PW 03 : Basic concept of OOP_part 02

Exercise 01: Polynomial and Monomial (Without Encapsulation)

In this version, all attributes are **public** (no encapsulation, no getters/setters).

1/ Define a class **Monome** with two attributes:

- **coefficient** (double): the coefficient of the monomial.
- **degree** (int): the degree of the monomial. Monomials are sequentially numbered.

2/ Complete the class with the following methods: - A constructor with two parameters. - A constructor with a single real parameter that creates a monomial with degree 0. - A method derive() that returns the derivative of the monomial. - A method evaluate(float x) that computes the value of the monomial for a given **x**. - A method display() that prints the monomial in the form cx^d. - A method product(Monome m) that returns the product of two monomials. - A method equals(Monome m) that checks whether two monomials are equal.

3/ Define a class **Polynome** of degree ≤ 2 (i.e., it has three monomials). Add the following methods:
a) A constructor that initializes the three monomials.

b) A method degree() that returns the degree of the polynomial (the maximum degree of its monomials).
c) Methods derive() and evaluate(float x), similar to those of the Monome class.

4/ Define a class **Application** that performs the following:

a) Create two polynomials p1 and p2 and display the message *"Identical polynomials"* if they are equal.
b) Compute p1(10).
c) Compute the derivative of p1 evaluated at x = 20.

Exercise 02: Encapsulation

Define the **BankAccount** class as follows:

1. **Instance Variables:**
   Bank accounts (BankAccount) are represented in the simplest possible way, that is, by two instance variables named respectively balance and id:

- **balance**: a double-precision number storing the account's balance.
- **id**: a string storing the owner's identity.

Constructors:
This class should have two constructors:

- **BankAccount(double initialBalance, String initialId):** a constructor with explicit initial values for the account balance and the owner's ID.
- **BankAccount():** a no-argument constructor that assigns a balance of 0 and sets the owner's name to "anonymous".

2. **Accessors and Mutators:**

- **String getId():** a getter method that returns a string representing the owner's identity.
- **void setId(String id):** a setter method allowing the owner's identity (string) to be changed.
- **double getBalance():** a getter method that returns the account balance.
- **void setBalance(double balance):** a setter method allowing the balance amount to be modified.

3. **Instance Methods:**

- **void withdraw(double amount):** debits the account by a strictly positive amount if the balance is at least equal to amount (does nothing otherwise).
- **void deposit(double amount):** credits the account by a strictly positive amount (does nothing if amount is not positive).
- **void transfer(double amount, BankAccount targetAccount):** transfers a strictly positive amount from the current account to targetAccount if the balance is at least equal to amount (does nothing otherwise).
- **void display():** displays a string describing the state of the BankAccount.

4. **Using the BankAccount class:**
   Declare a reference variable account of type BankAccount.
   Create/construct the corresponding bank account.
   Set Mohamed Kamel as the owner of the account, deposit 100 units, then another 50 units, withdraw 20 units to buy books, and compute the remaining balance.
   Assuming there exists another account referenced by the variable savingsAccount, transfer 100 units from account to savingsAccount.

Exercise 03: Scanner Class

Add the Scanner package in the main class:
import java.util.Scanner;

Input is performed using the following commands:
Scanner input = new Scanner(System.in); // Creation of the Scanner object
double ax = input.nextDouble(); // Reading a double
String ch = input.next(); // Reading a string
int age = input.nextInt(); // Reading an integer

1. Define a Date entity (class) including a method equals(Date d) that checks the equality between two dates.
2. Define a Message entity, giving the possibility to return or modify any field of the entity through methods defined in the class, and include a method for displaying its content.
3. Write a program that creates an array M of 10 messages and displays all the messages whose sending date is equal to a given date d.