VIVEKANANDA GLOBAL UNIVERSITY

Master of Computer Applications

Object Oriented Programming Using JAVA (PGCSA111)

Project Title: Developing a Number Guessing Game using Java.

Project Report

PROJECT GUIDE:
Mr. NARAYAN VYAS

SUBMITTED BY:
ANSHU PATEL 24CSA3BC086
ANKIT KUMAR 24CSA3BC125 KUSHWAHA

ACKNOWLEDGEMENT

I have taken this opportunity to express my gratitude and humble regards to the Vivekananda Global University to provide an opportunity to present a project on the " Developing a Number Guessing Game".

I would also be thankful to my project guide Mr. Narayan Vyas to help me in the completion of my project and the documentation. I have taken efforts in this project, but the success of this project would not be possible without their support and encouragement.

I would like to thank our dean "Dr. R C Tripathi" to help us in providing all the necessary books and other stuffs as and when required .I show my gratitude to the authors whose books has been proved as the guide in the completion of my project I am also thankful to my classmates and friends who have encouraged me in the course of completion of the project.

Thanks

Place : Jaipur
Date : 29-03-2025

DECLARATION

We hereby declare that this Project Report titled "Developing a Number Guessing Game" submitted by us and approved by our project guide, to the Vivekananda Global University, Jaipur is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project Guide:
Mr. Narayan Vyas

Student Name:
Anshu Patel 24CSA3BC086
Ankit Kumar Kushwaha 24CSA3BC125

Table of Contents

12. Challenges and Solutions
    - Input Handling
    - File Errors
    - Sorting Tie-breakers
    - Data Persistence

13. Educational Value
14. Conclusion
15. References

Developing a Number Guessing Game using Java

Abstract:

This report presents a detailed documentation of the Number Guessing Game, a Java-based console application developed to demonstrate essential programming concepts. The project is designed to reinforce core Java topics including conditionals, loops, user input handling, object-oriented programming, and file I/O. The game offers three difficulty levels, a seven-attempt gameplay system, and a leaderboard that stores the top three best scores.

The project promotes learning through interaction, offering immediate feedback and the option for repeated playthroughs. By incorporating persistent storage using text files, the game illustrates file manipulation and basic data structures in Java. Additionally, features such as emoji-based feedback and user-friendly prompts create an engaging and accessible interface. The leaderboard logic exemplifies sorting and comparison operations, further enriching the educational experience.

This report outlines the full lifecycle of the game's development — from planning and design to implementation, testing, and potential improvements — and serves as a case study for learning Java programming through a real-world application

1. Introduction:

The Number Guessing Game is a console-based Java application that challenges users to guess a randomly generated number within a specific range. The game is intended both as a fun pastime and an educational tool for beginner to intermediate Java programmers. It allows players to select difficulty levels, receive feedback on their guesses, and view a persistent leaderboard. This simple game incorporates many programming principles such as classes, objects, loops, conditional statements, and file handling**.**

The motivation behind this project is to bridge theoretical learning with practical implementation. Instead of reading abstract programming concepts, students can apply them in a hands-on project that is both enjoyable and instructive. By completing this project, users will gain experience in designing and implementing logic-based applications in Java.

2. Objectives:

The primary objectives of this project include:

- To implement a number guessing game using the Java programming language.
- To practice core programming concepts including control flow, loops, conditionals, and data handling.
- To develop and manage game logic for difficulty levels and attempt restrictions.
- To implement file-based storage for a persistent leaderboard.
- To provide a user-friendly interface through console interaction.
- To design the code using object-oriented principles.
- To identify opportunities for enhancements and improvements.

3. System Requirements:

3.1 Programming Language

- Java (version 8 or above)

3.2 Integrated Development Environments (IDEs)

- IntelliJ IDEA
- Eclipse
- NetBeans
- VS Code with Java extension

3.3 Platforms

- Cross-platform (Windows, macOS, Linux)
- Java Runtime Environment (JRE) required

3.4 External Resources

- No external libraries used
- Uses standard Java API libraries

4. Features:

1 4.1 Random Number Generation

Uses `Random` class to generate a number within a defined range based on difficulty level.

4.2 User Input

Utilizes `Scanner` to take input from users, including names and guesses.

4.3 Difficulty Levels

Three modes:

- Easy: 1 to 50
- Medium: 1 to 100
- Hard: 1 to 500

4.4 Limited Attempts

Maximum of 7 guesses per game to increase challenge.

4.5 Feedback Mechanism

Displays whether each guess is too high, too low, or correct.

4.6 Leaderboard System

Tracks and displays the top three players based on the fewest attempts

4.7 Data Persistence

Reads from and writes to a file (`leaderboard.txt`) to store top scores.

4.8 Emoji Integration

Enhances user interaction with celebratory (?) and failure (✕) emojis.

4.9 Replayability

Allows the user to play multiple rounds until they choose to stop.

5. System Design: The system consists of two main components:

1. Class Overview

- **Player Class:**
  Stores player name and number of attempts.
  Provides serialization and deserialization for file operations.

- **Main Class:**
  Contains game loop,logic,user interaction, and leaderboard management.

2. Data Flow

- Input: User enters name, difficulty level, and guesses.
- Process: Game logic determines result.
- Output: Game summary and leaderboard updates.

6. Implementation Details:

1. Game Loop

Implemented using a `do-while` loop to support replayability.

2. Difficulty Logic

Switch-case structure sets maximum number according to user choice.

3. Number Guessing Algorithm

Simple conditionals compare the guess to the target number.

4. Leaderboard Logic

Sorts all player scores, keeps the best three using `Comparator`.

5. File I/O

Uses `BufferedReader`, `FileWriter`, and `PrintWriter` for reading and writing leaderboard data.

6. Game Summary

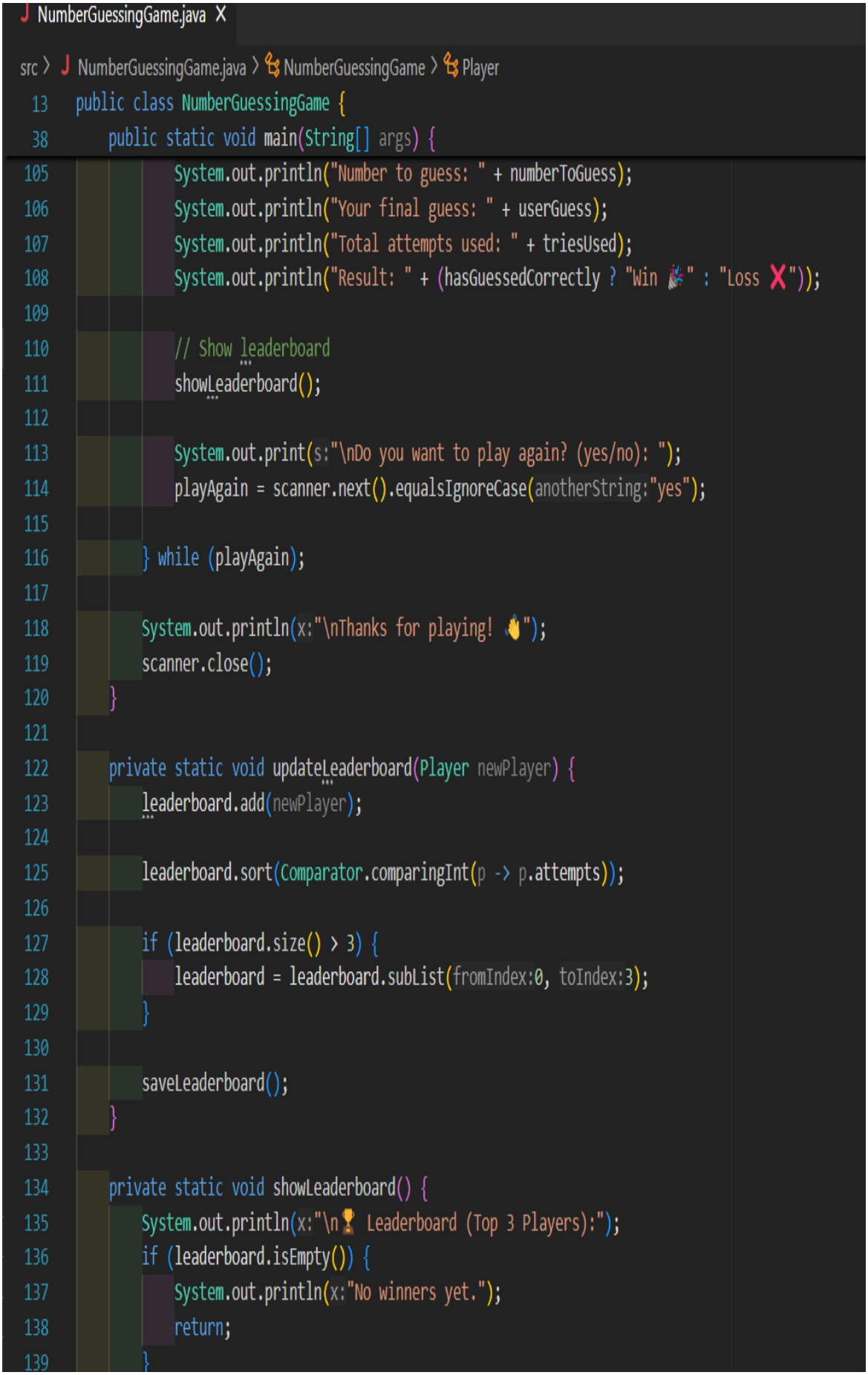
Prints a recap of the game including attempts used, guessed number, and result.

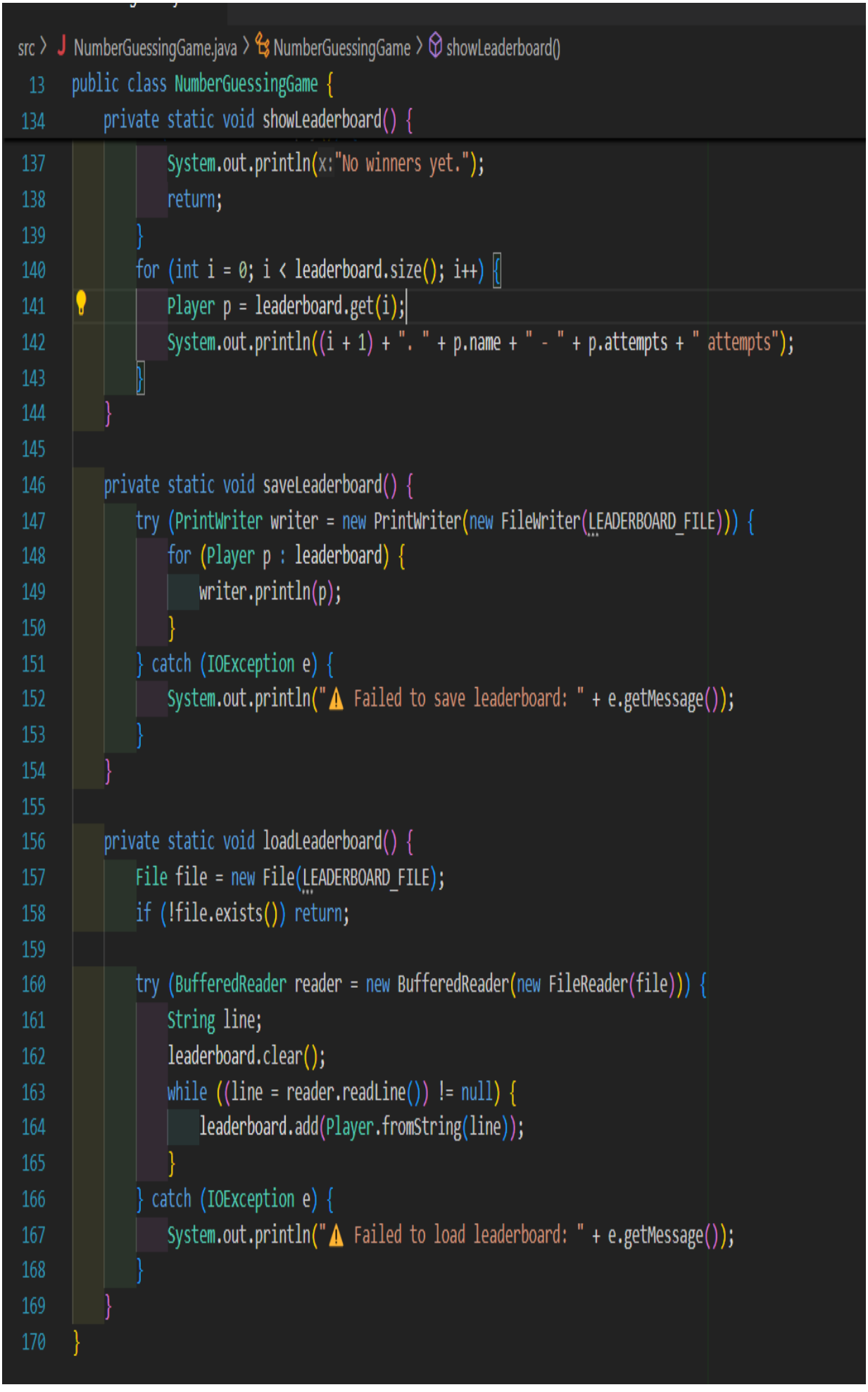
src > NumberGuessingGame.java > NumberGuessingGame > Player

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Random;
import java.util.Scanner;

public class NumberGuessingGame {

    static class Player {
        String name;
        int attempts;

        Player(String name, int attempts) {
            this.name = name;
            this.attempts = attempts;
        }

        @Override
        public String toString() {
            return name + "," + attempts;
        }

        public static Player fromString(String line) {
            String[] parts = line.split(regex:",");
            return new Player(parts[0], Integer.parseInt(parts[1]));
        }
    }

    static final String LEADERBOARD_FILE = "leaderboard.txt";
    static List<Player> leaderboard = new ArrayList<>();
```

```java
public class NumberGuessingGame {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        boolean playAgain;

        // Load existing leaderboard from file
        loadLeaderboard();

        System.out.println("🎮 Welcome to the Number Guessing Game with Leaderboard!");

        do {
            System.out.print("\nEnter your name: ");
            String playerName = scanner.next();

            // Select difficulty
            int maxRange = 100;
            System.out.println("\nSelect Difficulty Level:");
            System.out.println("1. Easy (1 to 50)");
            System.out.println("2. Medium (1 to 100)");
            System.out.println("3. Hard (1 to 500)");
            System.out.print("Enter your choice (1-3): ");
            int level = scanner.nextInt();

            switch (level) {
                case 1 -> maxRange = 50;
                case 2 -> maxRange = 100;
                case 3 -> maxRange = 500;
                default -> {
                    System.out.println("Invalid choice. Defaulting to Medium.");
                    maxRange = 100;
                }
            }

            int numberToGuess = random.nextInt(maxRange) + 1;
            int userGuess = 0;
```

```
13  public class NumberGuessingGame {
38      public static void main(String[] args) {
71          int numberToGuess = random.nextInt(maxRange) + 1;
72          int userGuess = 0;
73          int maxTries = 7;
74          int triesUsed = 0;
75          boolean hasGuessedCorrectly = false;
76
77          System.out.println("\nI've selected a number between 1 and " + maxRange + ".");
78          System.out.println("You have " + maxTries + " attempts. Good luck!");
79
80          while (triesUsed < maxTries) {
81              System.out.print("Attempt " + (triesUsed + 1) + ": Enter your guess: ");
82              userGuess = scanner.nextInt();
83              triesUsed++;
84
85              if (userGuess < numberToGuess) {
86                  System.out.println(x:"Too low!");
87              } else if (userGuess > numberToGuess) {
88                  System.out.println(x:"Too high!");
89              } else {
90                  hasGuessedCorrectly = true;
91                  System.out.println("🎉 Correct! You guessed the number in " + triesUsed + " tries.");
92                  updateLeaderboard(new Player(playerName, triesUsed));
93                  break;
94              }
95          }
96
97          if (!hasGuessedCorrectly) {
98              System.out.println("❌ Out of attempts! The number was: " + numberToGuess);
99          }
100
101         // Show game summary
102         System.out.println(x:"\n📊 Game Summary:");
103         System.out.println("Player: " + playerName);
104         System.out.println("Level: " + (level == 1 ? "Easy" : level == 2 ? "Medium" : "Hard"));
105         System.out.println("Number to guess: " + numberToGuess);
```


NumberGuessingGame.java
src > NumberGuessingGame.java > NumberGuessingGame > Player
public class NumberGuessingGame {
public static void main(String[] args) {
System.out.println("Number to guess: " + numberToGuess);
System.out.println("Your final guess: " + userGuess);
System.out.println("Total attempts used: " + triesUsed);
System.out.println("Result: " + (hasGuessedCorrectly ? "Win 🎉" : "Loss ❌"));
// Show leaderboard
showLeaderboard();
System.out.print(s:"\nDo you want to play again? (yes/no): ");
playAgain = scanner.next().equalsIgnoreCase(anotherString:"yes");
} while (playAgain);
System.out.println(x:"\nThanks for playing! 👋");
scanner.close();
}
private static void updateLeaderboard(Player newPlayer) {
leaderboard.add(newPlayer);
leaderboard.sort(Comparator.comparingInt(p -> p.attempts));
if (leaderboard.size() > 3) {
leaderboard = leaderboard.subList(fromIndex:0, toIndex:3);
}
saveLeaderboard();
}
private static void showLeaderboard() {
System.out.println(x:"\n🏆 Leaderboard (Top 3 Players):");
if (leaderboard.isEmpty()) {
System.out.println(x:"No winners yet.");
return;
}


src > NumberGuessingGame.java > NumberGuessingGame > showLeaderboard()
13 public class NumberGuessingGame {
134 private static void showLeaderboard() {
137 System.out.println(x:"No winners yet.");
138 return;
139 }
140 for (int i = 0; i < leaderboard.size(); i++) {
141 Player p = leaderboard.get(i);
142 System.out.println((i + 1) + ". " + p.name + " - " + p.attempts + " attempts");
143 }
144 }
145
146 private static void saveLeaderboard() {
147 try (PrintWriter writer = new PrintWriter(new FileWriter(LEADERBOARD_FILE))) {
148 for (Player p : leaderboard) {
149 writer.println(p);
150 }
151 } catch (IOException e) {
152 System.out.println("⚠ Failed to save leaderboard: " + e.getMessage());
153 }
154 }
155
156 private static void loadLeaderboard() {
157 File file = new File(LEADERBOARD_FILE);
158 if (!file.exists()) return;
159
160 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
161 String line;
162 leaderboard.clear();
163 while ((line = reader.readLine()) != null) {
164 leaderboard.add(Player.fromString(line));
165 }
166 } catch (IOException e) {
167 System.out.println("⚠ Failed to load leaderboard: " + e.getMessage());
168 }
169 }
170 }

8. User interface and Experience:

The interface is simple and text-based, suitable for console use. Emoji feedback and clear prompts enhance usability. Despite being console-based, the experience remains engaging through its pacing, instructions, and visual elements like progress indicators.

9. Leaderboard File Format:

→ **Format:** Each line in the leaderboard file follows the format : playerName, Attempts

.

→Update Rules:

Scores are added if the player wins.

Leaderboard retains only top 3 players based on the fewest attempts

.

→ Sample:

Alice,3
Bob,5
Charlie,6

```
erGuessingGame.java\bin NumberGuessingGame "
? Welcome to the Number Guessing Game with Leaderboard!

Enter your name: FrostByte

Select Difficulty Level:
1. Easy (1 to 50)
2. Medium (1 to 100)
3. Hard (1 to 500)
Enter your choice (1-3): 2

I've selected a number between 1 and 100.
You have 7 attempts. Good luck!
Attempt 1: Enter your guess: 50
Too high!
Attempt 2: Enter your guess: 25
Too high!
Attempt 3: Enter your guess: 15
Too high!
Attempt 4: Enter your guess: 10
Too high!
Attempt 5: Enter your guess: 5
Too low!
Attempt 6: Enter your guess: 7
Too low!
Attempt 7: Enter your guess: 8
? Correct! You guessed the number in 7 tries.

? Game Summary:
Player: FrostByte
Level: Medium
Number to guess: 8
Your final guess: 8
Total attempts used: 7
Result: Win ?

? Leaderboard (Top 3 Players):
1. smasher - 6 attempts
2. frostbyte - 7 attempts
```

```
erGuessingGame.java\bin NumberGuessingGame "
? Welcome to the Number Guessing Game with Leaderboard!

Enter your name: FrostByte

Select Difficulty Level:
1. Easy (1 to 50)
2. Medium (1 to 100)
3. Hard (1 to 500)
Enter your choice (1-3): 2

I've selected a number between 1 and 100.
You have 7 attempts. Good luck!
Attempt 1: Enter your guess: 50
Too high!
Attempt 2: Enter your guess: 25
Too high!
Attempt 3: Enter your guess: 15
Too high!
Attempt 4: Enter your guess: 10
Too high!
Attempt 5: Enter your guess: 5
Too low!
Attempt 6: Enter your guess: 7
Too low!
Attempt 7: Enter your guess: 8
? Correct! You guessed the number in 7 tries.

? Game Summary:
Player: FrostByte
Level: Medium
Number to guess: 8
Your final guess: 8
Total attempts used: 7
Result: Win ?

? Leaderboard (Top 3 Players):
1. smasher - 6 attempts
2. frostbyte - 7 attempts
```

10. Future Enhancements

10.1 GUI Integration

Replace the console interface with a graphical UI using JavaFX or Swing.

10.2 Timer Feature

Track and display the time taken to complete the game.

10.3 Multiplayer Mode

Allow multiple players to compete in the same session.

10.4 Cloud Leaderboard

Upload and compare scores online using a remote server.

10.5 Hint System

Add optional hints after a certain number of failed attempts.

11. Testing and Validation

11.1 Manual Testing

Each component was tested through direct interaction.

11.2 Test Scenarios

- Valid and invalid input
- Edge values for guessing
- File reading/writing errors

11.3 Edge Cases

- Non-numeric input
- Empty leaderboard file
- Duplicate scores

12. Challenges and Solutions

12.1 Input Handling

Challenge: Handling unexpected or non-numeric input. Solution: Implement try-catch blocks or input validation (future improvement).

12.2 File Errors

Challenge: File not found or write failures. Solution: Added error messages and fail-safe mechanisms.

12.3 Sorting Tie-breakers

Challenge: Multiple players with same score. Solution: Sort by order of appearance if scores are equal.

12.4 Data Persistence

Challenge: Ensuring leaderboard persists correctly between sessions. Solution: Use file I/O for saving and loading data.

13. Educational Value: This project is ideal for learners who want to transition from theoretical knowledge to real-world application. It encapsulates many key programming skills, from logic design and conditionals to file handling and user interaction. Through engaging gameplay and structured development, students can build confidence and proficiency in Java.

14. Conclusion:

The Number Guessing Game provides a solid foundation for understanding Java programming. It is simple enough for beginners yet contains enough depth to explore more advanced concepts such as sorting, file I/O, and object-oriented design. By combining fun gameplay with practical development skills, it successfully meets both educational and entertainment goals.

15. References:

- Java Documentation: https://docs.oracle.com/javase/8/docs/
- Stack Overflow (various threads)
- Oracle Java Tutorials
- W3Schools Java Tutorials