SKILL DEVELOPMENT

Project Report

DLithe Consultancy Services Pvt. Ltd.

DLithe™
Be Agile..Be in Demand

PROJECT REPORT ASSESSMENT

Student Name: Spoorti Arunkumar Doddamani

USN: 2JR23CS106

Assignment: Java

Organization: DLithe Consultancy Services Pvt. Ltd

Supervisor's Name: Archana SM

Submitted to

Signature of Training Supervisor

Date:

Signature of Students

Date

DLithe™
Be Agile..Be in Demand

TABLE OF CONTENTS

INTRODUCTION

The Rock-Paper-Scissors game is a classic hand game played between two people, where each player simultaneously forms one of three shapes with their hand: rock, paper, or scissors. It is a simple yet engaging game based on rules of logic and chance. The game has been used in various cultures as a decision-making tool and is widely known for its simplicity and fun.

This project aims to develop a digital version of the Rock-Paper-Scissors game using the Java programming language. The main objective is to allow a user to play against the computer in a console-based environment. In this implementation, the player is prompted to enter a choice among three options: Rock (0), Paper (1), or Scissors (2). The computer then randomly selects its choice, then displays both the choices using ASCII art, and then determines the winner based on the standard rules:

Rock beats Scissors

Scissors beats Paper

Paper beats Rock

The project demonstrates the use of basic programming concepts such as conditional statements, loops, arrays, random number generation, and user input handling. It also incorporates visual elements using ASCII art to enhance user interaction. This project serves as a foundational exercise for beginners in programming and helps in understanding core Java concepts in a fun and interactive way.

DLithe™
Be Agile..Be in Demand

ROCK-PAPER-SCISSORS REQUIREMENT

• Problem statement

The aim of this project is to develop a simple, interactive Rock-Paper-Scissors game using Java. The game should allow a human user to play against the computer in a command-line environment. The computer should make random choices, and the game must determine the winner based on classic rules.

• Objective

- To develop a user-friendly console-based game using Java.
- To implement logic for determining the winner.
- To handle invalid user inputs.
- To enable repeat play until the user decides to quit.

• Functional requirements:

- Game Input: The system shall prompt the user to enter a numeric value to select a move: 0 for Rock, 1 for Paper and 2 for Scissors. The system shall validate the user's input, if the input is not 0, 1, or 2, it shall display an appropriate error message. Then the system shall re-prompt the user until a valid input is received.
- Computer Move: The system shall randomly generate the computer's move from the three options (Rock, Paper, or Scissors), each move shall have an equal probability of being selected.
- Game Logic: The system shall compare the user's move with the computer's move. The result of the comparison shall be one of the following outcomes: User Wins (e.g., Rock beats Scissors), User Loses and Draw (both players choose the same move).
- Visual Display: The system shall display ASCII art that represents: The user's selected move and computer's selected move. The ASCII art must clearly indicate whether it represents Rock, Paper, or Scissors.
- Game Continuation: After displaying the outcome, the system shall prompt the user to play again with the message: "Play again? yes or no" the system behavior shall be as follows: If the user inputs "yes" start a new game round. If the user inputs "no" terminate the game with a closing message. For any other input display an error message and prompt again until a valid response is received either "yes" or "no".

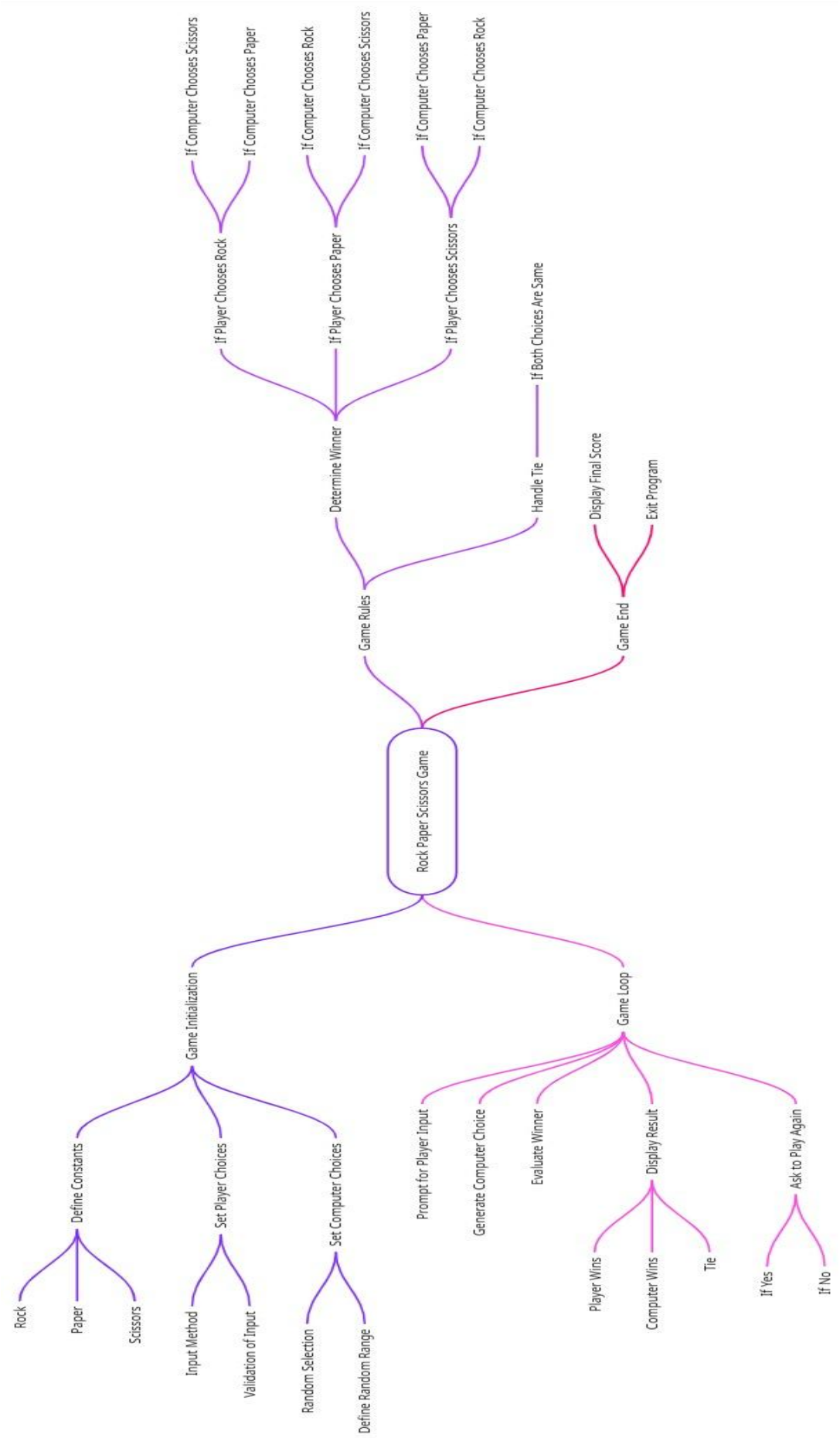
- **Non - Functional requirements**
  - Platform: Console-based Java application.
  - User-Friendliness: Clear prompts and friendly messages.
  - Input Validation: Prevent invalid input crashes.
  - Performance: Should respond instantly.
  - Portability: Should run on any system with Java installed.
  - Maintainability: Code should be clean, readable, and modular (optionally with functions).

- **Error handling**
  - Invalid number inputs (not 0, 1, or 2) will prompt the user to re-enter.
  - Incorrect response to “Play again?” prompt (not "yes"/"no") will ask again.

- **System Requirements**
  - Software: Java 17 or higher, any Java-supported IDE (Eclipse, IntelliJ, VS Code)
  - Hardware: Minimum 2 GB RAM, 1 GHz processor
  - Operating System: Windows / Linux / macOS

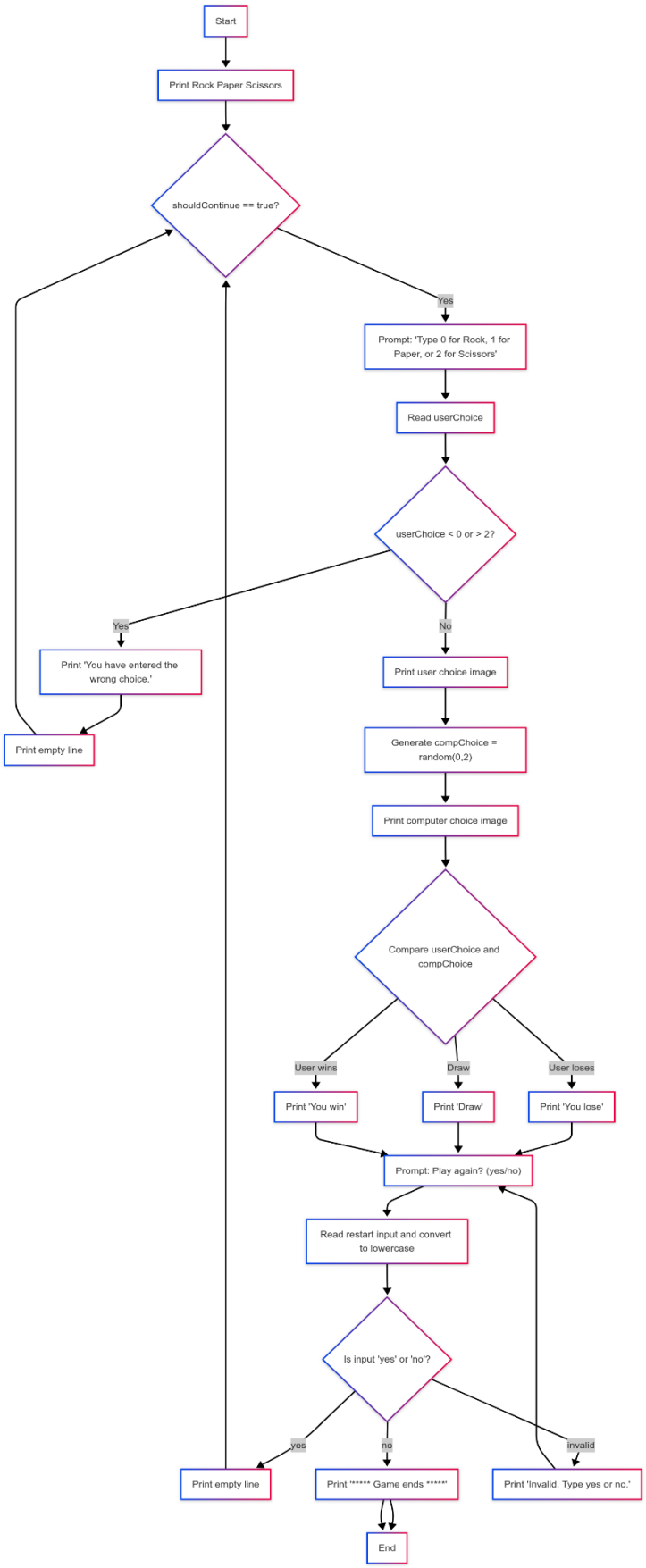
DLithe™
Be Agile..Be in Demand

MIND MAP



DLithe™
Be Agile..Be in Demand

FLOW CHART

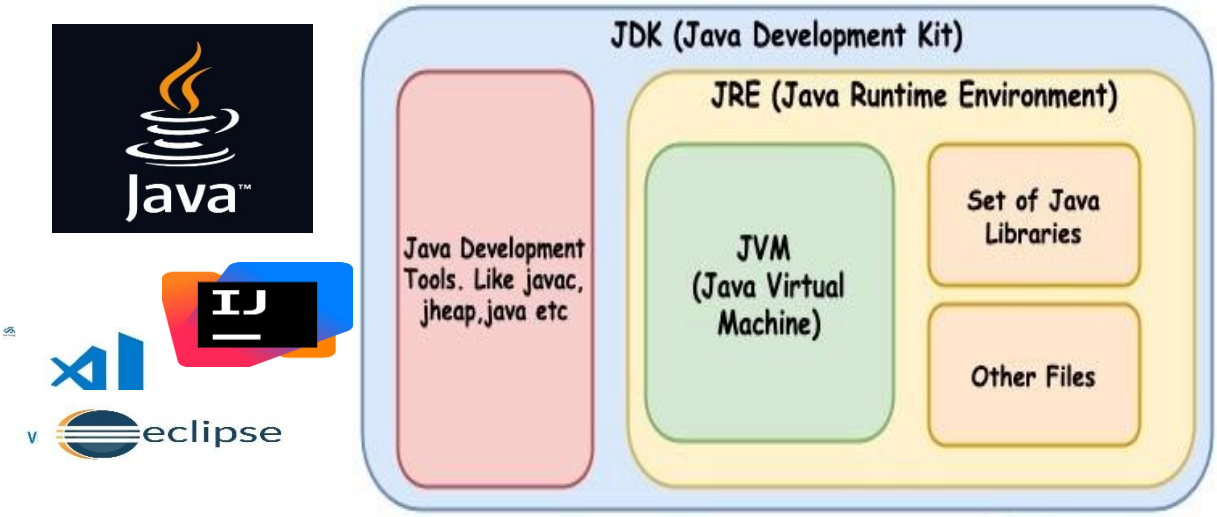


DLithe™
Be Agile..Be in Demand

PROJECT DEVELOPMENT IMAGES

```java
import java.util.Scanner;
import java.util.Random;
public class RockPaperScissorsGame {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        String[] gameImage = {
            """
                _______
            ---'    ____)
                  (_____)
                  (_____)
                  (____)
            ---.__(___)
            """,
            """
                _______
            ---'    ____)____
                      ______)
                     _______)
                    _______)
            ---.__________)
            """,
            """
                _______
            ---'    ____)____
                      ______)
                   __________)
                  (____)
            ---.__(___)
            """
        };boolean shouldContinue = true;
        System.out.println(x:"***** Rock Paper Scissors *****");
        while (shouldContinue){
            System.out.print(s:"Type 0 for Rock, 1 for Paper or 2 for Scissors \nEnter your choice: ");
            int userChoice = scanner.nextInt();
            scanner.nextLine();
            if (userChoice < 0 || userChoice > 2) {
                System.out.println(x:"You have entered the wrong choice. ");
                System.out.println();
                continue;}
            else {
                System.out.println("Your choice is:\n" + gameImage[userChoice]);
            }int compChoice = random.nextInt(bound:3);
            System.out.println("Computer choice is:\n" + gameImage[compChoice]);
            if ((userChoice == 0 && compChoice == 1) || (userChoice == 1 && compChoice == 2) || (userChoice == 2 && compChoice == 0)) {
                System.out.println(x:"You lose");
            } else if (userChoice == compChoice) {
                System.out.println(x:"Draw");
            }else {
                System.out.println(x:"You win");
            }boolean restartInvalid = true;
            while (restartInvalid) {
                System.out.print(s:"Play again? yes or no: ");
                String restart = scanner.nextLine().toLowerCase();
                if (restart.equals(anObject:"yes")) {
                    System.out.println();
                    shouldContinue = true;
                    restartInvalid = false;
                } else if (restart.equals(anObject:"no")) {
                    System.out.println(x:"***** Game ends *****");
                    shouldContinue = false;
                    restartInvalid = false;}
                else {
                    System.out.println(x:"Invalid. Type 'yes' or 'no'");}
            }}
        scanner.close();}}
```

TECHNOLOGIES USED



- **Programming Language:**

  Java: The core language used to implement the game logic. Java is platform-independent and supports object-oriented programming, making it ideal for simple interactive applications.

- **Integrated Development Environment (IDE):**

  Eclipse / IntelliJ IDEA / Visual Studio Code (VS Code): These IDEs were used for writing, editing, debugging, and running the Java code. They provide syntax highlighting, error checking, and code suggestions that improve productivity.

- **Java Libraries:**

  java.util.Scanner: Used to take input from the user during gameplay.

  java.util.Random: Used to generate random choices for the computer (Rock, Paper, or Scissors).

- **Deployment Environment:**
  - Console-based Execution:

    The game is deployed and executed using a command-line interface (CLI). This allows the game to be simple, lightweight, and easy to run on any system with a Java Runtime Environment (JRE).
  - Java Development Kit (JDK):

    A complete environment for Java development, including the compiler (javac) and Java Runtime Environment (JRE) necessary to compile and run the application.

DLithe™
Be Agile..Be in Demand

OUTPUT

```
***** Rock Paper Scissors *****
Type 0 for Rock, 1 for Paper or 2 for Scissors
Enter your choice: 1
Your choice is:

    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)

Computer choice is:

    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

You lose
Play again? yes or no: YES

Type 0 for Rock, 1 for Paper or 2 for Scissors
Enter your choice: 2
Your choice is:

    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

Computer choice is:

    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)

You win
Play again? yes or no: NO
***** Game ends *****
```

```
***** Rock Paper Scissors *****
Type 0 for Rock, 1 for Paper or 2 for Scissors
Enter your choice: 5
You have entered the wrong choice.

Type 0 for Rock, 1 for Paper or 2 for Scissors
Enter your choice: 2
Your choice is:

    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

Computer choice is:

    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)

You lose
Play again? yes or no: YEA
Invalid. Type 'yes' or 'no'
Play again? yes or no: YES

Type 0 for Rock, 1 for Paper or 2 for Scissors
Enter your choice: 1
Your choice is:

    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)

Computer choice is:

    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)

You win
Play again? yes or no: NO
***** Game ends *****
```

DLithe™
Be Agile..Be in Demand

TRAINING EXPERIENCE

During the skill development training program focused on Java programming, I gained a strong foundation in both core and object-oriented concepts of the language. The training started with an introduction to Java's basic syntax, data types, operators, and control structures, which helped me become comfortable with writing simple programs and understanding logic flow. As the sessions progressed, I was introduced to Object-Oriented Programming (OOP) concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction. I learned how to design and implement reusable and modular code using these principles, which greatly enhanced my programming skills. One of the key highlights of the training was hands-on experience through mini-projects and coding challenges. We developed real-world applications which improved my problem-solving abilities.

Additionally, the training covered essential Java libraries like java.util.Scanner for input handling and java.util.Random for generating random data, which were particularly useful in building interactive console-based applications. We also explored exception handling and learned how to build error-resilient programs.

The training sessions were conducted using Eclipse and IntelliJ IDEs, which helped me understand code structuring, debugging, and version control practices. I also gained insights into Java's memory management and the use of the Java Virtual Machine (JVM). Towards the end of the training, we implemented a few real-time applications. One such mini-project was the Rock Paper Scissors game, where I applied control structures, loops, and conditional logic, along with random number generation. These projects gave me confidence in applying Java for logic-building and user interaction.

This training provided me with a solid foundation in Java and prepared me for further exploration into advanced topics like multithreading, GUI-based applications, and database connectivity. It also instilled a habit of writing clean, well-commented, and maintainable code. Overall, the skill development experience in Java was highly enriching and has significantly contributed to my growth as a budding software developer. It not only improved my programming proficiency but also taught me how to approach problems logically and systematically.

DLithe™
Be Agile..Be in Demand

KEY LEARNINGS

- **Strong Foundation in Java Programming:**
  - Gained a clear understanding of Java syntax, data types, operators, loops, and conditionals. Developed logic-building skills to solve real-world problems using Java.
- **Object-Oriented Programming (OOP):**
  - Learned and implemented core OOP concepts such as Classes, Objects, Inheritance, Polymorphism, Encapsulation, and Abstraction.
  - Understood how to design modular, reusable, and maintainable code using OOP principles.
- **Practical Use of Java Libraries:**
  - Used built-in Java libraries like java.util.Scanner for input handling and java.util.Random for random operations. Learned how to import, use, and structure external and standard Java classes effectively.
- **IDE Proficiency:**
  - Became comfortable working with professional development environments like Eclipse, IntelliJ IDEA, and VS Code. Explored features like auto-suggestions, debugging tools, and error navigation to speed up development.
- **Error Handling and Debugging:**
  - Gained the ability to write robust code using exception handling with try-catch blocks. Learned techniques to identify, trace, and fix bugs efficiently.
- **Hands-On Mini-Projects:**
  - Applied knowledge in mini-projects such as Rock Paper Scissors using control structures, functions, and user interaction.
  - Understood the full cycle of coding, testing, and running console-based applications.
- **Best Coding Practices:**
  - Adopted clean coding standards including meaningful naming, indentation, and inline documentation.
  - Learned the importance of code readability, modularity, and version control basics.
- **Introduction to Software Development Lifecycle:**
  - Gained exposure to the steps involved in planning, designing, coding, testing, and maintaining Java applications.

CHALLENGES

- **Input Validation:**
  - Ensuring the user enters only valid inputs (0, 1, or 2) was a key challenge. Handling unexpected inputs like characters or numbers outside the allowed range required implementing proper checks and loops.
- **Game Logic Accuracy:**
  - Writing the correct logic to determine the winner (win, lose, draw) took careful condition checking. Small logical errors sometimes led to incorrect results.
- **Looping Structure and Replay Option:**
  - Implementing the feature to allow players to replay the game using loops and conditionals without restarting the program from scratch required proper flow control.
- **Use of Arrays and Random Function:**
  - Storing ASCII images in an array and using Random to generate computer moves was tricky at first. Ensuring the correct image mapped to the right move took attention to indexing.
- **Console-Based UI Design:**
  - Presenting the game in an understandable and engaging way using only text and symbols in the console was a creative challenge, especially aligning the ASCII art.
- **Code Readability and Modularity:**
  - Keeping the code clean and well-organized became a challenge as the program grew. Proper indentation, naming, and breaking logic into manageable sections helped improve readability.
- **Handling String and Integer Inputs Together:**
  - Mixing Scanner.nextInt() with Scanner.nextLine() sometimes caused input buffering issues, which required an understanding of Java's input system to fix.

APPLICATIONS

- **Learning Tool for Beginners**
  - This project is widely used as an introductory assignment in Java programming courses. Helps students learn essential programming concepts such as conditionals (if-else), loops, arrays, input handling (Scanner), and random number generation.
  - Reinforces logic-building and decision-making skills by requiring the programmer to compare choices and determine the winner.
- **Testing Platform for Game Logic**
  - The game provides a base platform to experiment with game development logic.
  - Programmers can test and understand how random behavior, user input, and state management work in a basic turn-based system. Serves as a foundation for building more complex games involving AI or multiplayer logic.
- **Demonstration of Console-Based UI**
  - Showcases how a basic user interface can be created using text and ASCII art in the command line. Useful in environments where GUIs are not available or needed, such as embedded systems or terminal-based applications.
- **Simulation of Probability and Fairness**
  - Can be extended to simulate large numbers of games and analyze probabilities of outcomes. Useful in teaching concepts of fairness, randomness, and statistical outcomes in both game theory and mathematics.
- **Foundation for AI and Machine Learning Projects**
  - This project can serve as a starting point for building an AI opponent that learns patterns in user input.
  - Developers can apply reinforcement learning algorithms or predictive modeling techniques to make smarter decisions based on past user moves.
- **Reusable Game Engine**
  - The basic logic of this game can be re-used and extended into other rule-based or logic-based games. Can be part of a larger game hub or integrated with GUIs, mobile apps, or web applications for interactive experiences.
- **Engagement and Entertainment**
  - Provides simple entertainment and engagement during idle times.
  - Can be used in classrooms, coding bootcamps, or workshops as a fun activity to encourage collaboration and coding.

CONCLUSION

The Rock Paper Scissors Game project provided a practical and engaging way to apply core Java programming concepts in a real-world scenario. Through the development of this game, I gained valuable hands-on experience with fundamental constructs such as conditional statements, loops, arrays, user input handling, and random number generation using the Scanner and Random classes from Java's utility libraries. This project also helped in improving logical thinking, problem-solving skills, and understanding the importance of code structure and flow control. Moreover, the implementation of input validation and game replay logic offered insight into building interactive and user-friendly console-based applications. Although simple in concept, the project lays a strong foundation for exploring more advanced topics such as GUI-based games, AI implementation, and networked multiplayer applications in future developments. In summary, this project not only enhanced my Java programming skills but also deepened my understanding of how to design and implement an interactive software solution, making it a significant step in my journey as a software developer.