Lab 6

Task 1: Employee Salary System (Accessors, Objects as Arguments & Return, Utility Methods)

Problem Statement:
Create a **class Employee** that:

- **Private Attributes:** name, salary
- **Public Methods:**
    - setEmployee(string name, double salary): Sets details.
    - getName(), getSalary(): Accessors.
    - giveRaise(double percentage): Increases salary.
    - compareSalary(Employee e): Returns the employee with a higher salary.
    - static double minWage(): Utility function returning a constant minimum wage.
- Implement accessors.
- Allow **salary comparison** between two employees:

Task 2: Complex Numbers (Objects as Arguments & Return, Cascaded Calls)

Problem Statement:
Create a **class Complex** that:

- **Attributes:** real, imag
- **Methods:**
    - setValues(double r, double i): Sets real and imaginary parts.
    - add(Complex c), subtract(Complex c): Returns a new Complex number.
    - multiply(Complex c): Returns a new Complex number.
    - display(): Prints the complex number.

Task 3: Library Book Management (Separate Declaration, Accessors, Objects as Return Types)

Problem Statement:
Create a **class Book** with:

- **Private Attributes:** title, author, price
- **Public Methods:**
    - setBook(string, string, double): Initializes details.
    - getTitle(), getAuthor(), getPrice(): Accessors.
    - applyDiscount(double percent): Reduces price and returns updated object.

- comparePrice(Book b): Returns the cheaper book.

Task 4: Car Fleet Management (Separate Declaration, Utility Methods, Cascaded Calls)

Problem Statement:
Create a **class Car** that:

- **Attributes:** brand, speed, fuel
- **Methods:**
  - setDetails(string, int, int): Initializes values.
  - accelerate(int): Increases speed, returns *this.
  - refuel(int): Adds fuel, returns *this.
  - display(): Prints details.
  - static int maxSpeed(): Utility method returning a car's max speed.

Task 5: E-Commerce Product System (Accessors, Utility Methods, Objects as Return Types)

Problem Statement:
Create a **class Product** with:

- **Attributes:** name, price, stock
- **Methods:**
  - setProduct(string, double, int): Sets values.
  - getPrice(), getStock(): Accessors.
  - buy(int quantity): Reduces stock, returns updated product.
  - applyDiscount(double percent): Reduces price.
  - static double taxRate(): Returns a fixed tax percentage.