Rock-Paper-Scissors Game Documentation

Introduction

The Rock, Paper, Scissors Game is a Java application that brings the classic hand game to life in a digital format. Unlike traditional versions, this implementation offers a single-player experience where the user competes against the computer's choices. The objective is to make strategic decisions and predict the computer's moves to achieve a winning outcome. This documentation provides an overview of the project structure, key functionalities, and technical aspects, emphasizing the engaging experience of playing Rock, Paper, Scissors against an AI opponent.

Abstraction

'La' class

Encapsulates the logic of the game

- Attributes
  - "private int human"- represents the user's choice
  - "private int computer"- represents the computer's randomly generated choice

    Where 0=Rock

    1=Paper

    2=Scissors

  - "int result" -where 0=Loss

    1=Tie

    2=Win

  - "private Random random"- Randomises the choice of the computer

- Methods
  - "public int play()" Generates a random computer choice and returns it
  - "public int getComputer()"-Returns the computer's choice
  - "public void setHuman(int human)"- Sets humans choice
  - "public int checkWinner()"- Compares the human and computer choices to determine the game results as an integer with the aforementioned rule

'Pla' class

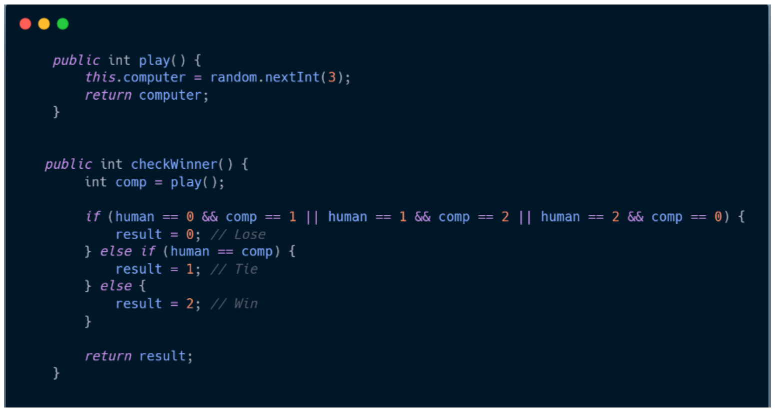
Is responsible for the GUI and user interaction

- Attributes
  - "private La rockGame"- Encapsulates the game logic

- “private ImageView viewImage”- Represents GUI element for displaying images
- Methods
- “public void start(Stage primaryStage)”-Sets up the GUI
- “private Button createButton(int choice)” – Abstracts the creation of buttons with images for the different choices
- “private void playGame(int humanChoice)”- Takes as input the choice of the human, abstracts the game logic when a button is clicked
  - Functionalities
- Game logic – The game follows the rules of the game Rock, Paper, Scissors
- The first class encapsulates the game logic, generates random computer choices and determines the winner based on the choices
- User Interface – The game provides a graphical user interface using JavaFX
- Components are organized in HBox-es and VBox-es or GridPane
- An ‘ImageView’ is used to display the computer's choice
- Button Clicking- Represent the user's choice
- Result Display- The game displays the results to the console
- Random computer choice
- Abstraction and Encapsulation- The game uses abstraction to hide implemented details

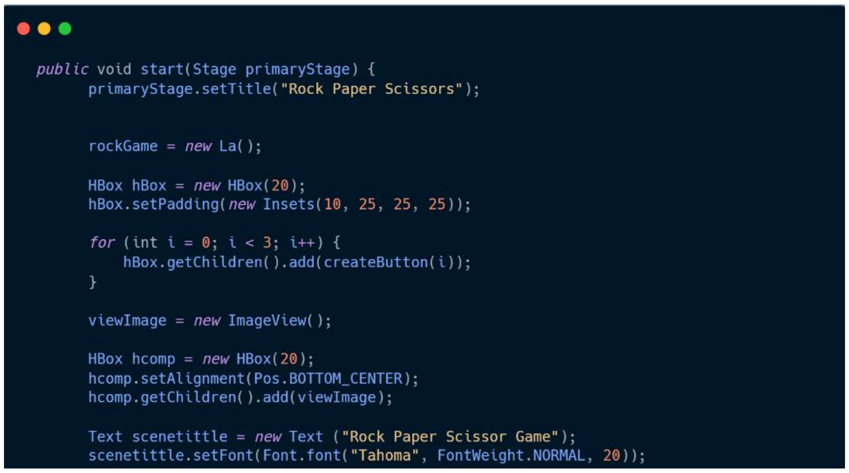
Methodology

- Game Logix



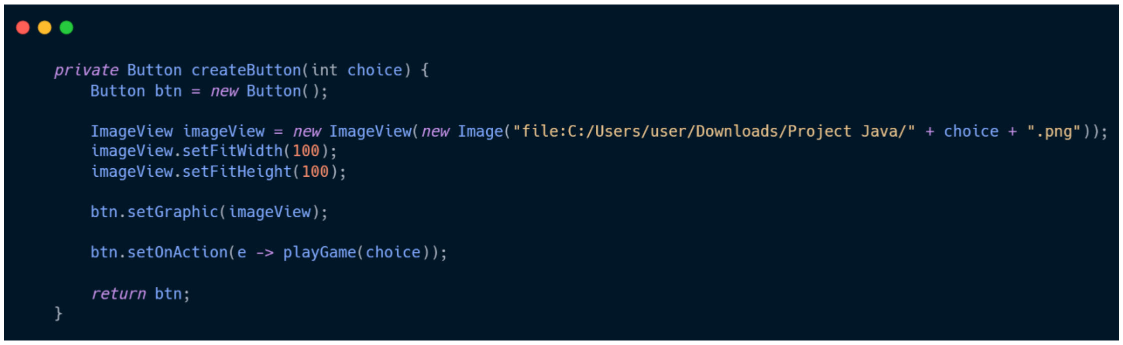
```
    public int play() {
        this.computer = random.nextInt(3);
        return computer;
    }


   public int checkWinner() {
        int comp = play();

        if (human == 0 && comp == 1 || human == 1 && comp == 2 || human == 2 && comp == 0) {
            result = 0; // Lose
        } else if (human == comp) {
            result = 1; // Tie
        } else {
            result = 2; // Win
        }

        return result;
    }
```

- Scene components



```
  public void start(Stage primaryStage) {
        primaryStage.setTitle("Rock Paper Scissors");


        rockGame = new La();

        HBox hBox = new HBox(20);
        hBox.setPadding(new Insets(10, 25, 25, 25));

        for (int i = 0; i < 3; i++) {
            hBox.getChildren().add(createButton(i));
        }

        viewImage = new ImageView();

        HBox hcomp = new HBox(20);
        hcomp.setAlignment(Pos.BOTTOM_CENTER);
        hcomp.getChildren().add(viewImage);

        Text scenetittle = new Text ("Rock Paper Scissor Game");
        scenetittle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
```

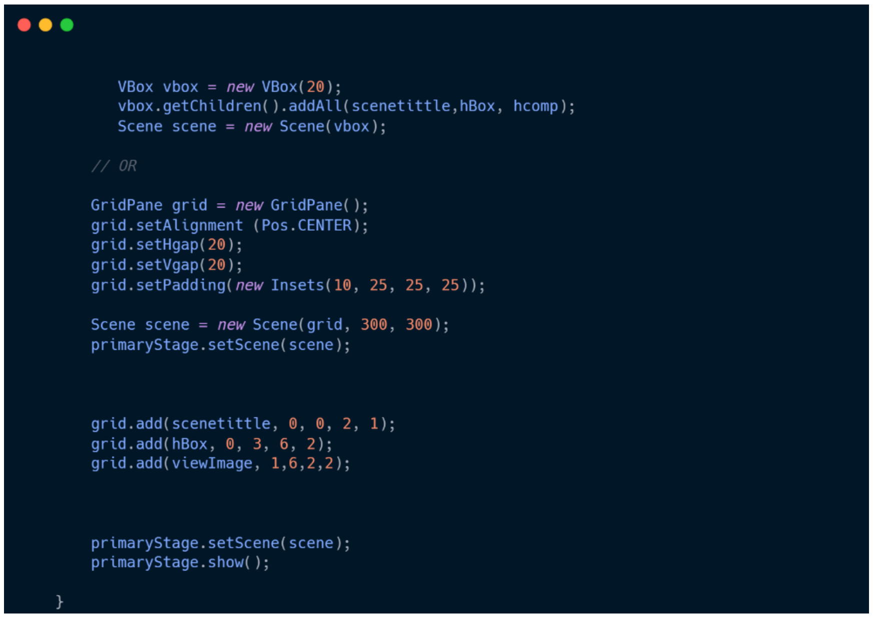
- The Creation of buttons with images from the folder



```
    private Button createButton(int choice) {
        Button btn = new Button();

        ImageView imageView = new ImageView(new Image("file:C:/Users/user/Downloads/Project Java/" + choice + ".png"));
        imageView.setFitWidth(100);
        imageView.setFitHeight(100);

        btn.setGraphic(imageView);

        btn.setOnAction(e -> playGame(choice));

        return btn;
    }
```

- Scene Layout

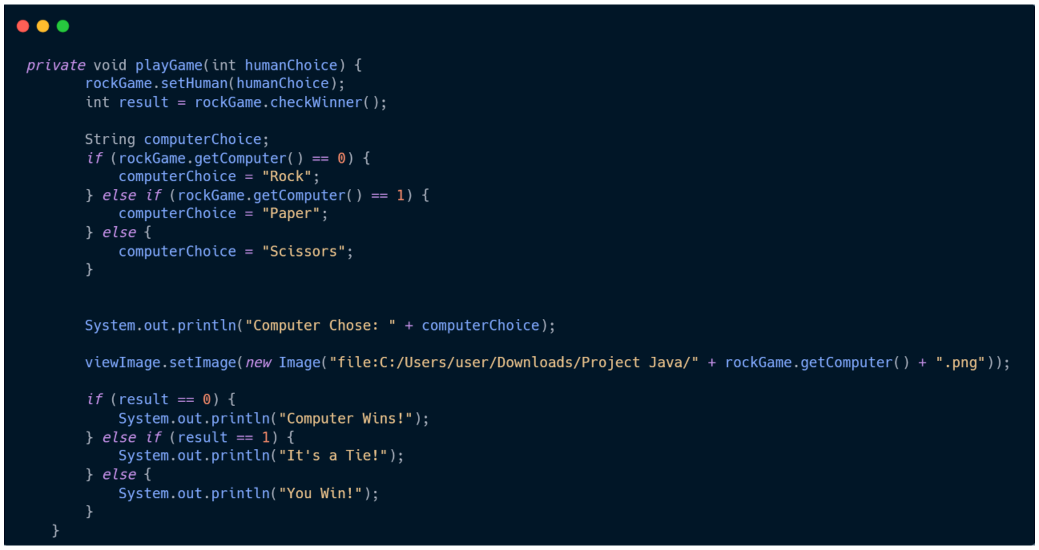
(could be a grid or a VBox)



```
            VBox vbox = new VBox(20);
            vbox.getChildren().addAll(scenetittle,hBox, hcomp);
            Scene scene = new Scene(vbox);

        // OR

        GridPane grid = new GridPane();
        grid.setAlignment (Pos.CENTER);
        grid.setHgap(20);
        grid.setVgap(20);
        grid.setPadding(new Insets(10, 25, 25, 25));

        Scene scene = new Scene(grid, 300, 300);
        primaryStage.setScene(scene);


        grid.add(scenetittle, 0, 0, 2, 1);
        grid.add(hBox, 0, 3, 6, 2);
        grid.add(viewImage, 1,6,2,2);


        primaryStage.setScene(scene);
        primaryStage.show();

    }
```

- Main block of code

Responsible and calls methods for handling the logic when a button is clicked



```
private void playGame(int humanChoice) {
        rockGame.setHuman(humanChoice);
        int result = rockGame.checkWinner();

        String computerChoice;
        if (rockGame.getComputer() == 0) {
            computerChoice = "Rock";
        } else if (rockGame.getComputer() == 1) {
            computerChoice = "Paper";
        } else {
            computerChoice = "Scissors";
        }


        System.out.println("Computer Chose: " + computerChoice);

        viewImage.setImage(new Image("file:C:/Users/user/Downloads/Project Java/" + rockGame.getComputer() + ".png"));

        if (result == 0) {
            System.out.println("Computer Wins!");
        } else if (result == 1) {
            System.out.println("It's a Tie!");
        } else {
            System.out.println("You Win!");
        }
    }
```

What is missing

- 2 Player Game Mode
- Menu Bar
- Text displayed on the window