Exercise 1: Basic Encapsulation (with Constructor & Method)

Create a class Student with the following:

- Private fields: name (String), age (int).
- Public getters and setters for both fields.
- Add a **constructor** to initialize name and age.
- Add a method displayInfo() to print student details.
- In the main method, create objects of Student using both constructor and setters, then print the details using the method.

Exercise 2: Validation with Encapsulation

Create a class BankAccount:

- Private fields: accountNumber (String), balance (double).
- Provide a **constructor** that initializes accountNumber and balance.
- Only allow positive balance in the setter; if negative, print "Invalid balance!".
- Write methods deposit(double amount) and withdraw(double amount) that update the balance safely.
- Add a method printStatement() to display account details.

Exercise 3: Encapsulation in Real-life Example

Create a class Employee:

- Private fields: name, salary.
- Provide a **constructor** to initialize fields (enforce salary > 10000).
- Add a method giveRaise(double amount) that increases salary only if amount > 0.
- Add a method showDetails() to print employee info.
- In the main method, test with valid and invalid salary and raise values.

Exercise 4: Read-Only Encapsulation

Create a class Book:

- Private fields: title, author.
- Provide a **constructor** to set values (no setters).
- Only provide getters.
- Add a method describe() that returns "Title by Author".
- In the main method, create multiple books and print details using the method.

Exercise 5: Encapsulation with Aggregation

Create two classes:

1. Address
    - Private fields: city, country.
    - Provide constructors, getters, and setters.
    - Add a method summary() to return "city, country".
2. Person
    - Private fields: name, Address.
    - Provide constructors, getters, and setters.
    - Add a method printProfile() to display person's name and address.

- In the main method, create a Person with an Address and print the profile.

Exercise 7: Product Inventory

Create a class Product:

- Private fields: id (String), name (String), price (double), quantity (int).
- Provide constructors, getters, and setters (price and quantity must be non-negative).
- Add methods:
    - increaseStock(int amount) – adds to quantity.
    - decreaseStock(int amount) – reduces quantity if enough stock is available.
    - printDetails() – prints product info.
- In the main method, simulate stock updates for multiple products.

Exercise 8: Car with Speed Control

Create a class Car:

- Private fields: brand (String), speed (int).
- Provide constructors, getters, and setters (speed cannot be negative).
- Add methods:
    - accelerate(int amount) – increases speed.
    - brake(int amount) – decreases speed but not below zero.
    - showStatus() – prints brand and current speed.
- In the main method, test accelerating and braking on different cars.

Exercise 9: Online Course System

Create two classes:

1. Course
    - Private fields: courseName, instructor, capacity.
    - Provide constructor and getters.
    - Add method showCourseInfo().
2. Student
    - Private fields: name, email.
    - Provide constructor, getters, and a method enroll(Course c) to print enrollment details.

- In the main method, create multiple students and enroll them into a course.

Exercise 10: Bank with Multiple Accounts

Create a class Bank:

- Private field: ArrayList<BankAccount> accounts.
- Provide constructors.
- Add methods:

- addAccount(BankAccount acc)
- findAccount(String accountNumber) – returns the matching account.
- printAllAccounts() – prints info for all accounts.

- Test by creating a bank, adding accounts, and performing deposits/withdrawals.

Exercise 11: Shopping Cart

Create a class ShoppingCart:

- Private field: ArrayList<Product> items.
- Provide constructor.
- Add methods:
  - addItem(Product p)
  - removeItem(String productId)
  - getTotalCost() – calculate and return total.
  - printCart() – show all items in the cart.
- In the main method, create products, add them to a shopping cart, and calculate total cost.

Exercise 12: University Management

Create three classes:

1. Professor
   - Private fields: name, department.
   - Provide constructors, getters, and a method showDetails().
2. Student
   - Private fields: name, rollNo.
   - Provide constructors, getters, and a method showDetails().
3. University
   - Private fields: ArrayList<Professor> professors, ArrayList<Student> students.
   - Provide methods to add professors and students.

    - Add a method printAllMembers().
- In the main method, simulate a university by adding professors and students, then display them.