Dice Split Project Documentation

Table of Contents

1. Project Overview;

Project Name: Dice Split

Description: Dice Split is an engaging game application where players roll dice to achieve high scores. The game keeps track of the highest score achieved, allowing players to challenge themselves and others. The application features a simple user interface and includes sound effects, animations and file I/O to enhance the gaming experience.

2. Features

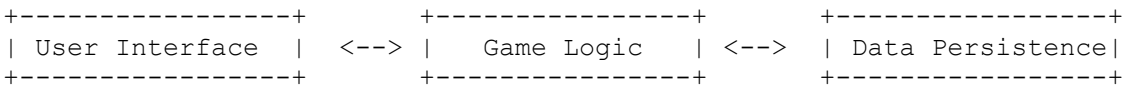
- **Dice Rolling:** Players can roll virtual dice to get random values.
- **High Score Tracking:** The game automatically tracks and updates the highest score achieved by any player using file I/O
- **User Interface:** The game features a simple and intuitive user interface that is easy to navigate, and makes the game enjoyable to play.
- **Sound Effects:** Background music and sound effects are included to enhance the gaming experience.

3. Architecture Overview

The Dice Split project follows a modular architecture, which includes the following components:

- **User Interface:** Manages user interactions and displays the game interface.
- **Game Logic:** Contains the core logic for rolling dice, calculating scores, and determining high scores.
- **Data Persistence:** Handles the storage and retrieval of high scores using an array list.

Architecture Diagram



4. Installation Instructions

Follow these steps to set up and run the Dice Split application on your local machine:

Prerequisites

- **Java Development Kit (JDK):** Ensure you have JDK 16 or later installed.
- **Gradle:** Ensure you have Gradle installed. You can download it from the Gradle website.

Steps

1. **Clone the Repository:**

   ```
   bash

   git clone <repository-url>
   cd Dice-Split
   ```

2. **Build the Project:** Use Gradle to build the project:

   ```
   bash

   ./gradlew build
   ```

3. **Run the Application:** Use Gradle to run the application:

   ```
   bash

   ./gradlew run
   ```

5. Usage Instructions

Starting the Game

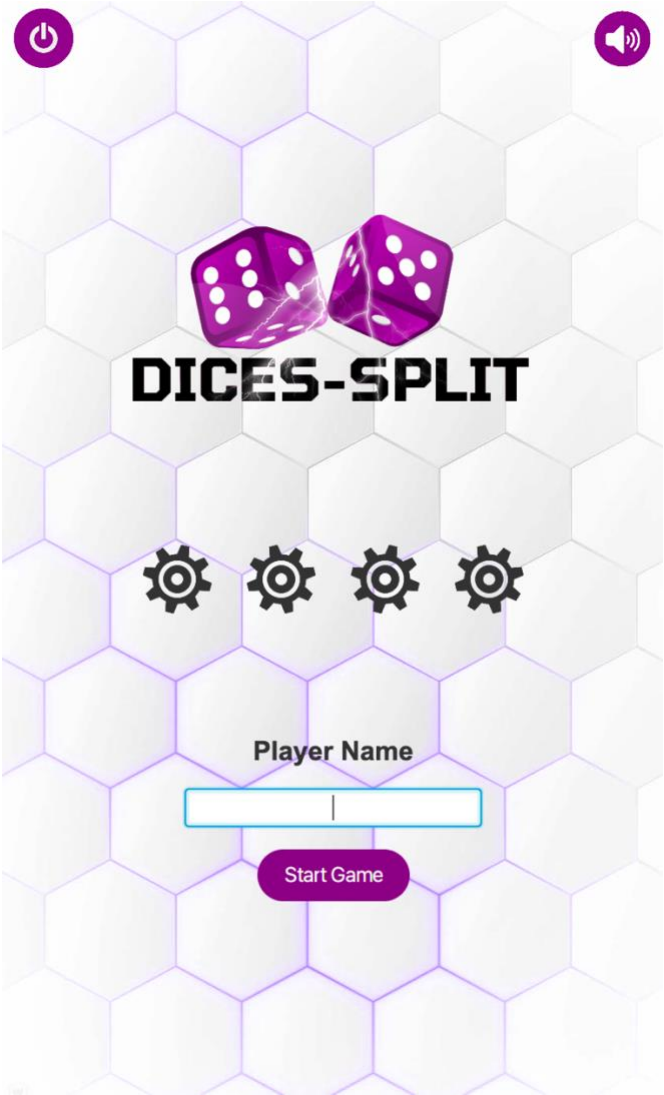
1. **Launch the Application:** After running the application, a window will open with the game interface.

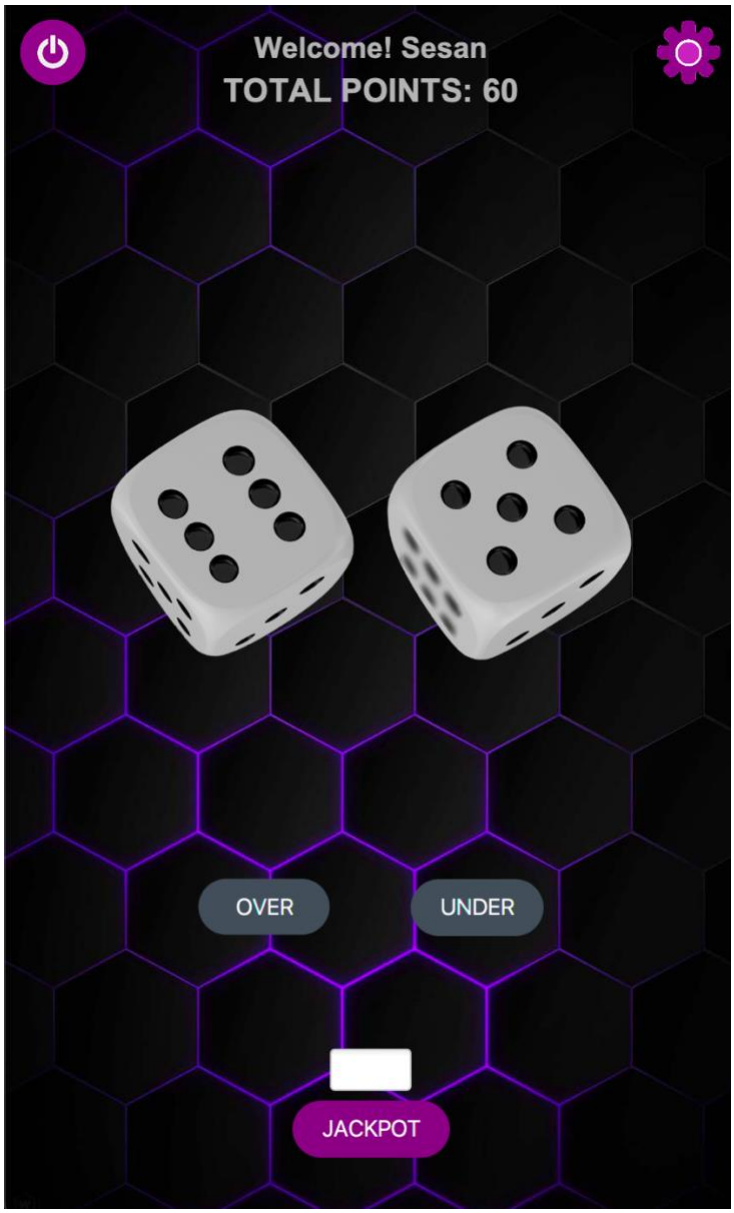
2. **Begin Playing:**
   - Click the "Over, Under or Jackpot" button to roll the dice.
   - The result of the dice roll will be displayed on the screen.

Gameplay

- **Rolling Dice:** Click the " Over, Under or Jackpot " button to roll the dice.
- **Scoring:** Starting balance is 60 points, you get 12 points for a win and lose 12 points for a loss. For jackpot, you get 120 points for a win and lose 24 points for a loss. Try to achieve the highest score possible.
- **High Score:** The highest score achieved is saved in the `Highscore.txt` file located in the root directory of the project.

Screenshots





6. Development and Build Instructions

Project Structure

- `src/`: Contains the source code, organized into packages.
- `build/`: Contains the build artifacts generated by Gradle.
- `gradle/`: Contains Gradle-specific files and configurations.
- `Music/`: Contains audio files used in the game.
- `Highscore.txt`: Stores the highest score achieved in the game.

Building the Project

Use the Gradle wrapper scripts to build the project:

```
bash
./gradlew build
```

Running Tests

Run the unit tests to ensure the project is working correctly:

```
bash
./gradlew test
```

7. Code Snippets

Rolling Dice

The following code snippet demonstrates how to roll a dice and get a random value between 1 and 6:

```
java

import java.util.Random;

public int rollDice() {
    Random random = new Random();
    return random.nextInt(6) + 1; // Dice roll between 1 and 6
}
```

Updating High Score

The following code snippet demonstrates how to update the high score if a new high score is achieved:

```
java

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public void updateHighScore(int score) {
    int currentHighScore = getHighScore();
    if (score > currentHighScore) {
        saveHighScore(score);
    }
}
```

```
public int getHighScore() {
    // Code to read the high score from Highscore.txt
    // ...
}

public void saveHighScore(int score) {
    try (BufferedWriter writer = new BufferedWriter(new
FileWriter("Highscore.txt"))) {
        writer.write(String.valueOf(score));
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

8. Troubleshooting

Common Issues

1. **Build Failures:**
    - Ensure all dependencies are correctly installed.
    - Check the Gradle configurations in the `build.gradle` and `settings.gradle` files.
2. **Runtime Errors:**
    - Check for error messages in the console and refer to the documentation.
    - Ensure all required files (e.g., `Highscore.txt`, audio files) are present in the expected directories.

Debugging Tips

- **Logging:** Use logging to trace issues and understand the application's flow.
- **Debugging Tools:** Use an IDE with debugging capabilities to set breakpoints and inspect variables.

9. Additional Resources

- **Gradle Documentation:** Gradle User Manual
- **Java Documentation:** Java SE Documentation
- **Git Documentation:** Pro Git Book

10. Application Building Process Documentation

Overview

This section provides a detailed step-by-step guide on the process of building the Dice Split application, from setting up the development environment to packaging the application for distribution.

Setting Up the Development Environment

1. **Install JDK:**
   - Download and install the Java Development Kit (JDK) from the Oracle website.
2. **Install Gradle:**
   - Download and install Gradle from the Gradle website.
3. **Clone the Repository:**

```
bash

git clone <repository-url>
cd Dice-Split
```

Building the Application

1. **Navigate to the Project Directory:**

```
bash

cd Dice-Split
```

2. **Run the Build Command:**
   - Use the Gradle wrapper script to build the project:

```
bash

./gradlew build
```

3. **Check Build Artifacts:**
   - The build artifacts will be located in the `build/` directory. This directory contains compiled classes, packaged JAR files, and other build outputs.

Running the Application

1. **Run the Application:**
   - Use the Gradle wrapper script to run the application:

```
bash
```

```
./gradlew run
```

2. **Verify the Application:**
   - Ensure that the application starts correctly and that you can interact with the user interface to roll the dice and track high scores.

Packaging the Application

1. **Create a Distributable Package:**
   - Use Gradle to create a distributable package (e.g., a JAR file):

```
bash

./gradlew jar
```

2. **Locate the Package:**
   - The JAR file will be located in the `build/libs/` directory. This file can be distributed to users for running the application.

11. Licensing Information

License

The Dice Split application is licensed under the MIT License. This means you are free to use, modify, and distribute the software, provided that you include the original copyright and permission notice in any copies or substantial portions of the software.

License Text

```
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Applying the License

To apply the MIT License to your project, include the license text in a `LICENSE` file at the root of your repository and add the following header to your source code files:

java

```
/*
 * MIT License
 *
 * Copyright (c) [2024] [Sesan Popoola, Abdul Aziz]
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
copy
 * of this software and associated documentation files (the "Software"), to
deal
 * in the Software without restriction, including without limitation the
rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE
 * SOFTWARE.
 */
```
