PROJECT DOCUMENTATION

Project Title: Student Data Reader (CSV Based)

Course: OOP IN JAVA (DIPLOMA IN INFORMATION TECHNOLOGY)

Introduction

The "**Student Data Reader**" is a simple Java mini project designed to read and display student information stored in a **CSV** (**Comma Separated Values**) file. The program parses the CSV file, extracts the header and student data, and presents the information in a structured and user-friendly output. A delay mechanism has been introduced to improve readability of results while displaying student records one by one.

This project demonstrates the usage of Java File Handling, Exception Handling, and Basic Data Structures to process real-world data effectively.

System Analysis

Existing System

Traditionally, Student records are viewed in spreadsheets (Excel/CSV) or manually stored in files. This makes the data less interactive and requires manual filtering.

Proposed System

This project automates the process of reading and displaying student data from a CSV file with a simple Java program. It ensures formatted output, error handling, and real-time display with controlled delays.

Advantages

- Simple and lightweight application.
- Easy to modify for larger datasets.
- Reduces human error in manual reading.
- Provides structured, formatted, and clear results.

System Requirements

Hardware Requirements

1. Processor: Intel i3 or above
2. RAM: 2 GB minimum
3. Storage: 50 MB free disk space

Software Requirements

1. Operating System: Windows/Linux/MacOS
2. JDK (Java Development Kit) 8 or above
3. Any IDE (Eclipse, IntelliJ, NetBeans) or text editor
4. CSV file with student data (Students.csv)

System Design

Flow of the Program

1. Open and read the CSV file.
2. Store the header fields (column names).
3. Read each student record line by line.
4. Validate and match column names with record values.

5. Display results in a formatted way.
6. Handle malformed lines and exceptions.

Implementation:

Tools and Technologies Used

1. Java I/O (BufferedReader, FileReader)
2. Exception Handling (IOException, InterruptedException)
3. Multithreading (Thread.sleep() for output delay)

Code Explanation

1. FileReader & BufferedReader: Used to read CSV file line by line.
2. String.split(): Splits header and student data into arrays.
3. Validation: Ensures data length matches header length.
4. Output Formatting: Displays student number, name, roll, etc. in clear format.
5. Delay (Thread.sleep): Adds 2-second pause for readability.

Testing

Types of Testing Performed

Unit Testing: Verified each method and file reading logic.
Integration Testing: Checked CSV file reading with different datasets.
Error Handling Testing: Tested missing file, malformed lines, and mismatched column counts.

Sample Output

Student Number #1

RollNo: 1
FirstName: Suraj
MiddleName:
LastName: Paudel
Age: 16
Department: IT
Address: Murgiya

And Soon.....

Conclusion

This mini project successfully demonstrates how Java can be used to handle file input/output operations, process CSV data, and present it in a structured and user-friendly format.

Limitations

- Works only with properly formatted CSV files.
- Large files may take longer due to sequential reading.

References

Java Documentation (Oracle)
TutorialsPoint – Java I/O
GeeksforGeeks – File Handling in Java