INFORMATICS INSTITUTE OF TECHNOLOGY

IN COLLABORATION WITH

UNIVERSITY OF WESTMINSTER (UOW)

BENG (HONS) SOFTWARE ENGINEERING

MODULE: 5COSCOO1W Object Oriented Programming

MODULE LEADER : MR. P.GUGANATHAN

COURSE WORK – 02

REPORT

SUBMISSION DATE : 5TH DECEMBER 2018

USERNAME: W1654191

UOW ID: W16541915/1

STUDENT ID: 2016323

GROUP : B

STUDENT FIRST NAME: KAJENDRAN

STUDENT SURNAME: CHANDRESWARAN

Table of Contents

Introduction

Functional Requirement

Non Functional requirement

Class Diagram

Codes

Screen shots

Payout Ratio

Introduction

This is a GUI base Slot machine game application. This slot machine GUI has three reels which have six different symbols. The player can play the game by spinning the reels if the symbols of three reels match, it will count as a win. To play the slot game the player should have credits and he have chose the bet range also. If the player won the round the bet amount and winning amount will be added to the remaining credits. Winning amount will be calculated according to the values that already assigned to the symbols. Reels are implemented with Thread.

The slot machine GUI has following components,

- Bet Once, this button will allow the user to bet one credit from initial credit amount to play the game one round and if he won the round winning credit will add to his remaining credits (winning credits will be calculated by the symbol values).

- Bet Max, this button will allow the user to bet three credit from the initial credit amount to play the game one round.

- Add coin, this button will allow the user to increase the credit amount by one to stay in the game, if credit amount goes to zero user will unable to play the game (user can add the coin if when bet amount is zero and his credit level is lower than 3).

- Reset, this button will allow the user to cancel the last betting (the bet amount will be added back to the credit).

- Spin, this button will allow the user to play the game (System will allow the user to play when he set the bet amount).

- Statistics, this button will allow the user to review his performance in the game (user can be able to watch the winning and losing).

- Print, this button will allow the user to print game statistics in a text file.

- Credit area, this label will show the credits that user have at the moment and when every time game starts user will have 10 credits.
- Bet area, this label will show the current bet amount at the moment and player can cancel the bet amount before playing the game.

- Three reels, this will hold three symbol when user click the spin button it will started rolling and when user click the reels it will stop.

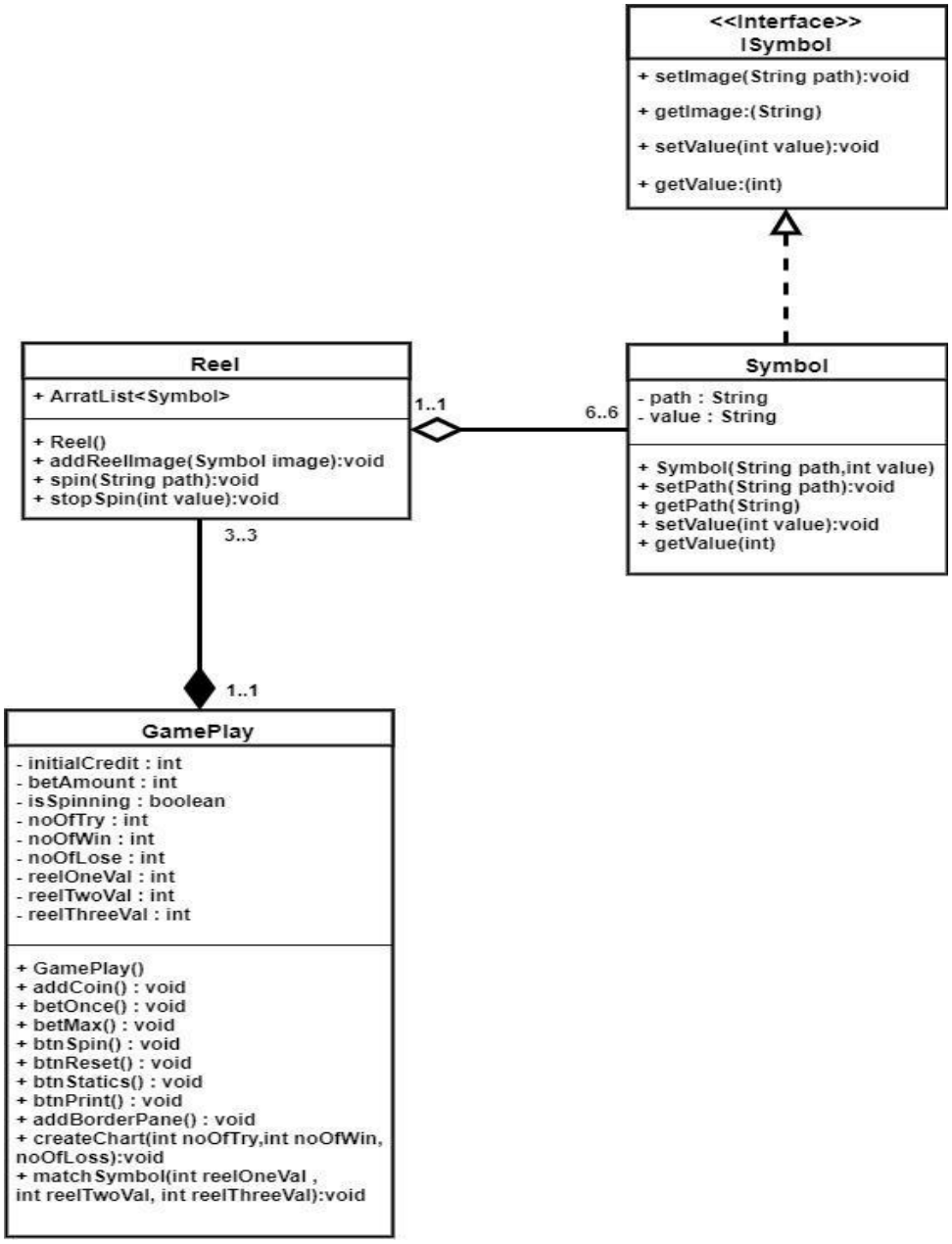
Functional Requirement

1- Adding coin, although user gets ten credits at the starting and when credit level comes under three.
2- Credit area, this show the credit amount after every betting and every rounds finished.
3- Betting one coin, user can bet only one coin at a time while click this button.
4- Betting three coins, user can able to bet three coins at a time while click this button.
5- Reset Credit, user can able to reset the betting amount before playing the game.
6- Statistic, user can able to view the player statistic in number detail and pie chart view.
7- Print, user can able to print the game play statistic in a text file formet.

Non Functional requirement

- Create a user friendly GUI.
- Usability of codes.
- Readability of codes.
- Quality of codes.

Class Diagram



Codes

- **ISymbol.java (interface)**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package coursework;

/**
 *
 * @author Asus
 */
public interface ISymbol {

    void setImage(String image); // this method to store the path of symbols(url of
the image will be stored in).

    String getImage(); // get and display the symbols

    void setValue(int v); // set the credit value of the each symbols

    int getValue(); // return the values of the symbols
}
```

- **Symbol class**

```
package coursework;

public class Symbol implements ISymbol{

    private String path=null; // to store the path of the images
    private int value; // to store the coin credits of the images

    public Symbol(String image,int value){ // constructor
        this.path = image;
        this.value = value;
    }
    public void setImage(String path){ // This method will add the path of the iamges
which stores in the folder.
        this.path = path;
    }

    public String getImage() { // This method is to retrive the image path.
        return path;
    }

    public void setValue(int v) { // this method will set the values to each images
        this.value = v;
    }
    public int getValue() { // this method will get the value of the images to
calculate the winner.
        return value;
    }
}
```

-
-
-

- **Reel class**

```
- package coursework;

  import java.util.ArrayList;
  import java.util.Collections;
  /**
   *
   * @author Asus
   */
  public class Reel{

      ArrayList<Symbol> images = new ArrayList<Symbol>();

      public Reel(){
          addreelImage();
      }

      public void addreelImage() { // intializing the array and adding the images
          Symbol bell = new Symbol("Images/bell.png", 6);
          Symbol cherry = new Symbol("Images/cherry.png", 2);
          Symbol lemon = new Symbol("Images/lemon.png", 3);
          Symbol plum = new Symbol("Images/plum.png", 4);
          Symbol redSeven = new Symbol("Images/redseven.png", 7);
          Symbol waterMelon = new Symbol("Images/watermelon.png", 5);
          images.add(bell);
          images.add(cherry);
          images.add(lemon);
          images.add(plum);
          images.add(redSeven);
          images.add(waterMelon);
      }

      public ArrayList Spin(ArrayList images){//This method will suffle the
  images which stores in the arraylist.
          Collections.shuffle(images);
          return images;
      }

     public void stopSpin(boolean spin) {
          spin = false;
          Thread.currentThread().stop();
      }

     public Symbol getReel(int value) { // this mehod is to get the value of the
  revelent images when the spin is stoped.
         for(int i = 0; i < images.size(); i++) {
             if(images.get(i).getValue() == value) {
                 return images.get(i);
             }
         }
         return null;
     }
  }
```

- **GamePlay class**

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package coursework;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
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.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author Asus
 */
public class GamePlay extends Application {
    private static int initialCredit = 10; // game starts with 10 coins per players
initially.
    private static int betAmount; // all the bet amount will be stored.
    static boolean isSpining = true; // cheak the reels are spinning or not , to stop
the threads.

    private  static  int noOfTry = 0; // stores the number of rounds played by the
perticular player.
    private static int noOfWin = 0; // stores the number of rounds won by the
perticular player.
    private static int onOfLoss = 0; // stores the number of rounds won by the
perticular player.

    private int reelOneVal = 0; // stores the values of sybmols when the reel one
stops rolling.
    private int reelTwoVal = 0; // stores the values of sybmols when the reel two
stops rolling.
    private int reelThreeVal = 0; // stores the values of sybmols when the reel three
stops rolling.

    @Override
    public void start(Stage primaryStage) {

        Reel reel1 = new Reel(); // creating Reel objects
        Reel reel2 = new Reel();
        Reel reel3 = new Reel();
```

```
        Label lblTitle = new Label("WELCOME TO SLOT CASINO"); // to set the the title
label.
        lblTitle.setStyle("-fx-border-color: white;");
        lblTitle.setStyle("-fx-font-size: 12pt;");
        lblTitle.setStyle("-fx-text-fill: white;");
        BorderPane border = new BorderPane(); // creating a border pane to place all
the objects.

        HBox addTitle = new HBox(); // creating a HBox to place the title label and
set it to top of the border pane.
        addTitle.setPadding(new Insets(40, 40, 40, 650));
        addTitle.setSpacing(0);
        addTitle.setStyle("-fx-background-color: #336699;");
        border.setTop(addTitle);
        addTitle.getChildren().add(lblTitle);

//        Image image1 = new Image("Images/bell.png");
//        Image image2 = new Image("Images/redseven.png");
//        Image image3 = new Image("Images/cherry.png");

        ImageView iView1 = new ImageView(reel1.images.get(0).getImage()); // creating
three image views to hold the Symbols in the three wheels.
        ImageView iView2 = new ImageView(reel2.images.get(0).getImage());
        ImageView iView3 = new ImageView(reel3.images.get(0).getImage());

        iView1.setFitHeight(410);
        iView1.setFitWidth(410);

        iView2.setFitHeight(410);
        iView2.setFitWidth(410);

        iView3.setFitHeight(410);
        iView3.setFitWidth(410);

        Label lblReel1 = new Label(); // setting the image views to the labels
        lblReel1.setGraphic(iView1);
        lblReel1.setStyle("-fx-border-color: white;");

        Label lblReel2 = new Label();
        lblReel2.setGraphic(iView2);
        lblReel2.setStyle("-fx-border-color: white;");

        Label lblReel3 = new Label();
        lblReel3.setGraphic(iView3);
        lblReel3.setStyle("-fx-border-color: white;");

        HBox reelBox = new HBox(); // creating a HBox to place all three label which
hold the symbols andit set to the center of border pane.
        reelBox.setPadding(new Insets(150, 40, 10, 265));
        reelBox.setSpacing(10);
        border.setCenter(reelBox);
        reelBox.getChildren().addAll(lblReel1,lblReel2,lblReel3);
        reelBox.setMinHeight(700);
        reelBox.setMinWidth(700);

        Button addCoin = new Button("Add Coin"); // creating buttons
        Button betOnce = new Button("Bet Once");
        Button betMax = new Button("Bet Max");
        Button btnReset = new Button("Bet Reset");
        Button btnStatics = new Button("Satistics");
        Button btnSpin = new Button("Spin");

        Label lblCredit = new Label("The Credit Amount : " + initialCredit + "$"); //
```

```
creating a labels to hold the credit area.
        lblCredit.setStyle("-fx-text-fill: white;");
        Label lblBet = new Label("Current Bet amount is : " + betAmount + "$"); //
creating a label to hold the bet amount.
        lblBet.setStyle("-fx-text-fill: white;");
        Label lblResult = new Label("Status"); // creating a label to hold the status
of the player (win or loss).
        lblResult.setStyle("-fx-text-fill: white;");

        HBox addLbl = new HBox(); // creating a HBox to place all three labels which
hold the cradit , bet amount and game status.
        addLbl.getChildren().addAll(lblCredit,lblResult,lblBet);
        addLbl.setPadding(new Insets(0,20,50,275));
        addLbl.setSpacing(200);

        HBox addButton = new HBox(); // creating a HBox to place all the sevan
buttons.

addButton.getChildren().addAll(addCoin,betOnce,betMax,btnSpin,btnReset,btnStatics);
        addButton.setPadding(new Insets(0,40,40,300));
        addButton.setSpacing(5);

        VBox dislpayObject = new VBox(); // creating a VBox to hold both addlbl and
addButton HBox.
        dislpayObject.getChildren().addAll(addLbl,addButton);
        dislpayObject.setPadding(new Insets(0,10,10,10));
        border.setBottom(dislpayObject);
        border.setMinHeight(500);
        border.setMinWidth(500);
        //-------------------------------------------------------------------------------
-------------------------------------------------
        addCoin.setOnAction(new EventHandler<ActionEvent>() { // button to add the
coins if the credit are low.
            @Override
            public void handle(ActionEvent event) {
                if( initialCredit > 3 || betAmount != 0) { // playe can add coin if
the credit level is lower than 3 and if the bet amount should be 0.
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("YOU MADE A MISTAKE!!!");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("You can add coins if credit level is low
than 3 or reset the bet amount and try again.");
                    alert.showAndWait();
                } else {
                    initialCredit += 1;
                    lblCredit.setText("The Credit Amount: " + initialCredit + "$");
                }
            }
        });
        betOnce.setOnAction(new EventHandler<ActionEvent>() { // button to add a
credit bet amount and less it from player's credit.
            @Override
            public void handle(ActionEvent event) {
               if(initialCredit ==0 ){
                   Alert alert = new Alert(Alert.AlertType.ERROR);
                   alert.setTitle("INSUFFICIENT CREDIT");
                   alert.setHeaderText("Flow our commands to get better experence:)");
                   alert.setContentText("Your credit level is insufficient to play,
please add coin to countinue.");
                   alert.showAndWait();
                      initialCredit = 0;
                      lblCredit.setText("The Credit Amount: " + initialCredit + "$");
```

```
                    lblBet.setText("Current Bet amount is : " + betAmount + "$");
                } else {
                lblReel1.setStyle("-fx-border-color: orange;");
                lblReel2.setStyle("-fx-border-color: orange;");
                lblReel3.setStyle("-fx-border-color: orange;");
                initialCredit -=1;
                betAmount += 1;
                lblCredit.setText("The Credit Amount: " + initialCredit + "$");
                lblBet.setText("Current Bet amount is : " + betAmount + "$");
               }
            }
        });

        betMax.setOnAction(new EventHandler<ActionEvent>() { // button to add three
credit bet amount and less it from player's credit.
            @Override
            public void handle(ActionEvent event) {
                if(initialCredit < 3){
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("INSUFFICIENT CREDIT");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("Your credit level is insufficient to play,
please add coin to countinue.");
                    alert.showAndWait();

                      lblCredit.setText("The Credit Amount: " + initialCredit + "$");
                      lblBet.setText("Current Bet amount is : " + betAmount + "$");
                } else {
                lblReel1.setStyle("-fx-border-color: orange;");
                lblReel2.setStyle("-fx-border-color: orange;");
                lblReel3.setStyle("-fx-border-color: orange;");
                initialCredit -=3;
                betAmount += 3;
                lblCredit.setText("The Credit Amount: " + initialCredit + "$");
                lblBet.setText("Current Bet amount is : " + betAmount + "$");
                }
            }
        });
        btnSpin.setOnAction(new EventHandler<ActionEvent>() { // button to roll the
reels which conatains the symbols.

            @Override
            public void handle(ActionEvent event) {
                isSpining = true;
                if (betAmount == 0){
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("YOU MADE A MISTAKE");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("You need to bet the credit first then you
can play.");
                    alert.showAndWait();
                } else {
                Thread thread3 = new Thread(new Runnable() { // creating the thread
                @Override
                public void run() {
                    while (isSpining) { // while it is true reels will roll.
                        int randomNumber = (int) Math.floor(Math.random() *
reel3.images.size()); // generates the random number according to the imahes
arraylist.
                        ImageView iv3 = new
ImageView(reel3.images.get(randomNumber).getImage());
```

```
                        reelThreeVal = reel3.images.get(randomNumber).getValue();
                        Platform.runLater(new Runnable() { // this method will help to
replace the symbols during the thread running.
                            @Override
                            public void run() {
                                lblReel3.setGraphic(iv3);
                            }

                        });
                        iv3.setFitHeight(410);
                        iv3.setFitWidth(410);
                        try {
                            Thread.currentThread().sleep(30);
                        } catch (InterruptedException ex) {

Logger.getLogger(GamePlay.class.getName()).log(Level.SEVERE, null, ex);
                        }
                    }

                }
            });

                Thread thread2 = new Thread(new Runnable() {
                @Override
                public void run() {
                    while(isSpining){
                    int randomNumber = (int) Math.floor(Math.random() *
reel2.images.size());
                            ImageView iv2 = new
ImageView(reel2.images.get(randomNumber).getImage());
                            reelTwoVal = reel2.images.get(randomNumber).getValue();
                            Platform.runLater(new Runnable() {
                                @Override
                                public void run() {
                                    lblReel2.setGraphic(iv2);
                                }
                            });
                            iv2.setFitHeight(410);
                            iv2.setFitWidth(410);
                            try {
                                Thread.currentThread().sleep(30);
                            } catch (InterruptedException ex) {

Logger.getLogger(GamePlay.class.getName()).log(Level.SEVERE, null, ex);
                            }
                    }
                }
            });
                Thread thread1 = new Thread(new Runnable() {
                @Override
                public void run() {
                    while(isSpining){
                    int randomNumber = (int) Math.floor(Math.random() *
reel1.images.size());
                            ImageView iv1 = new
ImageView(reel1.images.get(randomNumber).getImage());
                            reelOneVal = reel1.images.get(randomNumber).getValue();
                            Platform.runLater(new Runnable() {
                                @Override
                                public void run() {
                                    lblReel1.setGraphic(iv1);
                                }
```

```
                            });
                            iv1.setFitHeight(410);
                            iv1.setFitWidth(410);
                            try {
                                Thread.currentThread().sleep(30);
                            } catch (InterruptedException ex) {

Logger.getLogger(GamePlay.class.getName()).log(Level.SEVERE, null, ex);
                            }
                        }
                        }
            });
                        thread1.start(); // starting threads
                        thread2.start();
                        thread3.start();
                        noOfTry = noOfTry + 1; // keep counting of the rounds that a
perticular player plays.
                        lblReel1.setStyle("-fx-border-color: white;");
                        lblReel2.setStyle("-fx-border-color: white;");
                        lblReel3.setStyle("-fx-border-color: white;");
                        betOnce.setVisible(true);
                }
            }
        });
        btnReset.setOnAction(new EventHandler<ActionEvent>() { // button is use to
cancel the bet amount before playing.
            @Override
            public void handle(ActionEvent event) {
                if (betAmount == 0) {
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("YOU MADE A MISTAKE!!!");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("First you need to bet before reset the
credit.");
                    alert.showAndWait();
                } else {

                    if (betAmount > 0) {
                        int amount = betAmount;
                        initialCredit += amount;
                        betAmount = 0;
                    } else {
                        betAmount = 0;
                    }
                    lblReel1.setStyle("-fx-border-color: red;");
                    lblReel2.setStyle("-fx-border-color: red;");
                    lblReel3.setStyle("-fx-border-color: red;");
                    lblCredit.setText("The Credit Amount: " + initialCredit + "$");
                    lblBet.setText("Current Bet amount is : " + betAmount + "$");
                }
            }
        });
      lblReel1.setOnMouseClicked(new EventHandler<MouseEvent>() { // to stop the
rolling by click the reels
      @Override
      public void handle(MouseEvent e) {
        isSpining = false;
          if(betAmount==0){
              Alert alert = new Alert(Alert.AlertType.ERROR);
              alert.setTitle("YOU MADE A MISTAKE!!!");
              alert.setHeaderText("Flow our commands to get better experence:)");
              alert.setContentText("Please play and click the reels to stop them.");
```

```
            alert.showAndWait();
          } else {
              matchSymbol(reelOneVal, reelTwoVal, reelThreeVal, lblResult, lblCredit,
lblBet, lblReel1, lblReel2, lblReel3);
          }
      }
    });
      lblReel2.setOnMouseClicked(new EventHandler<MouseEvent>() {
      @Override
      public void handle(MouseEvent e) {
        isSpining = false;
        if(betAmount==0){
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("YOU MADE A MISTAKE!!!");
            alert.setHeaderText("Flow our commands to get better experence:)");
            alert.setContentText("Please play and click the reels to stop them.");
            alert.showAndWait();
        } else {
            matchSymbol(reelOneVal, reelTwoVal, reelThreeVal, lblResult, lblCredit,
lblBet, lblReel1, lblReel2, lblReel3);
        }
      }
     });
      lblReel3.setOnMouseClicked(new EventHandler<MouseEvent>() {
      @Override
      public void handle(MouseEvent e) {
        isSpining = false;
          if(betAmount==0){
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("YOU MADE A MISTAKE!!!");
            alert.setHeaderText("Flow our commands to get better experence:)");
            alert.setContentText("Please play and click the reels to stop them.");
            alert.showAndWait();
          } else {
              matchSymbol(reelOneVal, reelTwoVal, reelThreeVal, lblResult, lblCredit,
lblBet, lblReel1, lblReel2, lblReel3);
          }
      }
    });
      btnStatics.setOnAction(new EventHandler<ActionEvent>() { // button is to view
the player statistics in a pie chart view.
            @Override
            public void handle(ActionEvent event) {
                if(noOfTry == 0){
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("YOU MADE A MISTAKE!!!");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("First you need to play atlest a round to
cheak your statistic.");
                    alert.showAndWait();
                } else {
                    createChart();
                }
            }
      });
        //-------------------------------------------------------------------------------
-----------------------------------------------------

        Scene scene = new Scene(border, 1800, 1100); // creating the scene and adding
the borderpane which hold all the layout.
        primaryStage.setTitle("MY CASINO SLOT MACHINE - 2016323");
        primaryStage.setScene(scene);
```

```
        primaryStage.setResizable(true);
scene.getStylesheets().add(GamePlay.class.getResource("Machine.css").toExternalForm())
; // adding the external css styling sheet to the game.
        primaryStage.show();
    }
    public  void createChart(){
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("PLAYER STATISTICS");
        alert.setHeaderText("Flow our commands to get better experence:)");
        alert.setContentText("No of round played : " + noOfTry + "\nNo of rounds won :
" + noOfWin + "\nNo of rounds lost : " + onOfLoss);
        alert.showAndWait();
        PieChart pieChart = new PieChart();

        PieChart.Data slice1 = new PieChart.Data("Winning", noOfWin);
        PieChart.Data slice2 = new PieChart.Data("Lossing", onOfLoss);

        pieChart.getData().add(slice1);
        pieChart.getData().add(slice2);
        Button btnPrintD = new Button("print");
        btnPrintD.setMinWidth(80);
        btnPrintD.setMinHeight(50);
        VBox chatStatistic = new VBox(pieChart,btnPrintD);
       //
chatStatistic.getStylesheets().add(GamePlay.class.getResource("Machine.css").toExterna
lForm());
        chatStatistic.setPadding(new Insets(20,20,20,20));

        btnPrintD.setOnAction(new EventHandler<ActionEvent>() { // button to print the
player's statistics in a text file named "player_statistics".
            @Override
            public void handle(ActionEvent event) {
                if(noOfTry == 0){
                    Alert alert = new Alert(Alert.AlertType.ERROR);
                    alert.setTitle("YOU MADE A MISTAKE!!!");
                    alert.setHeaderText("Flow our commands to get better
experence:)");
                    alert.setContentText("First you need to play atlest a round to get
print your statistic printed.");
                    alert.showAndWait();
                } else {
                    PrintWriter writer = null;
                    try {
                        writer = new PrintWriter("player_statistics.txt", "UTF-8");
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }
                    writer.println("--------------------PLAYER DETAILS---------------
------");
                    writer.println("Number of time played : " + noOfTry);
                    writer.println("Number of time Won : " + noOfWin);
                    writer.println("Number of time lost : " + onOfLoss);
                    writer.println("Number of crdits Won : " + (initialCredit-10));
                    writer.println("Number of crdits Won : " + ((initialCredit-
10)/noOfTry));
                    writer.close();
                }
            }
        });
        Scene scene2 = new Scene(chatStatistic, 500, 500);
        Stage staticsStage = new Stage();
```

```
        staticsStage.setScene(scene2);
        staticsStage.setTitle("The player Staticsstics - 2016323");
        staticsStage.show();
    }
    public void matchSymbol(int x,int y,int z,Label lbl,Label lbl1,Label lbl2,Label
reel1,Label reel2,Label reel3){ // method to calculate the winning and lossing and
winning credits.
        if((reelOneVal == reelTwoVal) && (reelTwoVal == reelThreeVal)){
            noOfWin = noOfWin +1;
            initialCredit = (initialCredit + (betAmount * x))*3;
            betAmount = 0;
            lbl.setText("You Won");
            reel3.setStyle("-fx-border-color: green;");
            reel1.setStyle("-fx-border-color: green;");
            reel2.setStyle("-fx-border-color: green;");
        } else if (reelOneVal == reelTwoVal){
            noOfWin = noOfWin + 1;
            initialCredit = (initialCredit + (betAmount * x))*2;
            betAmount = 0;
            lbl.setText("You Won");
            reel2.setStyle("-fx-border-color: green;");
            reel1.setStyle("-fx-border-color: green;");
        } else if(reelTwoVal == reelThreeVal){
            noOfWin = noOfWin +1;
            initialCredit = (initialCredit + (betAmount * y))*2;
            betAmount = 0;
            lbl.setText("You Won");
            reel3.setStyle("-fx-border-color: green;");
            reel2.setStyle("-fx-border-color: green;");
        } else if(reelOneVal == reelThreeVal){
            noOfWin = noOfWin + 1;
            initialCredit = (initialCredit + (betAmount * z))*2;
            betAmount = 0;
            lbl.setText("You Won");
            reel3.setStyle("-fx-border-color: green;");
            reel1.setStyle("-fx-border-color: green;");
        } else {
            onOfLoss = onOfLoss + 1;
            betAmount = 0;
            lbl.setText("You Lost");
            reel3.setStyle("-fx-border-color: red;");
            reel1.setStyle("-fx-border-color: red;");
            reel2.setStyle("-fx-border-color: red;");
        }

        lbl2.setText("Current Bet amount is : " + betAmount + "$");
        lbl1.setText("The Credit Amount: " + initialCredit + "$");
    }
    public static void main(String[] args) {
        launch(args);
    }

}
```

- **Css file**

```
.root{
    -fx-font-size: 25pt;
    -fx-font-family: "Courier NeCourierw";
    -fx-background-color: #263959;
    -fx-spacing: 0;
    -fx-padding: 0;
    -fx-text-fill:white;

}
.iView1{
    -fx-border-color: red;
}
.lblTitle{

    -fx-font-size: 85pt;
    -fx-text-fill: red;
}

.button{
    -fx-border-color: whitesmoke;
    -fx-background-color: #263959;
    -fx-text-fill: whitesmoke;
    -fx-alignment: bottom-right;
}

.button:hover{
    -fx-background-color: white;
    -fx-text-fill: #263959;
    -fx-border-color: red;
}
```

Screen shots

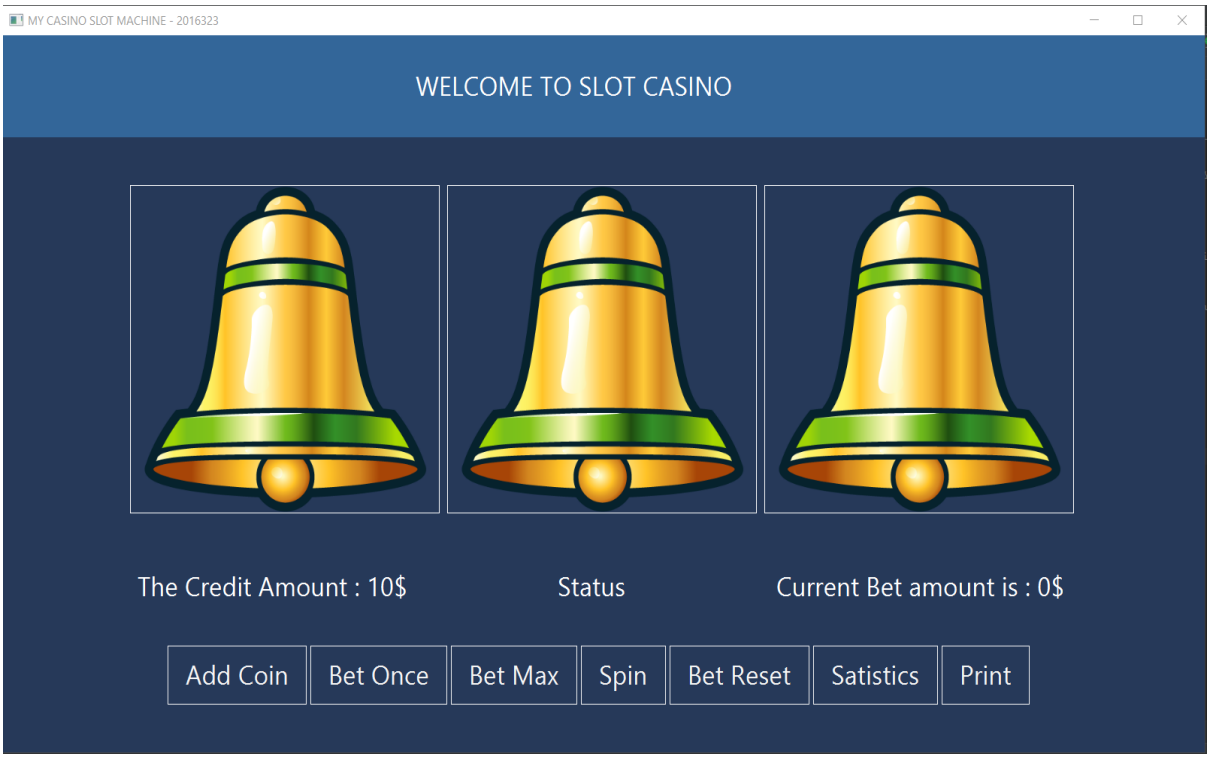
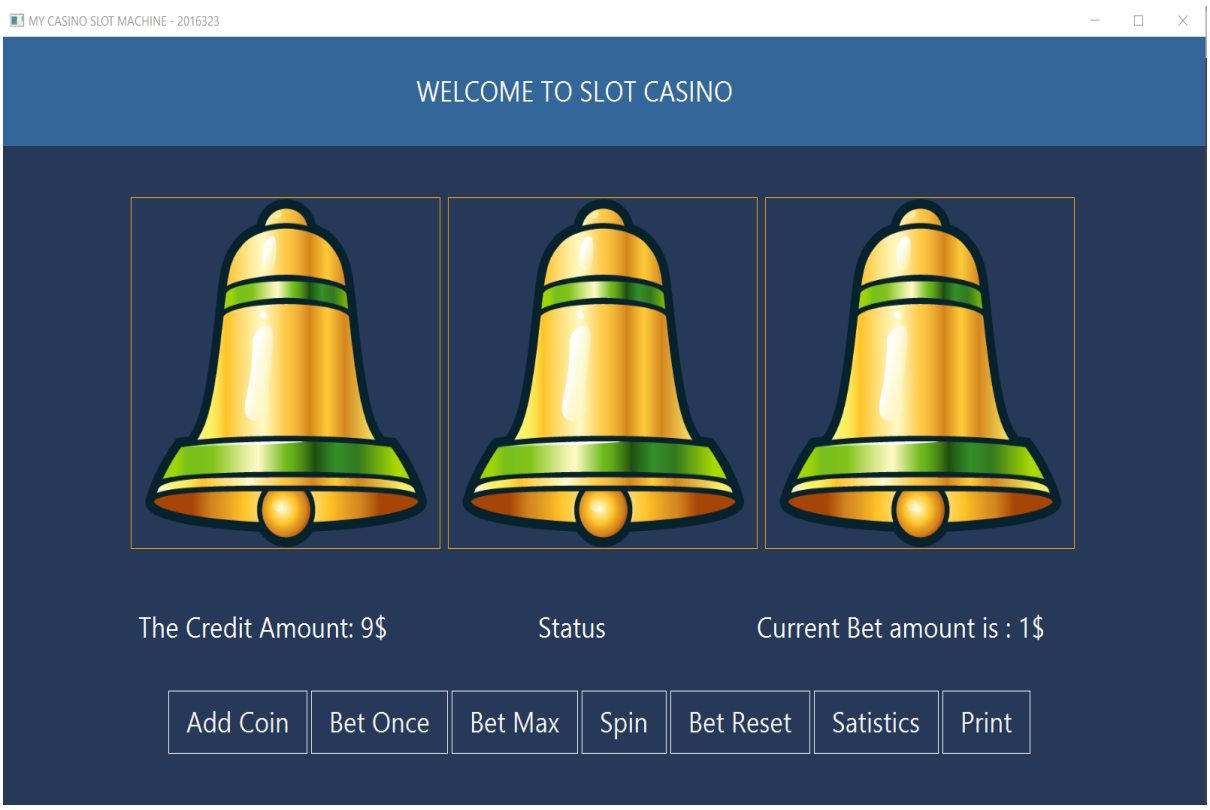


Figure 2 /Bet one credit



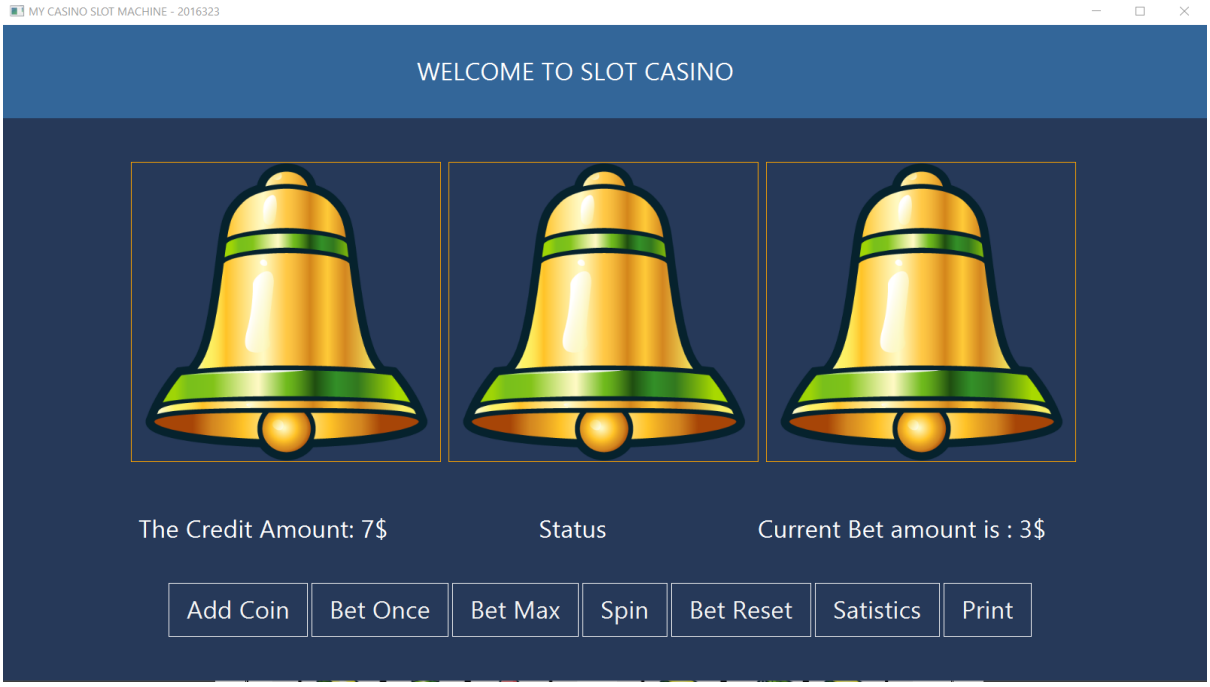


Figure 2 /Bet three credit

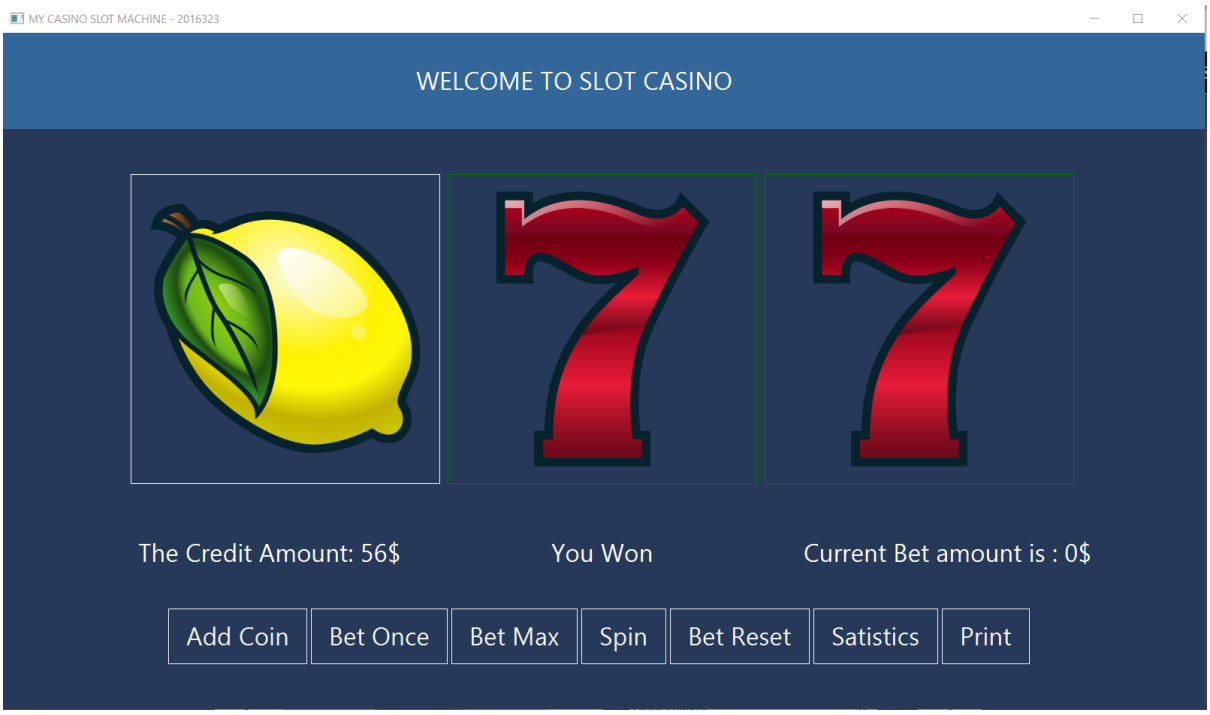


Figure 2 /Winning by matching two symbols

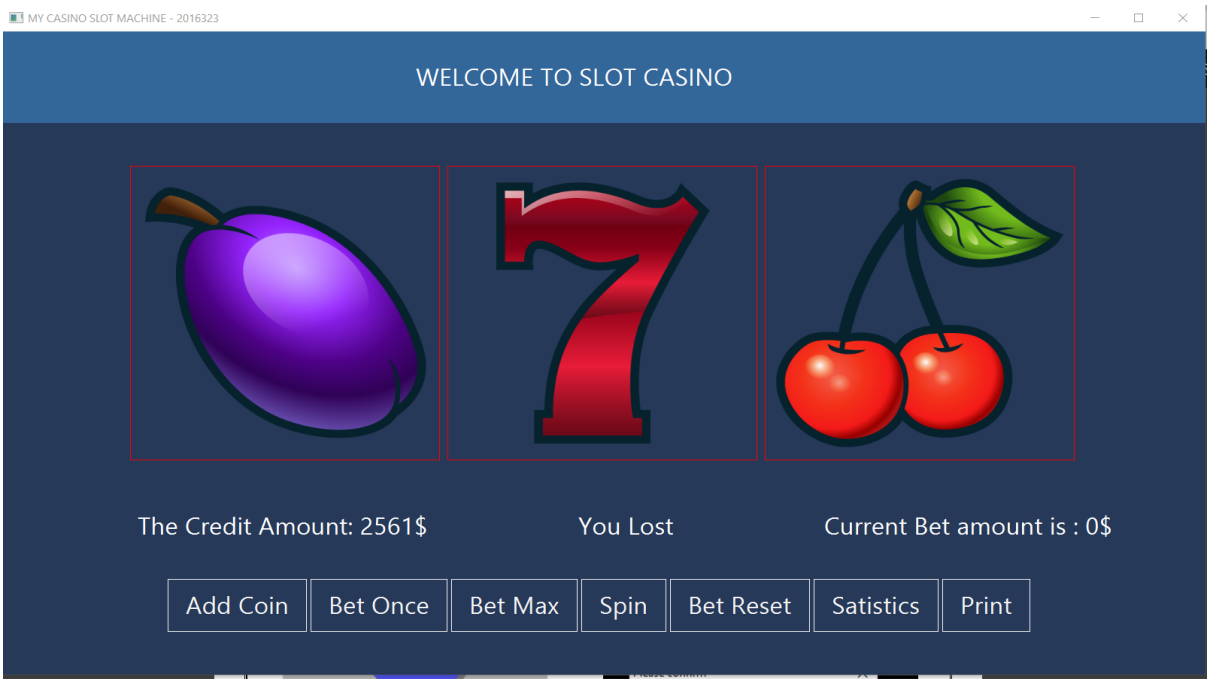


Figure 2 /Lossing by miss match of symbols

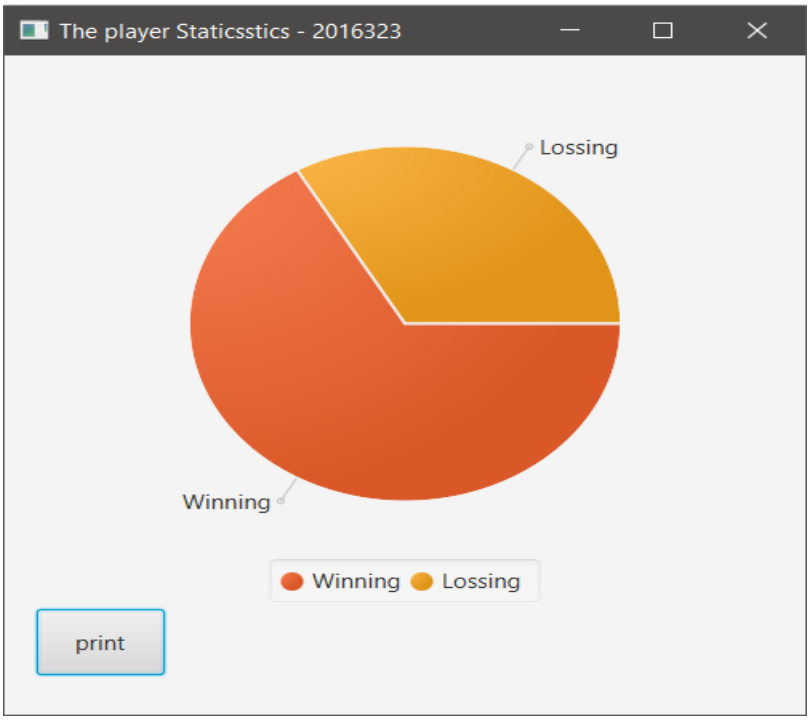


Figure 2 /Chart view before printing the statistics

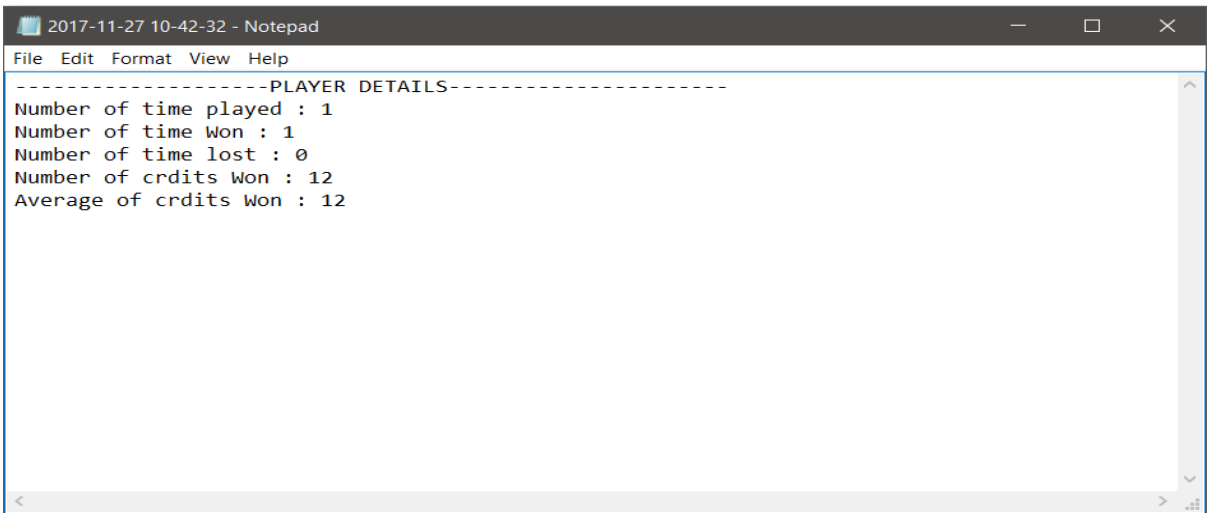


Figure 2 /Storig the player statistics in txt file

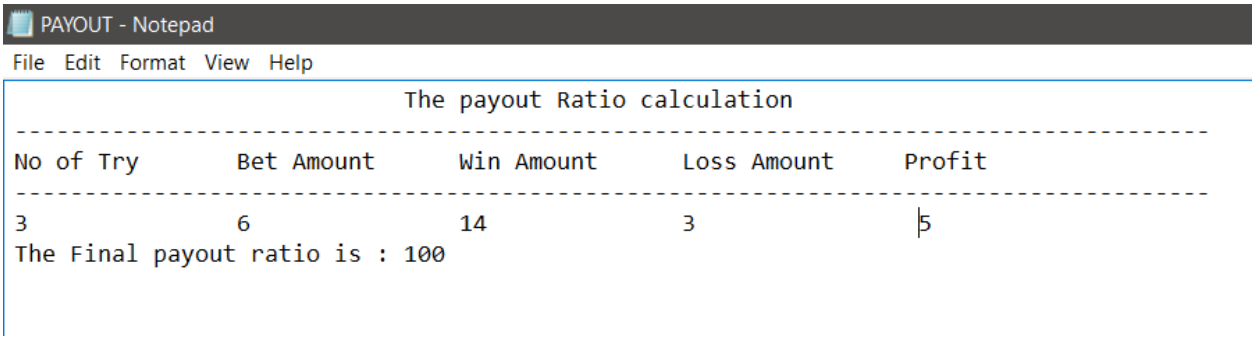


Figure 2 /Storig the payout detsils in txt file

Payout Ratio

Number of reels in the slot machine is three and each machine includes six symbols on it. Now we are calculating the payout ratio for two matching symbols.

No of reels – 03.
No of symbols – 06.
Combination of two matching symbols (six symbols for each) – 6 * 6 = 36 combinations.
So, the probability of two matching symbols in two reels – 1/36 = 0.0278

No we are calculating the payout of each symbols separately if the bet amount is 1$,

Symbol Name	Value	Multiply by the probability	payout
Seven	7	7 * 0.0278	0.1946
Watermelon	5	5 * 0.0278	0.139
Bell	6	6 * 0.0278	0.1668
Cherry	2	2 * 0.0278	0.0556
Lemon	3	3 * 0.0278	0.0834
Plum	4	4 * 0.0278	0.1112

Add all payout = 0.1946 + 0.139 + 0.1668 + 0.0556 + 0.0834 + 0.1112 = 0.7506
So, payout percentage is = 0.7506 * 100 = 75.06%

Combination of three matching symbols (six symbols for each) – 6 * 6 * 6 = 216 combinations.
So, the probability of three matching symbols in three reels - 1/216 = 0.00463

No we are calculating the payout of each symbols separately if the bet amount is 1$,

Symbol Name	Value	Multiply by the probability	Payout
Seven	7	7 * 0.00463	0.0324
Watermelon	5	5 * 0.00463	0.0232
Bell	6	6 * 0.00463	0.0278
Cherry	2	2 * 0.00463	0.0093
Lemon	3	3 * 0.00463	0.0139
Plum	4	4 * 0.00463	0.0185

Add all payout = 0.0324 + 0.0232 + 0.0278 + 0.0093 + 0.0139 + 0.0185 = 0.1251

Add both percentages - 0.7506 + 0.1251 = 0.8757
So, the percentage is = 0.8757 * 100 = 87.57%

How to make the payout percentage up to 90%,
Making the value of the symbol value to 7 now we see the new payout table,

Symbol Name	Value	Multiply by the probability	payout
Seven	7	7 * 0.0278	0.1946
Watermelon	5	5 * 0.0278	0.139
Bell	7	7 * 0.0278	0.1946
Cherry	2	2 * 0.0278	0.0556
Lemon	3	3 * 0.0278	0.0834
Plum	4	4 * 0.0278	0.1112

Add all payout = 0.1946 + 0.139 + 0. 1946 + 0.0556 + 0.0834 + 0.1112 = 0.7784

Symbol Name	Value	Multiply by the probability	Payout
Seven	7	7 * 0.00463	0.0324
Watermelon	5	5 * 0.00463	0.0232
Bell	7	7* 0.00463	0.0324
Cherry	2	2 * 0.00463	0.0093
Lemon	3	3 * 0.00463	0.0139
Plum	4	4 * 0.00463	0.0185

Add all payout = 0.0324 + 0.0232 + 0. 0324 + 0.0093 + 0.0139 + 0.0185 = 0.1297
The new payout ratio - 0.7784 + 0.1297 = 0.9081 * 100 = 90.81%

Web Application

-HTML code

```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="./gamePlay.css">
 <script src="./gamePlay.js "></script>
  <title>Slot_Machine by kajendran</title>
</head>
<body>
   <div class="modal" id="modalWindow">
    <div class="statistics">
      <div class="gamestatus">
        <div id="gamestatus__win"><h4> No of Win : </h4>
          <span id="showWin"></span>
        </div>
        <div id="gamestatus__loss"><h4> No of Loss : </h4>
          <span id="showLoss"></span>
        </div>
      </div>
      <div> <h4> Average : </h4>
        <span id="showAverage"></span>
      </div>
    </div>
  </div>
    <h1 class="pageTitle">Welcome to Slot Machine</h1>
  <main class="container">
    <div class="reel">
      <img class="reel_img" onclick="stopSpin(1)" id="img1" src="./meterial/bell.png"
alt="bell_Symbol">
      <img class="reel_img reel_img--center" onclick="stopSpin(2)" id="img2"
src="./meterial/bell.png" alt="bell_Symbol">
      <img class="reel_img" id="img3" onclick="stopSpin(3)" src="./meterial/bell.png"
alt="bell_Symbol">
    </div>
    <div class="view">
      <div><h2>
         Bet Amount :
        <span id="showBet">0</span>
      </h2>
      </div>
    </div>
      <div><h2>
     Credit :
```

```
      <span id="showBalance">10</span></h2>
    </div>
    <br>
   <div class="btnHolder">
    <button onclick="addCoin()">Add Coin</button>
    <button class="control__btn" onclick="betOne()">Bet Once</button>
    <button class="control__btn" onclick="betMax()">Bet Max</button>
    <button class="control__btn" onclick="reset()">Reset</button>
    <button class="control__btn" onclick="spin()">Spin</button>
    <button onclick="showStats()">Stats</button>
   </div>
  </main>
</body>
</html>
```

-JavaScript

```
var creditAmount= 10 // which hold the player's credit amount
var intervalTime
var isSpinning = false
var imageView1 = document.getElementById('img1')
var imageView2 = document.getElementById('img2')
var imageView3 = document.getElementById('img3')
var imgArray01
var imgArray02
var imgArray03
var spin1 = true
var spin2 = true
var spin3 = true
function displayCreditAmount () {
  var showBalance = document.getElementById('showBalanceAmount')
  showBalanceAmount.innerText = credit
}
var betAmount = 0 // which hold the player's bet amount
function displayBetAmount () {
  var showBet = document.getElementById('showBetAmount')
  showBetAmount.innerText = betAmount
}
function addBetAmount (val) { // add the credit to the bet amount
  if(credit < 1){
    window.alert(' Insufficent coins you can not play , Add coins and enjoy the game.');
  } else {
  creditAmount -= val
  betAmount += val
  displayCreditAmount() // call the display balance method
```

```
  displayBetAmount()  // call the display bet amount method
  }
}
function resetBetAmount () { // reset back the bet amount to the credit
  creditAmount = creditAmount + betAmount
  betAmount = 0
  displayCreditAmount() // call the display balance method
  displayBetAmount() // call the display bet amount method
}
function addCoin () { // increases the credit level when its low
  if(creditAmount > 9){
      window.alert(' You have enough amount to play');
  } else {
  creditAmount++
  displayCredit() // call the display balence method
  }
}
Function CreditAmount (val) { // winning credits will be added to the player's credit level
  credit += val
  displayCreditAmount() // call the display balence method
}
function betOneCredit () { // betting a credit to bet amount
  resetBetAmount() // call the reset method to cancel the bet amount
  if (credit > 0 && betAmount == 0) {
    addBetAmount(1) // call the make bet method to make bet and display it
  } else {
    window.alert(' You have already bet an amount reset it first.');
  }
}
function betMax () { // betting 3 coins to the bet amount
  resetBetAmount() // call the reset method to cancel the bet amount
  if (credit > 3 && betAmount == 0) {
    addBetAmount(3) // call the make bet method to make bet and display it
  } else {
     window.alert(' You have already bet an amount reset it first.');
  }
}

function resetCreditAmount () {
  resetBetAmount() // reset function call the reset method
}
var slotValue = [0, 0, 0]
function incrementSlotValue(position) {
  if (slotValue[position] == 5) {
```

```
    slotValue[position] = 0
  } else {
    slotValue[position]++
  }
}

var Symbols= [ // creating array and adding the images and values on it.
  {
    path: './meterial/redseven.png',
    payout: 7
  },
  {
    path: './meterial/bell.png',
    payout: 6
  },
  {
    path: './meterial/lemon.png',
    payout: 3
  },
  {
    path: './meterial/plum.png',
    payout: 4
  },
  {
    path: './meterial/cherry.png',
    payout: 2
  },
  {
    path: './meterial/watermelon.png',
    payout: 5
  }
]
function shuffleSymbols (array) { // method to suffle the array with generating random numbers.
  for (i = array.length - 1; i > 0; i -= 1) {
    j = Math.floor(Math.random() * (i + 1))
    temp = array[i]
    array[i] = array[j]
    array[j] = temp
  }
  return array.slice()
}
function spinReels () {
  if (betAmount === 0) {
    window.alert(' Bet first and play');
```

```
    return
  }
  isSpinning = true
  imgArr1 = shuffleSymbols(images)
  imgArr2 = shuffleSymbols(images)
  imgArr3 = shuffleSymbols(images)
  var turn = 1
  interval = setInterval(function() {
    if (turn % 4 === 0 && spin1) {
      incrementSlotValue(0)
      img1.src = imgArr1[slotValue[0]].path
    }
    if (turn % 5 === 0 && spin2) {
      incrementSlotValue(1)
      img2.src = imgArr2[slotValue[1]].path
    }
    if (turn % 3 === 0 && spin3) {
      incrementSlotValue(2)
      img3.src = imgArr3[slotValue[2]].path
    }
    turn++
    console.log(turn)
  }, 0)
}
var lostCount = 0 // hold the lossing count
function incrementLostCount () {
  lostCount++
  displayLostCount() // call the method to display the count
}
function displayLostCount () {
  var showLoss = document.getElementById('showLoss')
  showLoss.innerText = lostCount
}

function stopSpin (val) {
  if (! isSpinning) {
    return
  } else if (payoutArr[2] == payoutArr[0]) {
    win = payoutArr[2] * betAmount
  } else {
    win = 0 - betAmount
  }
  spin1 = true
  spin2 = true
```

```
  spin3 = true
  averageWin = ((averageWin * (winCount + lostCount)) + win)/(winCount + lostCount + 1) //
calculating the average winning credits
  if (win > 0) {
    incrementWinCount()
    winCredit(win)
    window.alert(' You have Won ' + win + ' Coins')
  }
  else {
    incrementLostCount()
    window.alert('You lost')
  }
  betAmount = 0
  displayBetAmount()
}
var modal = document.getElementById('modalWindow')
function showStats() { // display the statistics using the winning and lossing counts
  modal.style.display = "block"
  // calling the sub methods
  displayWinCount()
  displayLostCount()
  displayAverage()
  var winFlex = winCount
  var loseFlex = lostCount
  document.getElementById('winlose__win').setAttribute("style",`flex:${winFlex}`)
  document.getElementById('winlose__lose').setAttribute("style",`flex:${loseFlex}`)
}
```

- **CSS**

```
@import
url('https://fonts.googleapis.com/css?family=Fira+Sans:100,200,300,400,500,600,700,800,900|
Roboto:100,300,400,500,700,900');
 /* body__layout */
body {
  box-sizing: border-box;
  background-color: #114fb2;
```

```
  color:white;
  text-align:center;
  font-family: 'Roboto', sans-serif;
}
.pageTitle {
  background-color: #052454;
  border-radius: 0.4rem;
  margin: 0.1rem;
  text-align: center;
  padding: 1rem;
}
.container {
  max-width: 1005px;
  margin: 0 auto;
}
.reel {
  border: 2px solid #141414;
  border-radius: 0.2rem;
  max-width: 1200px;
  width: 90%;
  margin: 1rem auto;
  display: flex;
  justify-content: space-between;
}
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  padding-top: 100px; /* Location of the box */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: rgb(0,0,0); /* Fallback color */
  background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
}

/* button__elements */
button {
  padding: 0.3rem 1rem;
  border: 2px solid #141414;
  width: 120px;
  height: 50px;
```

```
  font-size: 18px;
  border-radius: 0.4rem;
}
button:hover {
  width: 140px;
  height: 70px;
  border: 2px solid white;
}
.pure-button {
  padding: 0.2rem 1rem;
}
.btnHolder {
  display: flex;
  justify-content: space-around;
}

/* Details__holder */
.info {
  display: flex;
  justify-content: space-around;
  margin: 1rem;
}
.view {
  display: flex;
  justify-content: center;
  margin: 1rem;
}

/* slot__imageViewer */
.reel_img {
  flex: 1;
  max-width: 300px;
  max-height: 300px;
}
.reel_img--center {
  border-left: 2px solid #141414;
  border-right: 2px solid #141414;
}
/* Display_game_status */
.gamestatus {
  display: flex;
  margin-bottom: 1rem;
}
```

ScreenShots