Programming 2

Tutorial 2

Exercise 1 (required)

Create a class named **`Product`** with three attributes: `name`, `price`, and `discount`. The class should also include two methods:

- Calculate import tax (10% of the product price)
- Display information on the screen

The information displayed on the screen should include:

- ✓ Product name
- ✓ Unit price
- ✓ Discount
- ✓ Import tax

Product
+ name: String + price: double + discount: double
+ getImportTax(): double + displayInfo(): void + input(): void

Exercise 2 (required)

Write a program to create 2 products, with information entered from the keyboard, and then call the display method to output the information of the 2 created **`Product`** objects.

Create a class containing the `main()` method:

In the **`main()`** method, create 2 objects, `product1` and `product2`, from the `Product` class.

Call the **`input()`** method of the `product1` and `product2` objects to input data from the keyboard.

Call the **`displayInfo()`** method of the `product1` and `product2` objects to output the information of each object to the screen.

Exercise 3

Upgrade the `Product` class by adding `public` access modifiers for the `displayInfo()` method and `private` for the `getImportTax()` method. Additionally, add two constructors. The first constructor takes three parameters: `name`, `price`, and `discount`. The second constructor takes two parameters: `name` and `price` (implicitly assuming no discount).

Write a program to create two products, one with a discount and one without, then display the information of the two products on the screen.

Exercise 4

Extend the `Product` class from Exercises 1-3 by adding static attributes to track information across all `Product` instances.

1. Add the following static attributes to the `Product` class:
    - `productCount` - tracks the total number of Product objects created
    - `totalRevenue` - tracks the total potential revenue from all products (price × quantity for all products - discounted amount)
    - `totalDiscount` - tracks the total discounts of all products
2. Add a non-static attribute:
    - `quantity` - represents the quantity in stock for each product
3. Modify the constructors to:
    - Increment `productCount` each time a `Product` is created
    - Update `totalRevenue` and `totalDiscount` accordingly
4. Add the following static methods:
    - `getProductCount()` - returns the total number of products created
    - `getTotalRevenue()` - returns the total potential revenue (accounted for discounts)
    - `getAverageDiscount()` - returns the average discount percentage
    - `displayStatistics()` - displays all static information (total products, total revenue, average discount)
5. Add an instance method:
    - `updateQuantity(int newQuantity)` - updates the quantity and recalculates total revenue

Write a program called `ProductStatisticsDemo` that:

- Creates at least 3 `Product` objects with different prices, discounts, and quantities
- Displays each product's information
- Calls `Product.displayStatistics()` to show the statistics
- Updates the quantity of one product and displays the updated statistics

Sample Output:

```
Product: Laptop
Price: $1000.00
Discount: 10%
Quantity: 50
Import Tax: $100.00

Product: Mouse
Price: $25.00
Discount: 5%
Quantity: 200
Import Tax: $2.50

Product: Keyboard
Price: $75.00
Discount: 0%
Quantity: 100
Import Tax: $7.50

=== Product Statistics ===
Total Products Created: 3
Total Potential Revenue: $57250.00
Average Discount: 5.00%
```

Exercise 5 (easy, optional)

Extend the below `BankAccount` example:

```
public class BankAccount {
    double balance;
    int transactions;
    public BankAccount(double initial) {
        this.balance = initial;
        this.transactions = 1;
    }

    public void deposit(double amount) {
        balance += amount;
        transactions++;
    }

    public void withdraw(double amount) {
        balance -= amount;
        transactions++;
    }

    public void monthlyFee() {
        this.withdraw(10);
    }
}
```

- Add a method named `annualInterest` to add annual interest to the account's balance.

  (*) Hint: *You should add an attribute called `interestRate` for this feature and modify the constructor to initialize this attribute.*

- A bank account should have the name of the account holder. Implement this feature.

  (*) Hint: *You should add an attribute called `holderName` for this feature and modify the constructor to initialize this attribute.*

- Add a method called `toString()` which returns a string of this format:
  (if the balance is negative, put the – sign before the dollar sign)

  `Benson, $117.25`

  `Mathew, -$17.50`

- Implement a method called `transfer` to send money from one bank account to another.

There is a `$0.5` fee per transfer. Also, if the account does not have enough money to transfer and pay the fee, print out suitable error messages.

- Write a program called `BankAccountDemo` to test/demonstrate this `BankAccount` class.

Submission

Submit a **zip** file containing all Java programs to this tutorial's submission box in the course website on FIT LMS.