**Exercise: Implementing Interfaces - Student and Games**

**Objective:**

Learn how to define and implement multiple interfaces in Java by modeling a scenario involving students who participate in games.

**Requirements:**

1. Create an interface named Student with the following method:
   * void study();   → represents the student studying.
2. Create another interface named Games with the following method:
   * void play();   → represents a game-playing activity.
3. Create a class named SchoolStudent that implements both Student and Games interfaces.
4. Implement the methods with meaningful output (e.g., printing what the student is doing).
5. In the main method:
   * Create an object of SchoolStudent.
   * Call both study() and play() methods using that object.

**Expected Learning Outcomes:**

* Understand how to define interfaces.
* Learn how to implement multiple interfaces in a single class.
* Practice calling interface methods using an implementing class.

**Exercise2 : Custom Exception with Student Marks Validation**

**Objective:**

Learn how to create and use a **custom checked exception** in Java to handle invalid input for a student management system.

**Requirements:**

1. Create a class named Student with the following fields:
   * String name
   * int marks
2. Create a custom checked exception class named InvalidMarksException that extends Exception.
3. In the Student class constructor or a method, validate that marks are in the range **0 to 100**.
   * If the marks are out of range, **throw InvalidMarksException** with a meaningful error message.
4. Create a StudentTest class with a main() method to:
   * Accept or assign values to at least one student.
   * Handle the exception using **try-catch**.
   * Display an appropriate message if the marks are invalid.

**Expected Learning Outcomes:**

* Define and use custom exception classes.
* Apply throw, throws, and try-catch correctly.
* Practice encapsulation and input validation in Java classes.