**Report: Rock-Paper-Scissors Game with Data Structures**

**Introduction:**

This report provides an overview of a Rock-Paper-Scissors game developed using Java. The project aims to showcase the use of data structures such as stacks, queues, and arrays for managing gameplay data, round history, and player interactions. The game offers various modes including player vs CPU, player vs player, and includes functionality to save game history and display top scores.

**Objective:**

The objective of the project is to develop a simple but functional Rock-Paper-Scissors game that:

* Supports single-player and two-player modes.
* Keeps track of game history and top scores.
* Uses multiple data structures to organize the game logic.

**Data Structures Used:**

**1. Stack:**

A stack is used to store the results of each game (either single-player or two-player). The stack ensures that results are stored in a last-in-first-out (LIFO) order, which facilitates easy retrieval and saving of game history.

**2. Linear Queue:**

A queue is used to handle the inputs during gameplay. Since the game involves alternating player actions, the queue allows storing inputs and processing them in a first-in-first-out (FIFO) order. The queue is used to store each player’s choice for each round and the CPU’s random choices during single-player mode.

**3. ArrayList:**

ArrayLists are used to dynamically manage game history data, allowing for flexibility in adding and retrieving historical data. They are used to store and read game history from files, as well as to sort game results.

**4. Array:**

The sort() method uses an array to store game results and sorts them by score using Bubble Sort.

**Game Modes:**

**1. Player vs CPU:**

The player competes against the computer, making three rounds of choices. The CPU’s choices are randomly generated, and the winner of each round is determined based on the classic rules of Rock-Paper-Scissors.

**2. Player vs Player:**

Two players can compete against each other. Both players make three rounds of choices, and the winner is determined based on the same rules.

**Game Flow and User Interaction:**

The user is presented with a menu to choose the game mode, view history, or exit the program. Depending on the selected mode, the user will be asked to enter their choices or the names of the players. The game proceeds through the rounds, with results being displayed at the end of each game mode. Game history is saved to text files, which are then read and displayed when the "History & Top Score" option is chosen.

**Features and Functionality:**

**Input Validation:** Ensures that the user enters valid choices (0 for Rock, 1 for Scissor, and 2 for Paper).

**Game Results:** Displays the winner of each round and updates the history with the scores.

**Saving Game History:** The game results are saved to text files (SinglePlayerHistory.txt and TwoPlayerHistory.txt), preserving the history between sessions.

**Sorting Top Scores:** The results are sorted by score, displaying the top scores for single-player and two-player modes.

**Console Clear:** The console is cleared between rounds for a smoother user experience (platform-dependent).

**Exit:** The game can be exited at any time, saving the current history.

**Conclusion:**

The Rock-Paper-Scissors game effectively utilizes basic data structures to manage game data, save game history, and display top scores. The project serves as an example of how simple games can be enhanced using fundamental computer science concepts such as stacks, queues, and arrays. The system is scalable and can be further extended to include more advanced features such as multiplayer support over a network, graphical user interfaces, or even a leaderboard.

**Developers:**

Syed Muhammad Abdul Moiz – 23FA-001-CS

Muhammad Taha Nadeem – 23FA-003-CS

Prince Irshad – 23FA-035-CS

Shahrif Rahman – 23FA-025-CS