File Read &Print it as Output - Documentation

1. Title & Purpose

Project Name: File Read & Print It as Output in Java
Purpose: Reads a CSV file and prints each student record with detailed fields line by line.

2. Features

- Reads a CSV file (`Students.csv`).
- Splits each line by commas.
- Dynamically reads header columns.
- Prints each student's details with column names.
- Adds a 2-second delay between printing each student's record.

3. Requirements

- Java Development Kit (JDK) installed on your system.
- A CSV file named `Students.csv` placed in the same directory as the Java code.
- Basic knowledge of Java programming and command line usage.

4. Code Explanation

- **Key Classes/Methods:**
  - `BufferedReader`: Efficiently reads text from input stream (CSV file).
  - `FileReader`: Reads character files.
  - `split(",")`: Splits each line by commas into an array of Strings.
- **Logic Overview:**
  1. Open `Students.csv` using `BufferedReader`.
  2. Read the first line to extract column headers.
  3. For each subsequent line (student record):
     - Split line by commas.
     - Print each field with its corresponding header.
     - Pause for 2 seconds before reading the next student.

5. How to Run

1. Save the Java code in a file named `Mini_Project.java`.
2. Ensure the `Students.csv` file is in the same folder.
3. Open your command prompt or terminal.
4. Run the compiled program:

   ```
   java Mini_Project
   ```

6. Sample Input/Output

Sample `Students.csv` content:

```
RollNo,FirstName,MiddleName,LastName,Age,Department,Address
26, Suraj, Kumar, Paudel,20, Information Technology, Murgiya
```

Expected Output:

```
Student Number #1

RollNo: 101

FirstName: Suraj

MiddleName: Kumar

LastName: Paudel

Age: 20

Department: Computer

Address: Butwal

_____________________________

(wait 2 seconds)

Student Number #2

_____________________________
```

7. Possible Extensions (Optional)

- Add support for different delimiters (e.g., tabs, semicolons).
- Add error handling for missing or malformed data lines.
- Implement command line arguments to specify file path.
- Add writing updated data back to CSV.
- Allow user input to search/filter students.

8. Conclusion

This project demonstrates basic file handling in Java, including reading text files line by line, parsing CSV format, and printing structured data. It also introduces handling delays in output using `Thread.sleep()`. This forms a strong foundation for further projects involving file processing and user interaction.