(IMAGESIZE);
imageView3.setFitWidth(IMAGESIZE);
GridPane.setConstraints(imageView3, 3, 0);

//spin
Button spin = new Button("Spin");
spin.setFont( Font.font("Verdana",18));
GridPane.setConstraints(spin,2,1);
spin.setPrefHeight(50);
spin.setMinSize(100,10);
//span

Button betone = new Button("Bet One");
spin.setFont( Font.font("Verdana",18));
```

```
GridPane.setRowIndex(betone,2);
GridPane.setColumnIndex(betone,1);
betone.setPrefHeight(50);
betone.setMinSize(100,10);

Button betmax = new Button("Bet Max");
spin.setFont( Font.font("Verdana",18));
GridPane.setConstraints(betmax,3,2);
betmax.setPrefHeight(50);
betmax.setMinSize(100,10);

//add coin
Button addcoin = new Button("Add Coin");
spin.setFont(Font.font("Verdana",18));
spin.setOnAction(e-> System.out.println("Add a coin to the credit")); //have to do
GridPane.setConstraints(addcoin,1,3);
addcoin.setPrefHeight(50);
addcoin.setMinSize(100,10);


Button reset = new Button("Reset");
spin.setFont(Font.font("Verdana",18));
GridPane.setConstraints(reset,2,3);
reset.setPrefHeight(50);
reset.setMinSize(100,10);

Button statistics = new Button("Statistics");
spin.setFont(Font.font("Verdana",18));
GridPane.setConstraints(statistics,3,3);
statistics.setPrefHeight(50);
statistics.setMinSize(100,10);

// Label
Label credtLabel = new Label("  Credit Area:  ");
GridPane.setConstraints(credtLabel, 0, 2);
credtLabel.setPrefHeight(50);
credtLabel.setMinSize(100,10);

Label currentlbl = new Label();
GridPane.setConstraints(currentlbl, 0, 3);
credtLabel.setPrefHeight(50);
```

```
        credtLabel.setMinSize(100,10);

        Label credtLabel2 = new Label("  Bet Area:   ");
        GridPane.setConstraints(credtLabel2, 4, 2);
        credtLabel2.setPrefHeight(50);
        credtLabel2.setMinSize(100,10);

        Label betlbl = new Label();
        GridPane.setConstraints(betlbl, 4, 3);
        credtLabel.setPrefHeight(50);
        credtLabel.setMinSize(100,10);

        //Add everything to grid

grid.getChildren().addAll(imageView1,imageView2,imageView3,spin,betone,betmax,addcoin,reset,statis
tics,credtLabel,credtLabel2,currentlbl,betlbl);

        Scene scene = new Scene(grid, 700, 350);
        scene.getStylesheets().add("Styles/s.css");
        window.setScene(scene);
        window.show();

        Thread t1 = new Thread(new Spinner(imageView1,imageView2,imageView3));
        t1.isDaemon();

        currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
        betlbl.setText("  Bet credits  "+(Integer.toString( bet)));

        spin.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                if (bet >0){
                    if(!spining && first){
                        t1.start();
                        first = false;
                        noOfSpin= (noOfSpin-1)+1;
                    }else if(!first){
                        t1.resume();
                    }
                }else {
                    Alert alert = new Alert(Alert.AlertType.WARNING);
```

```
            alert.setHeaderText("Please bet credits first!");
            alert.show();
        }
    }
});


imageView1.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        t1.suspend();
        Image img1 = imageView1.getImage();
        Image img2 = imageView2.getImage();
        Image img3 = imageView3.getImage();


        if ((img1==img2) && (img1==img3)&& (img2==img3)){
            currentCredit =getValue(img1)*bet;
            credit+=currentCredit;
            bet = 0;
            GUI.noOfwins+=1;
            System.out.println(noOfwins);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("You have won "+currentCredit+" credits!");
            alert.show();
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
            win+=1;
        }
        else if(img1==img2 ){
            currentCredit =getValue(img1)*bet;
            credit+=currentCredit;
            bet = 0;
            GUI.noOfwins+=1;
            System.out.println(noOfwins);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("You have won "+currentCredit+" credits!");
            alert.show();
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
            win+=1;
```

```
        }
        else if(img1==img3 ){
            currentCredit =getValue(img1)*bet;
            credit+=currentCredit;
            bet = 0;
            GUI.noOfwins+=1;
            System.out.println(noOfwins);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("You have won  "+currentCredit+" credits!");
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
            alert.show();
            win+=1;
            System.out.println(currentCredit);
        }
        else if(img3==img2 ){
            currentCredit =getValue(img1)*bet;
            credit+=currentCredit;
            bet = 0;
            GUI.noOfwins+=1;
            System.out.println(noOfwins);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("You have won  "+currentCredit+"  credits!");
            alert.show();
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
            win+=1;
        }else {
            bet = 0;
            GUI.noOfLoses+=1;
            System.out.println(noOfLoses);
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("You have lose!");
            alert.show();
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
            lose+=1;
        }
    }
});
```

```
betone.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        if (credit-1>=0){
            credit-=1;
            bet +=1;
            noOfbet+=1;
            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));

        }else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setHeaderText("Insufficient credits!");
            alert.setContentText("You have run out credits!");
            alert.show();
        }
    }
});


reset.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        if (bet>0) {

            credit += bet;
            bet =0;

            currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
            betlbl.setText("  Bet credits  "+(Integer.toString( bet)));
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setContentText("Can't reset you haven't bet coins.!");
            alert.show();
        }
    }
});



betmax.setOnMouseClicked(new EventHandler<MouseEvent>() {
```

```
    @Override
    public void handle(MouseEvent event) {
      if (credit>=3){
        credit-=3;
        bet +=3;
        noOfbet+=1;
        currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
        betlbl.setText("  Bet credits  "+(Integer.toString(bet)));

      }else if(credit>3 & credit>0) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setHeaderText("not enough credits! \n  Please try to Bet one");
        alert.show();
      }else {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setHeaderText("Insufficient credits!");
        alert.setContentText("You have run out credits!");
        alert.show();
      }
    }
});


addcoin.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
      if (credit<=0){
        credit+=1;
        currentlbl.setText("  Current credits  "+(Integer.toString(credit)));
        betlbl.setText("  Bet credits  "+(Integer.toString(bet)));
      }else {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setHeaderText("You have enough credits to play!");
        alert.setContentText("Please bet credits and play!");
        alert.show();
      }
    }
});


//statistics.setOnMouseClicked(e -> window.setScene(scene2));
```

```
statistics.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        //System.out.println("loses" +noOfLoses);
        //System.out.println("credits" + currentCredit);
        GridPane grid2 = new GridPane();
        grid2.setAlignment(Pos.CENTER);
        grid2.setPadding(new Insets(10, 10, 10, 10));

        Label labelwin= new Label("   wins :   ");
        labelwin.setFont( Font.font("Verdana",18));
        GridPane.setConstraints(labelwin,0,0);
        labelwin.setPrefHeight(50);
        labelwin.setMinSize(100,10);

        Label labelA1= new Label();
        labelA1.setText(Integer.toString(currentCredit));
        labelA1.setFont( Font.font("Verdana",18));
        GridPane.setConstraints(labelA1,1,0);
        labelA1.setPrefHeight(50);
        labelA1.setMinSize(100,10);
        labelA1.setText(Integer.toString(win));

        Label labelLose= new Label("   Loses :   ");
        GridPane.setConstraints(labelLose,0,1);
        labelLose.setFont( Font.font("Verdana",18));
        labelLose.setPrefHeight(50);
        labelLose.setMinSize(100,10);


        Label labelB1= new Label();
        labelB1.setText("   "+(Integer.toString(noOfLoses)));
        GridPane.setConstraints(labelB1,1,1);
        labelB1.setFont( Font.font("Verdana",18));
        labelB1.setPrefHeight(50);
        labelB1.setMinSize(100,10);


        Label labelC= new Label("   Average credits :   ");
        GridPane.setConstraints(labelC,0,2);
        //labelC.setText(Integer.toString((int)((double)win/(double)noOfSpin)));
        labelC.setFont( Font.font("Verdana",18));
```

```
        labelC.setPrefHeight(50);
        labelC.setMinSize(150,10);

        Label labelC1= new Label();
        avarage=((double)(credit - noOfbet))/noOfSpin;
        //labelC1.setText("   "+(Integer.toString(avarage)));
        GridPane.setConstraints(labelC1,1,2);
        labelC1.setFont( Font.font("Verdana",18));
        labelC1.setPrefHeight(50);
        labelC1.setMinSize(100,10);
        labelC1.setText(Double.toString(avarage));

        PieChart pieChart = new PieChart();
        pieChart.setData(dataInput());
        GridPane.setConstraints(pieChart,2,0,2,2);

        Button btnA= new Button("Save statistics");
        GridPane.setConstraints(btnA,0,3);
        labelC1.setFont( Font.font("Verdana",18));
        labelC1.setPrefHeight(50);
        labelC1.setMinSize(100,10);

        Button btnB= new Button("   OK   ");
        btnB.setOnAction(e -> window.setScene(scene));
        GridPane.setConstraints(btnB,1,3);
        labelC1.setFont( Font.font("Verdana",18));
        labelC1.setPrefHeight(50);
        labelC1.setMinSize(180,10);

    grid2.getChildren().addAll(labelwin,labelA1,labelLose,labelB1,labelC,labelC1,btnA,btnB,pieChart);
        scene2 = new Scene(grid2, 800, 800);
        scene.getStylesheets().add("Styles/g.css");
        window.setScene(scene2);

        btnA.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {

                try {
                    FileWriting.FileWriting();

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

```
        });

    }

    public static int getValue(Image image) {
        ArrayList<Symbol> arr = Reel.spin();
        for(Symbol s : arr){
            if(image==s.getImage()){
                System.out.println(" Value is : " + s.getvalue());
                return s.getvalue();
            }
        }
        return 0;
    }

    private ObservableList<PieChart.Data> dataInput() {
        ObservableList<PieChart.Data> answer = FXCollections.observableArrayList();
        answer.addAll(
                new PieChart.Data("Wins: ",currentCredit),
                new PieChart.Data("Losses: ",noOfLoses)
        );
        return answer;
    }

}
```

Spinner class

```
package sample;
import javafx.scene.image.ImageView;

import java.util.ArrayList;

public class Spinner implements Runnable{

    private ImageView img1;
    private ImageView img2;
    private ImageView img3;

    public Spinner(ImageView img1, ImageView img2, ImageView img3) {
        this.img1 = img1;
        this.img2 = img2;
        this.img3 = img3;
    }

    int count=0;
    @Override
```

```
    public void run() {
        ArrayList<Symbol> reel1= Reel.spin();
        ArrayList<Symbol> reel2= Reel.spin();
        ArrayList<Symbol> reel3= Reel.spin();


        while (true){
            Symbol cur1 = reel1.get(count);
            img1.setImage(cur1.getImage());

            Symbol cur2 = reel2.get(count);
            img2.setImage(cur2.getImage());

            Symbol cur3 = reel3.get(count);
            img3.setImage(cur3.getImage());

            if (count<5) {
                count++;
            } else {
                count=0;
            }

            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Reel class

```
package sample;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;

public class Reel{
  //  public static void main(String[] args) {
        Random random = new Random();
        int wordIndex = random.nextInt(7);
    //}
```

```
    static Symbol redseven = new Symbol(new Image("Images/redseven.png"), 7);
    static Symbol bell = new Symbol(new Image("Images/bell.png"), 6);
    static Symbol lemon = new Symbol(new Image ("Images/lemon.png"), 3);
    static Symbol watermelon = new Symbol(new Image("Images/watermelon.png"), 5);
    static Symbol plum = new Symbol(new Image("Images/plum.png"), 4);
    static Symbol cherry = new Symbol(new Image("Images/cherry.png"), 2);

  public static ArrayList<Symbol> spin() {
      ArrayList<Symbol> symbol_array = new ArrayList<>();
      symbol_array.add(redseven);
      symbol_array.add(bell);
      symbol_array.add(watermelon);
      symbol_array.add(lemon);
      symbol_array.add(plum);
      symbol_array.add(cherry);

      Collections.shuffle(symbol_array);
      return symbol_array;
}}
```

FileWriting class :

```
package sample;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class FileWriting {

  public static void FileWriting() throws IOException {
    DateFormat dateFormat = new SimpleDateFormat("yy_MM_dd,HH_mm_ss");
    Date date = new Date();
    String file = String.valueOf(dateFormat.format(date));

    try {
      FileWriter fw = new FileWriter(file);
      PrintWriter pw = new PrintWriter(fw);
      pw.println("Total number of wins   : " + GUI.noOfwins);
      pw.println("Total number of losses : " + GUI.noOfLoses);
      pw.println("Avarage of credits : " + GUI.avarage);
      pw.close();
    } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }}
```

Images class :

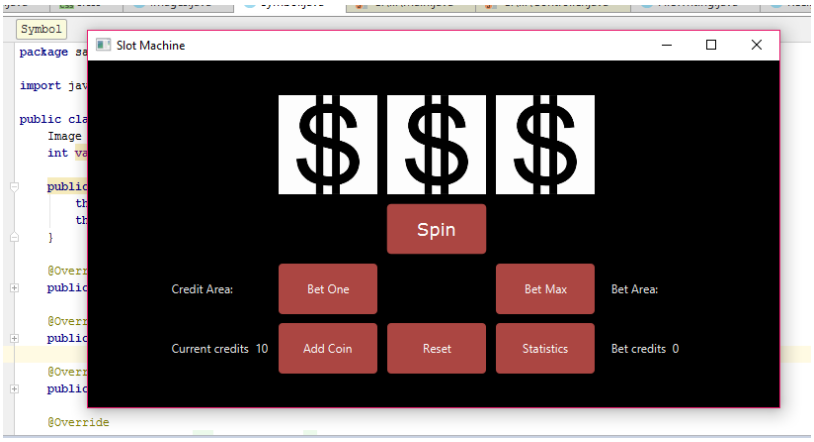
```
package sample;

import javafx.scene.image.Image;


public class Images {
    public static Image k1 = new Image("/Images/bell.png");
    public static Image k2 = new Image("/Images/cherry.png");
    public static Image k3 = new Image("/Images/lemon.png");
    public static Image k4 = new Image("/Images/plum.png");
    public static Image k5 = new Image("/Images/redseven.png");
    public static Image k6 = new Image("/Images/watermelon.png");
    public static Image k7 = new Image("/Images/dolar.png");

}
```

Symbol class :

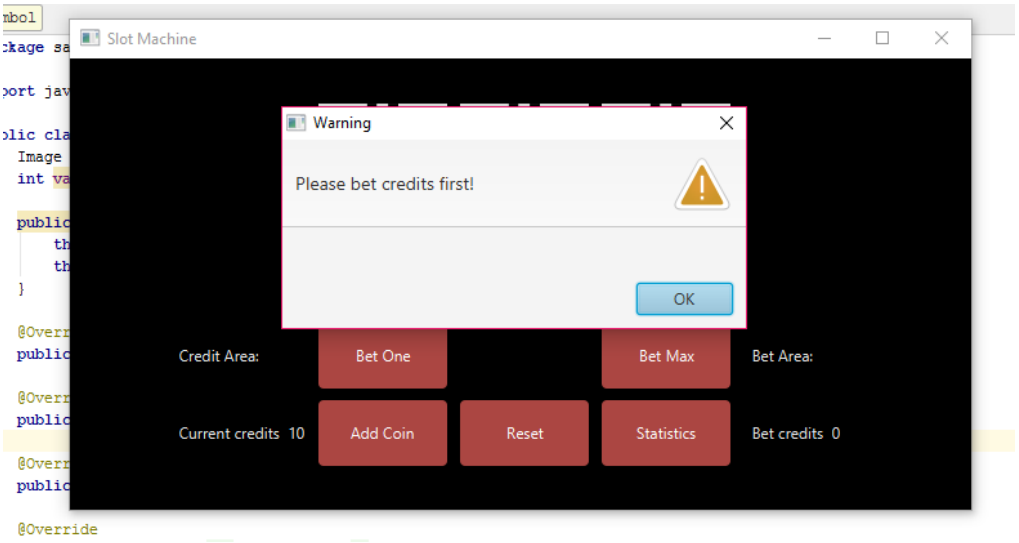
```
package sample;

import javafx.scene.image.Image;

public class Symbol implements ISymbol {
    Image image;
    int value;

    public Symbol(Image image, int value) {
        this.image = image;
        this.value = value;
    }

    @Override
    public void setImage(Image image) {
        this.image= image;
    }

    @Override
    public Image getImage() {
        return image;
    }
```

```
    @Override
    public void setValue(int value) {
        this.value= value;
    }

    @Override
    public int getvalue() {
        return value;
    }
}
```

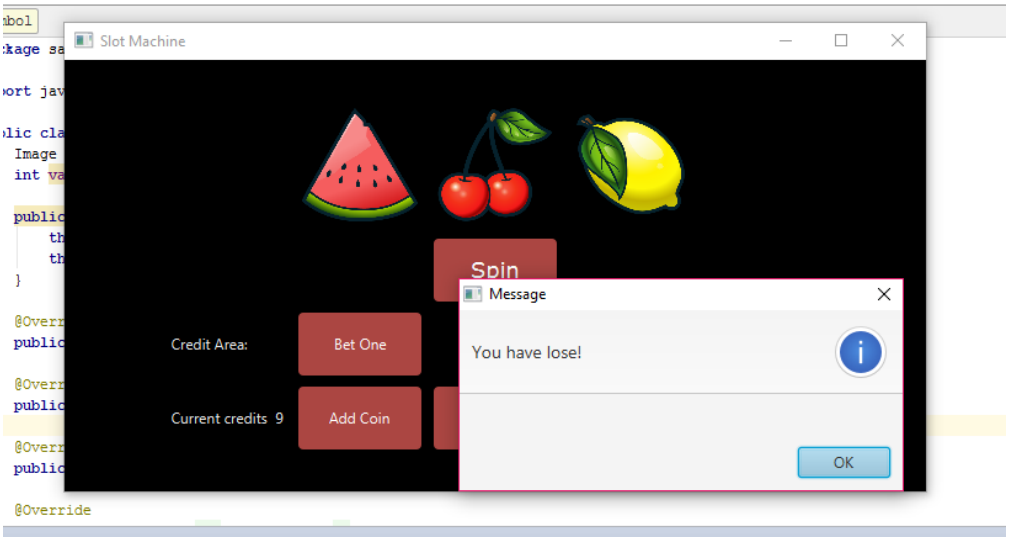
Screen shots :



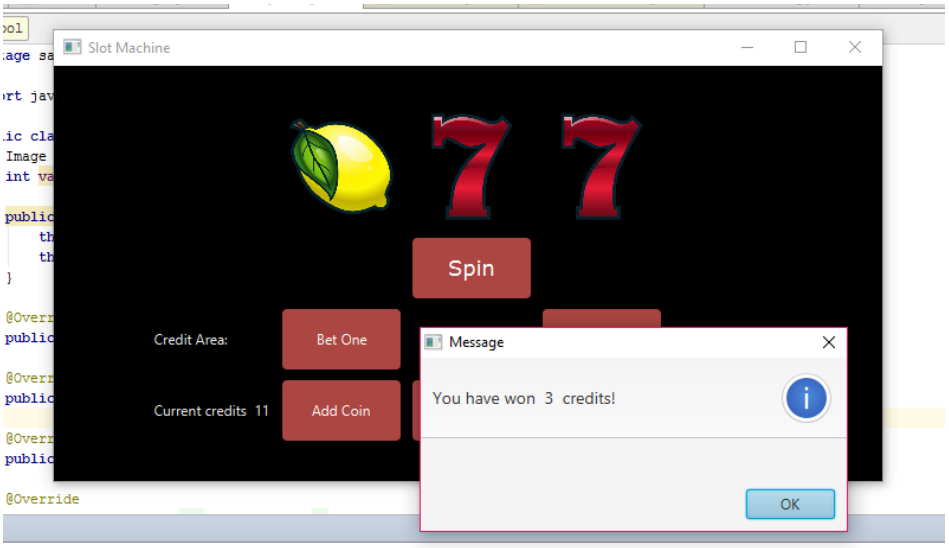
When user try to spin without bet credits:



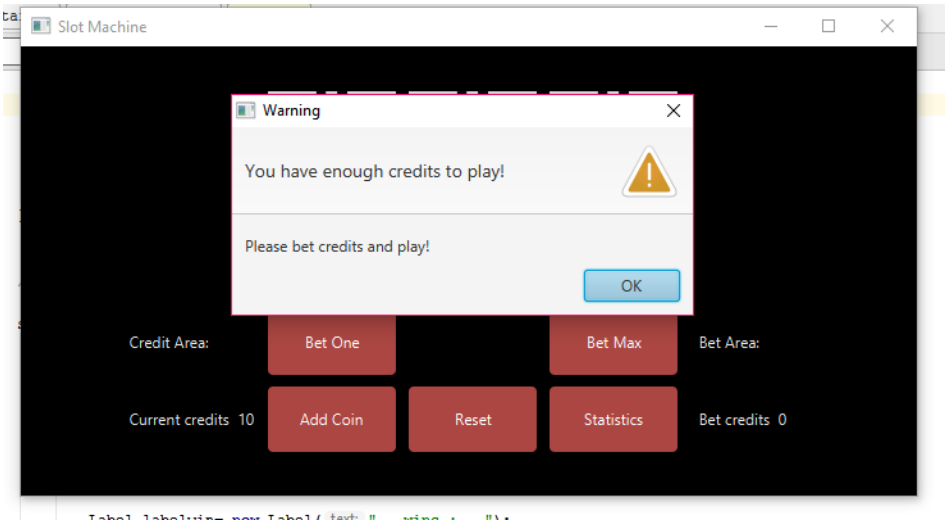
When non of reels are not equal :



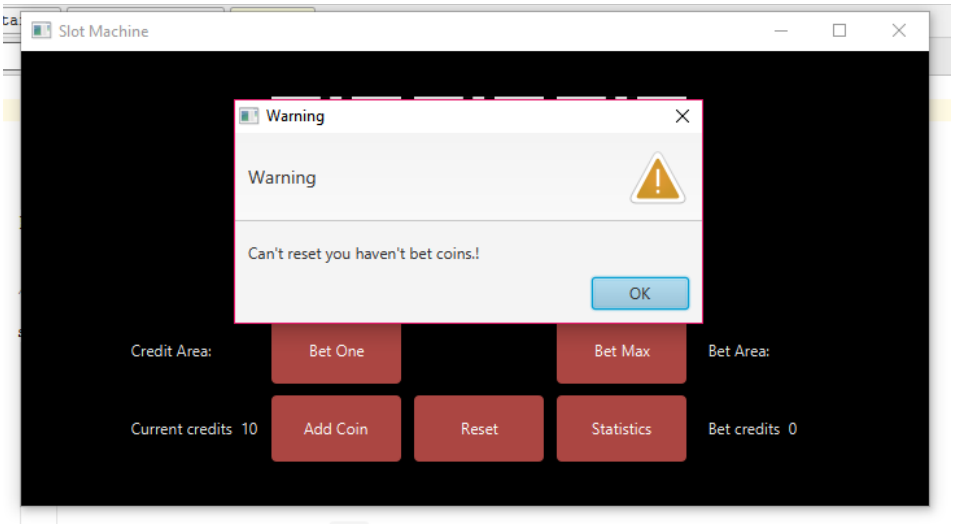
When two or three reels are matching :



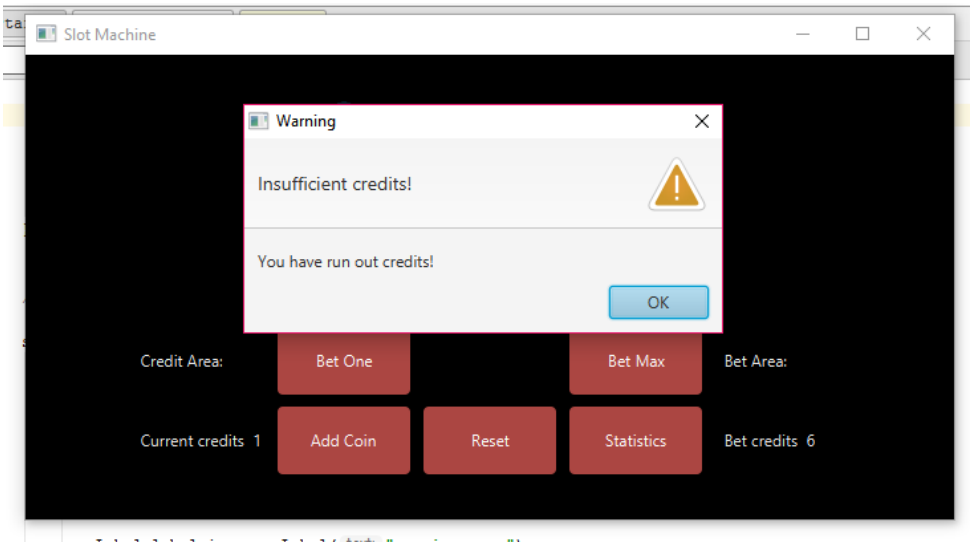
When user try to add more credits when he has money already:



When user try to reset without bet credits:



When user try to bet credits after runout of credits:



File writing: