PR2:

File Handling

```
import java.io.*;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws IOException {
        // Create a scanner to take input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt the user for input
        System.out.print("Enter a string to write to the file: ");
        String userInput = scanner.nextLine();

        // Writing user input to a custom file
        FileWriter writer = new FileWriter("myfile.txt");
        writer.write(userInput);
        writer.close();
        System.out.println("Data written to the file.");

        // Reading from the custom file
        FileReader reader = new FileReader("myfile.txt");
        int character;
        System.out.println("\nReading the file content:");
        while ((character = reader.read()) != -1) {
            System.out.print((char) character);
```

```
        }
        reader.close();
        System.out.println(); // For a newline after printing the content

        // Appending new data to the existing file
        appendDataToFile("myfile.txt");

        // Reading from the file again after appending
        reader = new FileReader("myfile.txt");
        System.out.println("\nReading the updated file content:");
        while ((character = reader.read()) != -1) {
            System.out.print((char) character);
        }
        reader.close();
    }

    // Method to append data to the file
    public static void appendDataToFile(String filePath) throws IOException {
        // Open the file in append mode (true means append)
        Scanner scanner = new Scanner(System.in);

        // Prompt the user for input
        System.out.print("Enter a string to write to the file: ");
        String userInput = scanner.nextLine();

        FileWriter writer = new FileWriter(filePath, true);
```

```
        // Write new data to the file
        writer.write(userInput);
        writer.close();

        // Notify the user
        System.out.println("\nData appended to the file.");
    }
}
```

File handling refers to the process of storing, reading, and manipulating data in files on a computer. Java provides several classes for file handling, such as File, FileReader, FileWriter, BufferedReader, BufferedWriter, and others, which allow users to perform operations on files efficiently.

In this specific program, we are focusing on:

- **Writing data to a file** using FileWriter
- **Reading data from a file** using FileReader
- **Appending data to an existing file** using FileWriter with append mode enabled

File Writing:

- **FileWriter** is used to write characters to a file. If the file doesn't exist, it will be created. By default, FileWriter overwrites any existing content in the file.
- To avoid overwriting and append new data, we use the constructor new FileWriter("filePath", true), where the second parameter true ensures that data is appended instead of overwritten.

File Reading:

- **FileReader** is used to read characters from a file. This allows the program to fetch and display the contents of a file line by line or character by character.
- In this program, the file is read using a while loop that reads each character using read(), and it continues until the end of the file (read() returns -1).

Appending Data to a File:

- To append data to an existing file, we open the file in **append mode** using new FileWriter("filePath", true). This ensures that instead of overwriting the file, the new data is added at the end of the existing content.