JAVA PROJECT

DICE SPLIT

BY

SESAN POPOOLA & ABDUL AZIZ

16TH JULY 2024

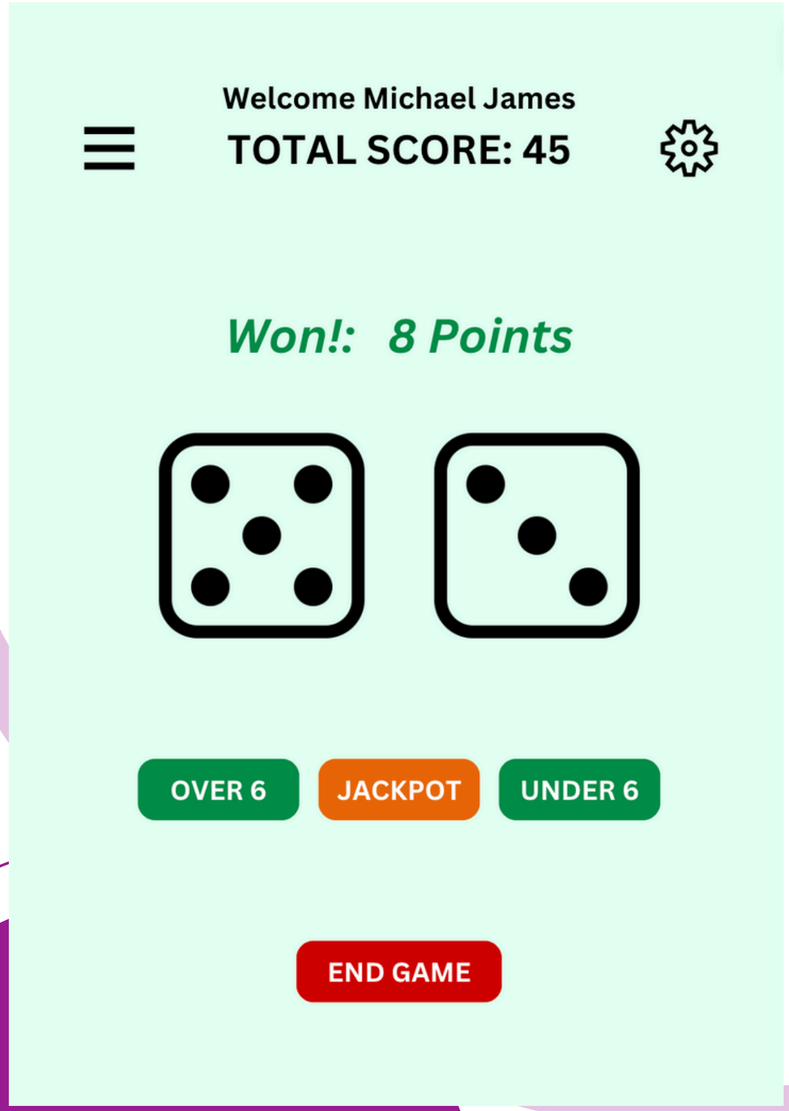
Project Concept & Idea

This project was an original idea, we never sorted out game dynamic ideas from any previously built game. We made a plan, put what we wanted together and started building.

Most of the decisions we made came as we made progress with the codes. On the right, you will find what our original outlook for the game was.

As we make progress with the presentation, you will see what we ended up with, comparing our finished product to the initial idea we had.

The application ended up looking way better than what we imagined.



Inputs & Obstacles

On the right you will find a few stats. It took around 100+ combined hours of coding to ge the job done. We also had to iterated through color schemes and desing ideas more than 20 times and we ended up writing around 1,500 lines of codes to make the application look as presented today.

While it was a combined team effort, Abdul spent more time working on the home page and credit page while i worked on the Main game page and settings page.

Gamelogic, animations and sound effects was a complete team work as we both had our part to play.

The way we split the work between ourselves made it possible for a smooth work between myself and Abdul.

Every code idea used to build this project was taught by the professor, we never really needed any outside help, and there was really no obstacle that a little brainstorming couldn’t fix

100+
Hours

20+
Design Iterations

1,500+
Lines Of Codes

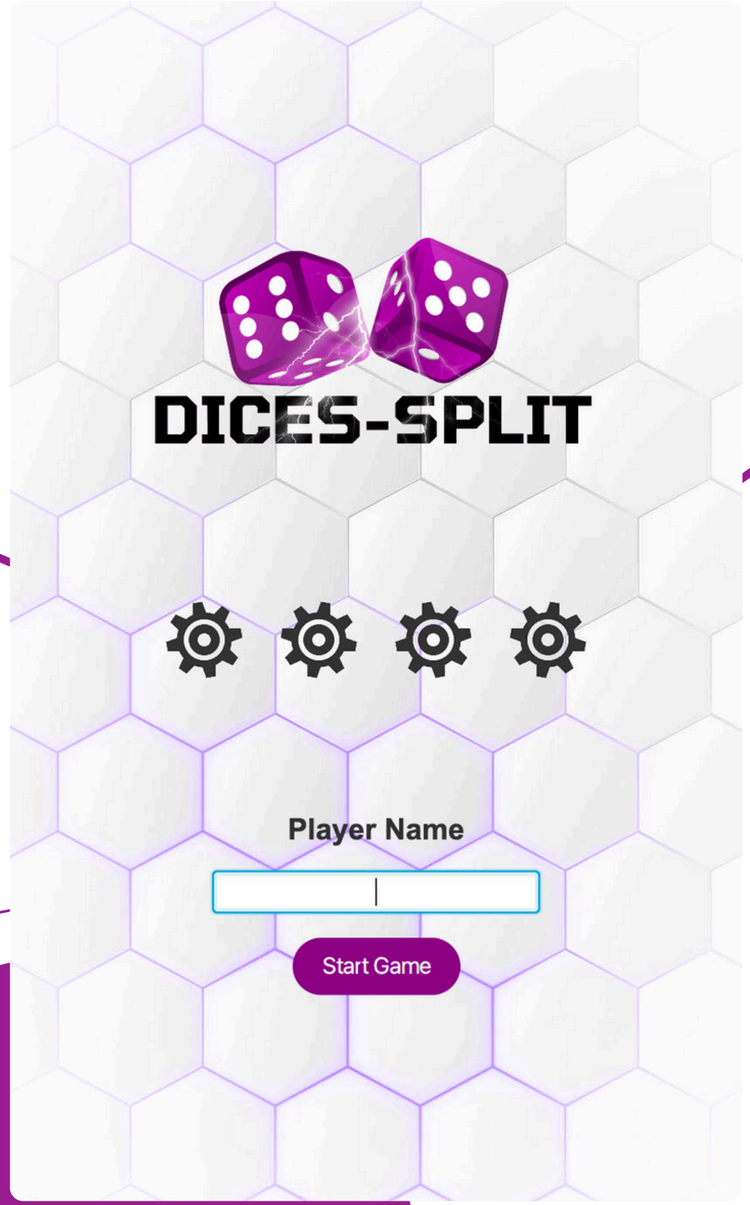
INTRODUCTION

HOME PAGE

The image on the right shows the launch page (Home Page). A player can decide to enter their name to start the game or not. This page is designed to be as simple as possible to get rid of any distractions so players can get straight into the action.

The page extends a stack pane so, a border pane with all the necessary nodes is stacked on top the page with a transparent background so the background design is visible.

```java
public class HomePage extends StackPane {

    // Declaration of a static Theme variable
    7 usages
    public static Theme theme;

    // Constructor for the HomePage class
    1 usage
    HomePage(Stage primaryStage) {

        // Create a BorderPane for layout management
        BorderPane pane = new BorderPane();

        // Set the background of the pane to transparent
        pane.setStyle("-fx-background-color: transparent;");
```



1

2

3

4

5

6

7

8

9

Main Game

Just like the home page, the game page extends a stack pane and a border pane containing all the nodes stacked on top.

There are 3 main buttons to play the game. Over, Under and Jackpot, Here's a code snippet for a quick overview

```
public GamePage(Stage primaryStage, Scene scene1, Scene scene2) {

    // Initialize sound effects status text
    sfx = new Text( s: "On");

    // Load dice roll sound effect
    // Sound from over and out youtube channel - https://youtu.be/MK0l_TQH6aA?si=ggHjeeZ4vMteU5ED
    Media diceSFX = new Media(new File( pathname: "Music/sound2.MP3").toURI().toString());
    diceSFXPlayer = new MediaPlayer(diceSFX);

    // Load game over sound effect
    // Sound Name is Tuba Sting from uppbeat.io - https://uppbeat.io/sfx/tuba-sting-wrong-answer/3111/17041
    Media gameOverSfx = new Media(new File( pathname: "Music/sound6.MP3").toURI().toString());
    MediaPlayer gameOverSfxPlayer = new MediaPlayer(gameOverSfx);

    // Create a BorderPane for layout management
    BorderPane pane = new BorderPane();
    pane.setStyle("-fx-background-color: transparent;");
    this.setStyle("-fx-background-color: transparent;");

    // HBox for play buttons with spacing
    HBox playButtons = new HBox( v: 50);

    // VBox for game view with spacing and padding
    VBox gameView = new VBox( v: 30);
    gameView.setPadding(new Insets( v: 0,  v1: 0,  v2: 70,  v3: 0));
```

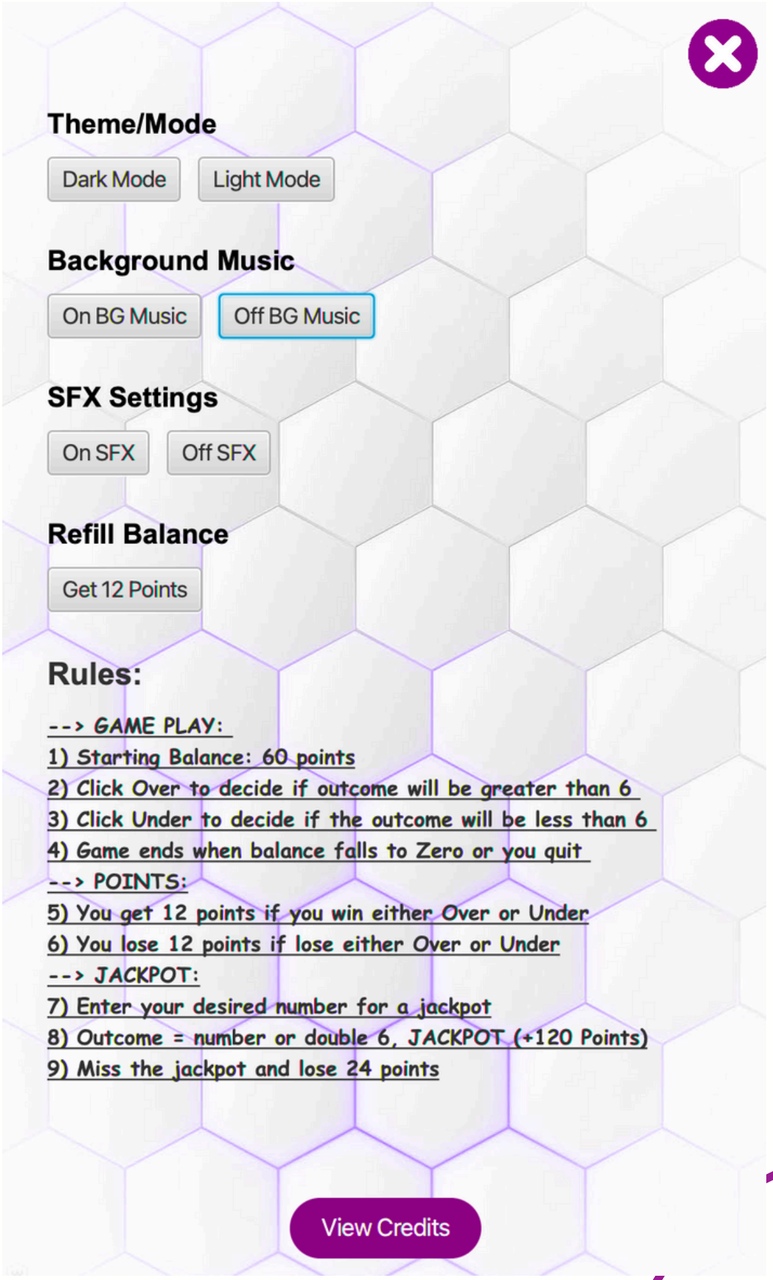
Settings Page

1
2
3
4
5
6
7
8
9

On the right, you will find the settings page where most of the game controls can be made, You can turn of and on background music and sound effects. Change from light to dark mode,

refill your balance and also find a list of the game rules to help the player get acquainted with the game.

```
public Settings(Stage primaryStage) {

    // Set the background color of the settings screen to transparent
    this.setStyle("-fx-background-color: transparent;");

    Theme theme = new Theme(); // Create a new Theme object
    Animations animate = new Animations(); // Create a new Animations object

    BorderPane pane = new BorderPane();
    pane.setStyle("-fx-background-color: transparent;");

    // Text for displaying successful operations
    Text successful = new Text( s: "");
    successful.setFill(Color.GREEN);
    successful.setFont(Font.font( s: "Times New Roman", FontWeight.BOLD, v: 16));

    // Theme Mode Settings
    Text mode = new Text( s: "Theme/Mode");
    mode.setFont(Font.font( s: "Arial", FontWeight.BOLD, v: 16));

    HBox themeButtons = new HBox( v: 10); // Horizontal box for theme buttons
    VBox themeBox = new VBox( v: 10); // Vertical box for theme settings
```



Credits Page

1
2
3
4
5
6
7
8
9

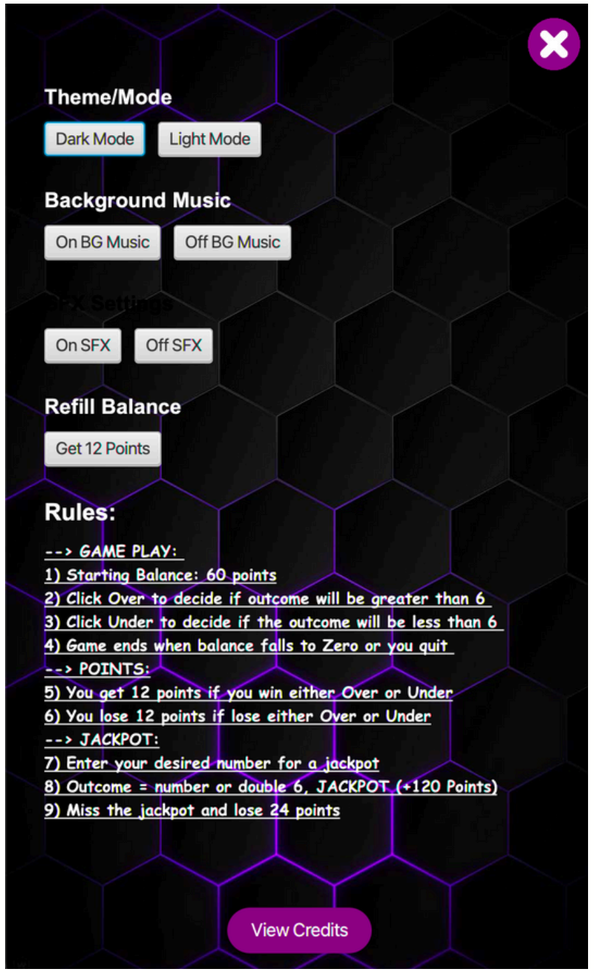
On the right, you's find the credit page where all course and credits have been attributed accordingly.

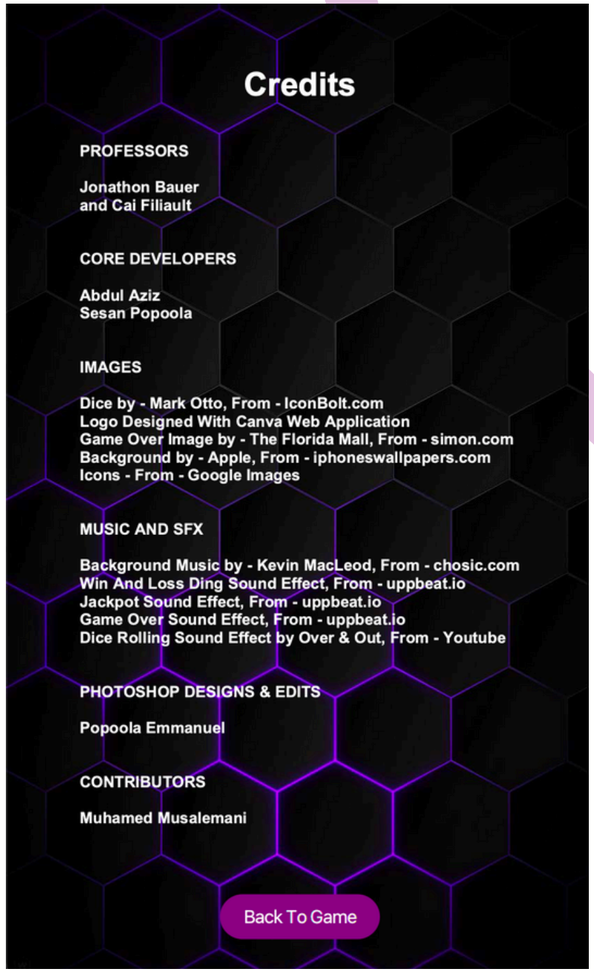
You can find the code snippet below;

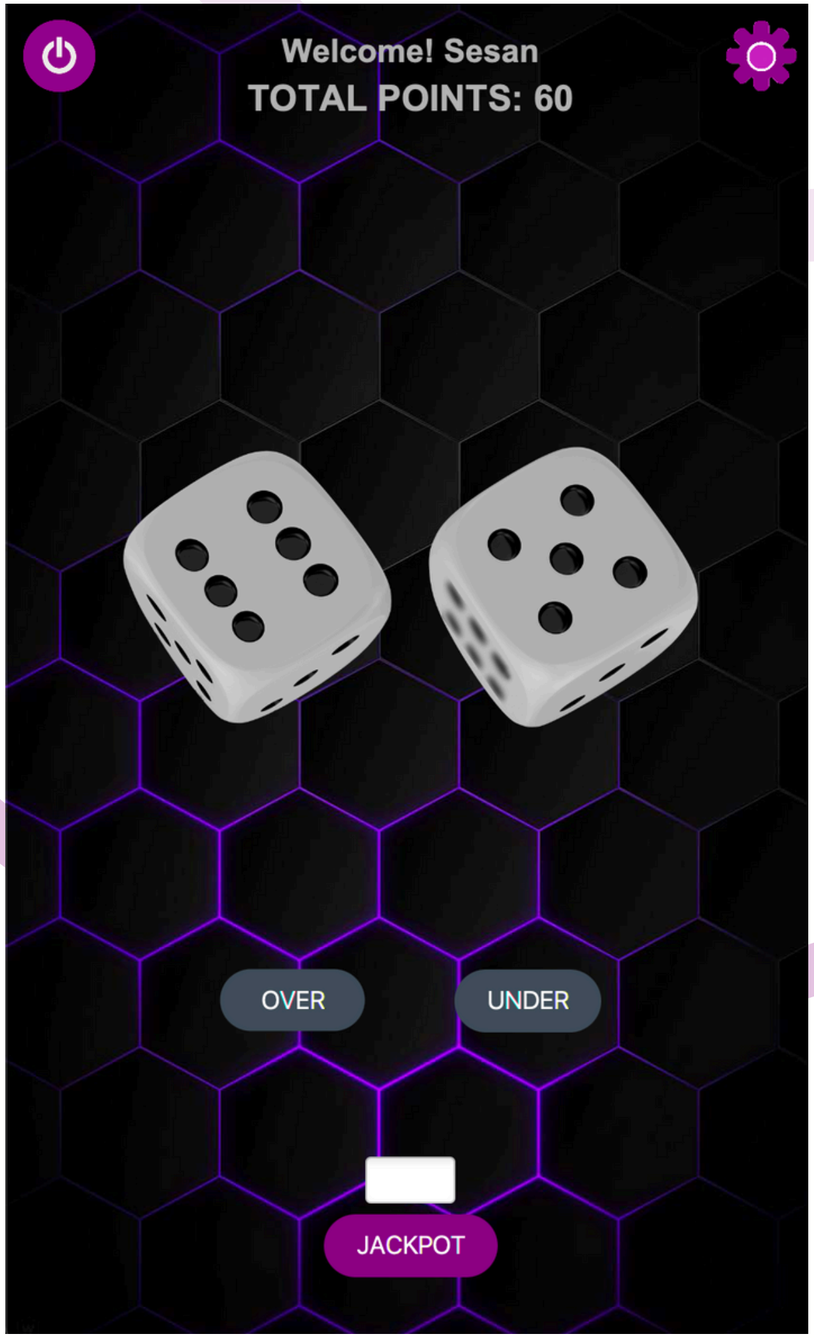
```
public class CreditPage extends StackPane {
    // Declaration of static variables for theme, credits, and credit contents text
    7 usages
    public static Theme theme;
    5 usages
    public static Text credits;
    5 usages
    public static Text creditContents;

    // Constructor for the CreditPage class
    1 usage
    CreditPage(Stage primaryStage) {

        // Initialize the theme
        theme = new Theme();

        // Create a BorderPane for layout management
        BorderPane pane = new BorderPane();

        // Create and configure the credits title text
        credits = new Text( s: "Credits");
        credits.setFont(Font.font( s: "Arial", FontWeight.BOLD,  v: 25));

        // Create and configure the text for credit contents
        creditContents = new Text( s: "PROFESSORS \n\n" +
                "Jonathon Bauer\n" +
                "and Cai Filiault\n\n\n" +
                "CORE DEVELOPERS\n\n" +
```

Dark Mode

Here are some images of the Dark theme mode







1
2
3
4
5
6
7
8
9

Game Logic

1
2
3
4
5
6
7
8
9

There's a subclass called GameLogic, as the name implies, this is where all the codes related to gameplay are placed, there's over 400 lines of codes and they're all methods for different actions.

On the right, you will find a method to roll dice whenever a player decides to play.

Below is a method to handle the logic behind the Jackpot which is invoked whenever the Jackpot button is clicked. It also perfectly handles exceptions.

```
// Method for rolling dice when JACKPOT button is clicked
1 usage
public void rollDiceJackpot(Text outcomeText, Text totalPoints, TextField jackpotField, Text hintText, Text winOrLose) {

    dingFailPlayer.stop();
    jackpotSfxPlayer.stop();
    diceRolling = true;

    try {
        int userJackpot = Integer.parseInt(jackpotField.getText());

        int jackPot = 12;

        if (userJackpot > 1 && userJackpot < 13) {

            rollDice();

            if (GamePage.sfx.getText() == "On") {
                GamePage.diceSFXPlayer.play();
            }
```

```
// Method to roll dice
4 usages
public void rollDice() {

    setSpacing(10);

    Random random = new Random();

    rollOne = random.nextInt( origin: 1,  bound: 7);
    rollTwo = random.nextInt( origin: 1,  bound: 7);

    // Load dice images based on the theme and dice roll
    Image diceOne = new Image(getClass().getResourceAsStream( name: getTheme() + rollOne + ".png"));
    ImageView firstDice = new ImageView(diceOne);

    Image diceTwo = new Image(getClass().getResourceAsStream( name: getTheme() + rollTwo + ".png"));
    ImageView secondDice = new ImageView(diceTwo);

    // Set dice image dimensions
    firstDice.setFitWidth(160);
    firstDice.setFitHeight(160);

    secondDice.setFitWidth(160);
    secondDice.setFitHeight(160);

    // Initialize rotate transitions for dice
    rotateTransition1 = new RotateTransition(Duration.millis( v: 1800), firstDice);
    rotateTransition1.setByAngle(1440);
    rotateTransition1.setCycleCount(1);
    rotateTransition1.play();

    rotateTransition2 = new RotateTransition(Duration.millis( v: 2200), secondDice);
    rotateTransition2.setByAngle(1440);
    rotateTransition2.setCycleCount(1);
    rotateTransition2.play();

    // Arrays to store dice images for animation
    Image[] diceImages1 = new Image[6];
    for (int i = 0; i < 6; i++) {
        diceImages1[i] = new Image(getClass().getResourceAsStream( name: getTheme() + (i + 1) + ".png"));
    }
```

File I/O

1
2
3
4
5
6
7
8
9

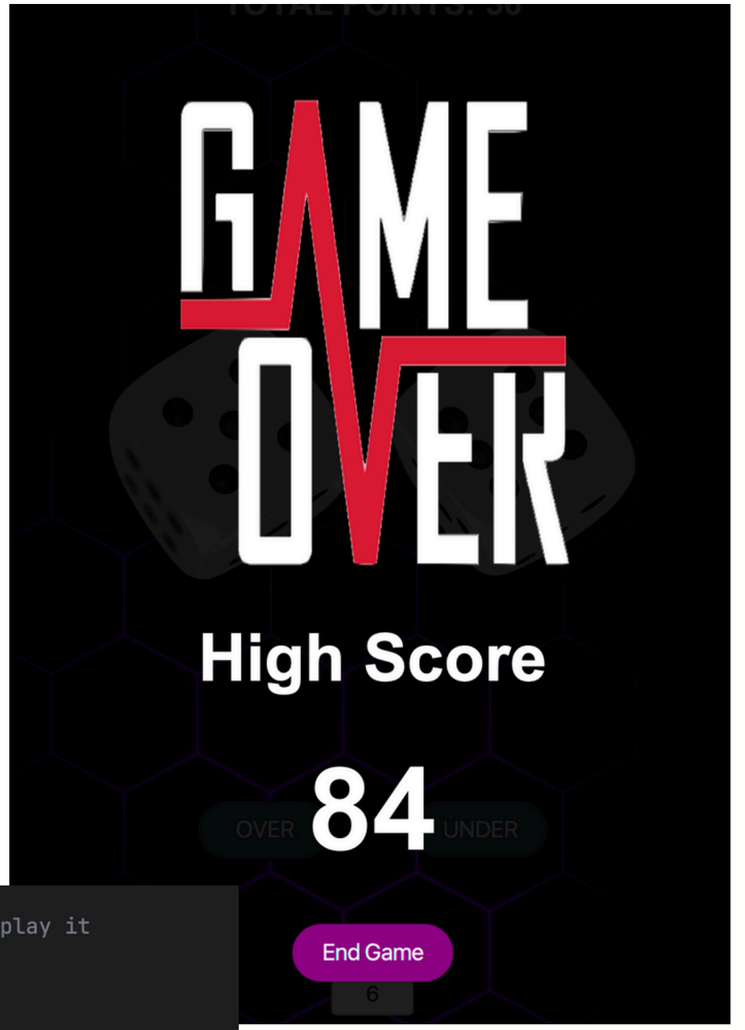
File I/O was used to handle the high score logic when the game ends. During the game, an array list is used to store users scored points. As soon as the game ends, it produces the highest score the user had before the game ends and saves it to the high score text file.

This high score file is then read and the highest score is printed on the game over page as you can see on the right.

Below are the code snippet to save the highest score to file in the Game Logic class and to read highest score from file in the Game Over Class

```
// Method to save the highest score to a file
4 usages
public void saveHighestScore() {

    if (getBalancePoints() < 1 || gameEnded) {
        int highestScore = Collections.max(highScores);

        try {
            PrintWriter output = new PrintWriter(file);

            output.print(highestScore);

            output.close();

        } catch (Exception a) {
            System.out.println("An Error Occurred");
        }
    }
}
```

```
// Try to read the high score from the file and display it
try {
    Scanner input = new Scanner(file);

    while (input.hasNext()) {
        String highestScore = input.next();
        highScoreNumber.setText(highestScore);
    }

} catch (Exception a) {
    System.out.println("An Error Occurred");
}
```



Animations

1 2 3 4 5 6 7 8 9

Animations were used to add some graphical appeal to the game and also make the game play enjoyable. We used them for the spinning dice, when points where being recorded and on the home page.

Most codes related to animations can be found in the Animation class. You can see the code snippet on the right and at the bottom of this page.

```
3 usages
public void successfulSettings(Text text){

    FadeTransition fadeTransition = new FadeTransition(Duration.millis( v: 2000), text);
    fadeTransition.setFromValue(0);
    fadeTransition.setToValue(1);
    fadeTransition.setCycleCount(2);
    fadeTransition.setAutoReverse(true);
    fadeTransition.play();
}
```

```
public class Animations {

    6 usages
    public void outcomeText(Text text){

        TranslateTransition translateTransition = new TranslateTransition(Duration.millis( v: 1200), text);
        translateTransition.setByY(-80);
        translateTransition.play();

        FadeTransition fadeTransition = new FadeTransition(Duration.millis( v: 1200), text);
        fadeTransition.setFromValue(1);
        fadeTransition.setToValue(0);
        fadeTransition.setCycleCount(1);

        TranslateTransition translateTransition1 = new TranslateTransition(Duration.millis( v: 10), text);
        translateTransition1.setByY(80);

        ParallelTransition parallelTransition = new ParallelTransition();
        parallelTransition.getChildren().addAll(translateTransition, fadeTransition);

        SequentialTransition sequentialTransition = new SequentialTransition();
        sequentialTransition.getChildren().addAll(parallelTransition, translateTransition1);
        sequentialTransition.setCycleCount(1);
        sequentialTransition.play();

    }

    6 usages
    public void winOrLoss(Text text){

        FadeTransition fadeTransition = new FadeTransition(Duration.millis( v: 200), text);
        fadeTransition.setFromValue(0);
        fadeTransition.setToValue(1);
        fadeTransition.setCycleCount(6);
        fadeTransition.setAutoReverse(true);
        fadeTransition.play();

    }
```

Sound Effect & Background Music

1 2 3 4 5 6 7 8 9

Sound effects are an essential part of an enjoyable game, so we added a few. There's a total of 5 sound effects and a background music.

1) There's a sound effect for the rolling dice
2) For a positive point addition when you win
3) For a negative point subtraction when you loose
4) For successfully winning a Jackpot
5) For when the game ends
6) A general background music

Background music and sound effects can be turned off and on in the settings page.

```
// Sound Name is bell congratulations from uppbeat.io - https://uppbeat.io/sfx/bell-congratulations/798
// Media for positive sound effect
1 usage
public static Media ding = new Media(new File( pathname: "Music/sound3.MP3").toURI().toString());
4 usages
public static MediaPlayer dingPlayer = new MediaPlayer(ding);

// Sound Name is negative glass from uppbeat.io - https://uppbeat.io/sfx/notification-alert-negative-gl
// Media for negative sound effect
1 usage
public static Media dingFail = new Media(new File( pathname: "Music/sound4.MP3").toURI().toString());
6 usages
public static MediaPlayer dingFailPlayer = new MediaPlayer(dingFail);

// Sound Name is Winner Trumpet from uppbeat.io - https://uppbeat.io/sfx/winner-trumpet-fanfare/11352/2
// Media for jackpot sound effect
1 usage
public static Media jackpotSfx = new Media(new File( pathname: "Music/sound5.MP3").toURI().toString());
2 usages
public static MediaPlayer jackpotSfxPlayer = new MediaPlayer(jackpotSfx);
```

DICES-SPLIT

Thank You