CO225-Apr2025 : Software Construction

Lab 06 : Exception Handling

Question 01

Loading Student Engagement Data from a CSV File

You are building an application to analyze student engagement data from a CSV file (student_data.csv). Each line contains:

studentId,firstName,lastName,score,activeEngagementTime

The application must parse each line, construct a StudentProfile object, and validate the data.

This is what needs to be done :

- File I/O & Parsing
  - Attempt to open `student_data.csv`.
  - If the file is missing, catch `FileNotFoundException` and print:
    Error: student_data.csv not found. Place it in the application directory.
- Data Validation
  - Split each line on commas and verify there are exactly 5 fields. Throw `IllegalArgumentException` if invalid.
  - Validate that `score` and `activeEngagementTime` are integers. Catch `NumberFormatException` and report the malformed line.
- Finally Block
  - Ensure the `Scanner` (or stream) is closed in a `finally` block.
- Output
  - After processing, print:
    Loaded Students:
    < loaded students details >
    Please refer to the screenshot below(black one) to get a better idea.

Expected Code Files are as follows :

- StudentProfile.java

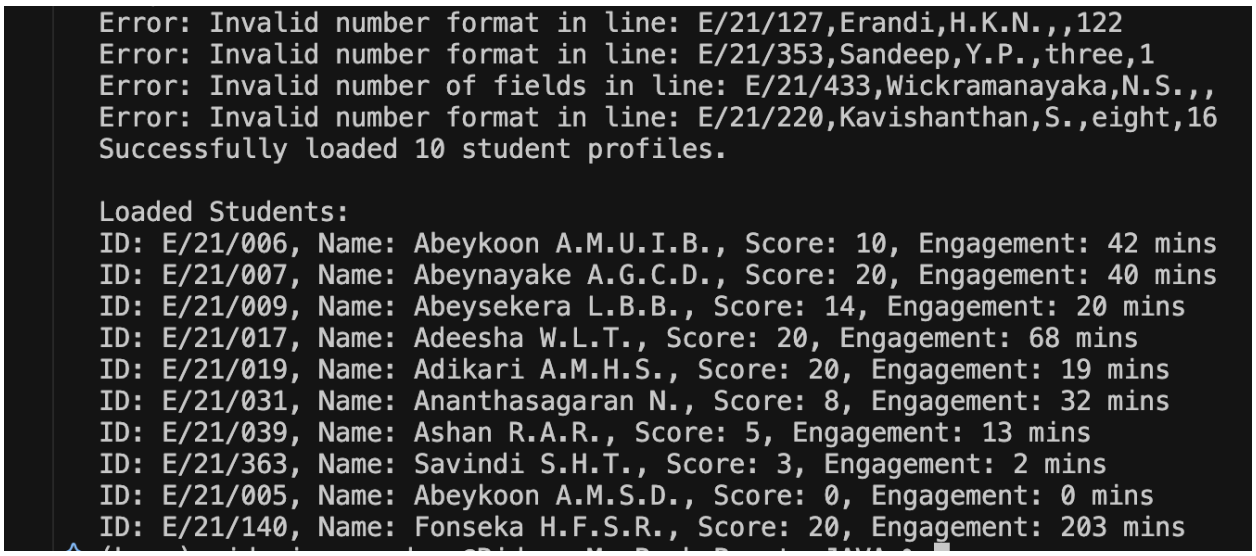
```
public class StudentProfile {
    private String studentId;
    private String firstName;
    private String lastName;
    private int score;
    private int activeEngagementTime;
    // Constructor, getters, and setters
}
```

- ProfileLoader.java
  Contains `main()` with the above logic.

student_data.csv is given in this Lab Folder.

studentId	firstName	lastName	score	activeEngagementTime
E/21/006	Abeykoon	A.M.U.I.B.	10	42
E/21/007	Abeynayake	A.G.C.D.	20	40
E/21/009	Abeysekera	L.B.B.	14	20
E/21/017	Adeesha	W.L.T.	20	68
E/21/019	Adikari	A.M.H.S.	20	19
E/21/031	Ananthasagaran	N.	8	32
E/21/039	Ashan	R.A.R.	5	13
E/21/127	Erandi	H.K.N.		122 // Missing score (invalid)
E/21/353	Sandeep	Y.P.	three	1 // Non-integer score (invalid)
E/21/363	Savindi	S.H.T.	3	2 // Valid
E/21/433	Wickramanayaka	N.S.		// Missing fields (invalid)
E/21/005	Abeykoon	A.M.S.D.	0	0 // Valid (zero engagement)
E/21/140	Fonseka	H.F.S.R.	20	203 // Valid (high engagement)
E/21/220	Kavishanthan	S.	eight	16 // Non-integer score (invalid)

Expected outcome as follows :



Question 02

Goliath National Bank (GNB) Transaction System

Goliath National Bank (GNB), the fictional mega-bank from HIMYM, is upgrading its internal transaction system. Barney Stinson, a proud employee of GNB, insists on making it “legendary,” but the dev team has some bugs to squash first.

As part of the engineering team, you’re tasked with building a robust bank account class for GNB that handles real-world exceptions gracefully, especially during deposits, withdrawals, and transfers between accounts.

To ensure financial correctness and clean failure handling, GNB requires that any invalid operation (such as overdrafts or negative amounts) throw a well-defined, meaningful exception and all errors must be logged with a time stamp in the final system report

Implement the following:

- Custom Exception Classes
  Create two custom exception classes:
  - InvalidAmountException: Thrown when a deposit or withdrawal amount is ≤ 0.
  - InsufficientFundsException: Thrown when a withdrawal or transfer exceeds the account balance

- BankAccount.java Class
  Design a class called BankAccount with the following features:

```
public class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double initialBalance);

    public void deposit(double amount) throws InvalidAmountException;

    public void withdraw(double amount) throws
InvalidAmountException, InsufficientFundsException;

    public void transferTo(BankAccount target, double amount)
        throws InvalidAmountException, InsufficientFundsException;

    public String toString(); // Returns account info and balance
}
```

- TransactionDemo.java
  In a main() method, simulate several transactions at GNB:

  BankAccount barneyAccount = new BankAccount("GNB123", 1000.00);
  BankAccount marshallAccount = new BankAccount("GNB456", 500.00);

- **Perform the following:**
  - **A successful deposit into Barney's account.**
  - **An invalid withdrawal** (a negative amount).
  - **A valid withdrawal.**
  - **An overdrawn transfer from Barney to Marshall.**
  - **A valid transfer from Barney to Marshall.**
- Each transaction should be inside a try-catch-finally block:
  - Catch and print the error message if an exception is thrown.
  - Use the finally block to print a time-stamped message like:
    [2025-05-29 12:00:03] Transaction attempt completed.
- Finally, print both account balances.