1. Basic Encapsulation: Employee Details

Problem:
Create a class Employee with private attributes name and salary. Provide public getter and setter methods to access and modify these attributes. Write a program to create an Employee object, set its details, and display them.

2. Encapsulation with Validation: Bank Account

Problem:
Create a BankAccount class with private attributes accountNumber and balance. Provide getter and setter methods for these attributes, but ensure that the balance cannot be negative. Write a program to test this validation.

3. Student Grades with Encapsulation

Problem:
Create a `Student` class with private attributes `name`, `rollNumber`, and `grade`. Add getter and setter methods with validation to ensure grades are between `A` and `F`.

4. Product Inventory System

Problem:
Create a `Product` class with private attributes `productName`, `price`, and `quantity`. Add methods to set and get these values. Include validation to ensure `price` and `quantity` are non-negative.

5. Encapsulation with Calculated Fields: Circle Area

Problem:
Create a `Circle` class with a private field `radius`. Provide a method to calculate the area of the circle. Ensure that `radius` cannot be negative.

6. Encapsulation for Library System

Problem:
Create a `Book` class with private attributes `title`, `author`, and `isBorrowed`. Provide getter and setter methods to manage the borrow status. Write a program to display book details and borrow/return operations.

7. Encapsulation with Read-Only Fields: Immutable User

Problem:
Create a `User` class with private final attributes `username` and `email`. Provide getter methods only. Write a program to demonstrate how the object is immutable.

8. Account Management with Deposit and Withdrawal

Problem:
Create a `BankAccount` class with private attributes `accountNumber` and `balance`. Add methods to deposit and withdraw money. Validate that withdrawal cannot exceed the current balance.

9. Encapsulation in Vehicle Management

Problem:
Create a `Car` class with private attributes `brand`, `model`, and `speed`. Provide setter methods to change speed, but ensure speed does not exceed a maximum limit (e.g., 200 km/h).

10. Employee Promotion System

Problem:
Create an `Employee` class with private fields `name`, `designation`, and `salary`. Provide a method `promote(String newDesignation, double increment)` that updates the employee's designation and salary.