Task :3

Exception Handling Mechanism in Java

Date: 10-10-2025

Write a program to accept and print the student details during runtime. The details will include student id, student name, and department id. The program should raise an exception if user inputs incomplete or incorrect data. The entered value should meet the following conditions:

* Student name should be a string
* Student id should be an integer
* Department id should be an integer between 1 to 5

Aim:
To write a Java program that accepts and prints student details — student ID, student name, and department ID — during runtime.The program should validate user input and raise exceptions for incomplete or incorrect data.

Algorithm:

1. Start the program.
2. Accept inputs for student ID, name, and department ID from the user.
3. Validate inputs:
    - Student ID must be an integer.
    - Student name must be a string (no digits allowed).
    - Department ID must be an integer between 1 and 5.
4. If validation fails, throw an exception with an appropriate error message.
5. If validation passes, display all student details.
6. End the program.

Program:

```
import java.util.Scanner;

/**
 * Program to accept and display student details at runtime.
 * Validates user input and raises exceptions for invalid data.
 */
public class StudentDetails {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
```

```
        try {
            // Input Student ID
            System.out.print("Enter Student ID (integer): ");
            String sidInput = sc.nextLine();
            int studentId = Integer.parseInt(sidInput); // may throw NumberFormatException

            // Input Student Name
            System.out.print("Enter Student Name (string): ");
            String studentName = sc.nextLine();

            // Validate name (no digits)
            if (studentName.isEmpty() || !studentName.matches("[a-zA-Z ]+")) {
                throw new IllegalArgumentException("Invalid Name! Name should contain only
letters.");
            }

            // Input Department ID
            System.out.print("Enter Department ID (1 to 5): ");
            String deptInput = sc.nextLine();
            int deptId = Integer.parseInt(deptInput);

            // Validate Department ID range
            if (deptId < 1 || deptId > 5) {
                throw new IllegalArgumentException("Invalid Department ID! It must be between
1 and 5.");
            }

            // Display Student Details
            System.out.println("\n----- Student Details -----");
            System.out.println("Student ID: " + studentId);
            System.out.println("Student Name: " + studentName);
            System.out.println("Department ID: " + deptId);

        } catch (NumberFormatException e) {
            System.out.println("Error: Please enter numeric values for ID fields.");
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: Incomplete or incorrect data entered!");
        } finally {
            sc.close();
        }
    }
}
```

Sample Output 1 (Valid Input):

```
Enter Student ID (integer): 101
Enter Student Name (string): Kishor
Enter Department ID (1 to 5): 3

----- Student Details -----
Student ID: 101
Student Name: Kishor
Department ID: 3
```

Sample Output 2 (Invalid Name):

```
Enter Student ID (integer): 102
Enter Student Name (string): Kishor123
Enter Department ID (1 to 5): 2
Error: Invalid Name! Name should contain only letters.
```

Sample Output 3 (Invalid Department ID):

```
Enter Student ID (integer): 103
Enter Student Name (string): Arun
Enter Department ID (1 to 5): 8
Error: Invalid Department ID! It must be between 1 and 5.
```

Result:

Thus, the Java program successfully accepts student details at runtime, validates each input, and raises appropriate exceptions for incorrect or incomplete data.