Self Practice - Day 06 - Class and Objects, Encapsulation

1. Question 1

Problem Statement:

Print the average of three numbers entered by user by creating a class named 'Average' having a method to calculate and print the average.

Analysis:

- Create a class named 'Average'.
- Create a method named 'calculate' inside the class.
- The method should accept three numbers as input.
- Calculate the average of the three numbers.
- Print the average.

Code

```java
class Average{
    public void calculate(int a, int b, int c){
        float avg = (a+b+c)/3;
        System.out.println("Average: "+avg);
    }
}

public class Question1{
    public static void main(String[] args){
        Average avg = new Average();
        avg.calculate(10, 20, 30);
    }
}
```

Output

```
Average: 20.0
```

---

2. Question 2

Problem Statement:

1. A class called Author, which models an author of a book, is designed as shown in the class diagram.
2. A class called Book, which models a book written by ONE author and composes an instance of Author as its instance variable, is designed as shown in the class diagram. Test the methods of Book and Author Class by creating instance of Book class.

Class Diagram

```
Book:
    - isbn : String
    - name : String
    - author : Author
    - price : double
```

```
    - qty : int

    + Book(isbn : String, name : String, author : Author, price : double)
    + Book(isbn : String, name : String, author : Author, price : double, qty : int)
    + getIsbn() : String
    + getAuthor() : Author
    + getName() : String
    + getPrice() : double
    + setPrice(price : double) : void
    + getQty() : int
    + setQty(qty : int) : void
    + toString() : String

Author:
    - name : String
    - email : String

    + Author(name : String, email : String)
    + getName() : String
    + setName(name : String) : void
    + getEmail() : String
    + setEmail(email : String) : void
    + toString() : String
```

Analysis:

- Create a class named 'Author'.
- Create a class named 'Book'.
- Create a constructor for both classes.
- Create methods to get and set the values.
- Create a method to display the values.

Code

```java
class Author{
    String name;
    String email;

    public Author(String name, String email){
        this.name = name;
        this.email = email;
    }

    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

    public String getEmail(){
        return email;
    }

    public void setEmail(String email){
        this.email = email;
    }

    public String toString(){
        return "Author: "+name+" ("+email+")";
    }
```

```
}

class Book{
    String isbn;
    String name;
    Author author;
    double price;
    int qty;

    public Book(String isbn, String name, Author author, double price){
        this.isbn = isbn;
        this.name = name;
        this.author = author;
        this.price = price;
    }

    public Book(String isbn, String name, Author author, double price, int qty){
        this.isbn = isbn;
        this.name = name;
        this.author = author;
        this.price = price;
        this.qty = qty;
    }

    public String getIsbn(){
        return isbn;
    }

    public Author getAuthor(){
        return author;
    }

    public String getName(){
        return name;
    }

    public double getPrice(){
        return price;
    }

    public void setPrice(double price){
        this.price = price;
    }

    public int getQty(){
        return qty;
    }

    public void setQty(int qty){
        this.qty = qty;
    }

    public String toString(){
        return name+" by "+author;
    }
}

public class Question2{
    public static void main(String[] args){
        Author author = new Author("Shabari", "2k21cse137@kiot.ac.in");
        Book book = new Book("123456", "The Logic God", author, 100.0, 10);
        System.out.println(book);
        System.out.println("ISBN: "+book.getIsbn());
        System.out.println("Name: "+book.getName());
        System.out.println("Price: "+book.getPrice());
        System.out.println("Qty: "+book.getQty());
```

```
    }
}
```

Output

```
The Logic God by Author: Shabari (2k21cse137@kiot.ac.in)
ISBN: 123456
Name: The Logic God
Price: 100.0
Qty: 10
```

---

3. Question 3

Problem Statement:

You are developing a simple program to manage product information for an online store. To represent each product, you want to create a Java class with default constructors that initialize the product details to default values.

Define a Java class named Product with the following specifications:

1. Private instance variables to store the product ID, name, price, and quantity.
2. Implement a default constructor that initializes the product details as follows:
    - Product ID: 0
    - Product name: "Unknown"
    - Price: 0.0
    - Quantity: 0
3. Implement another constructor that takes parameters for product ID, name, price, and quantity, and initializes the corresponding instance variables with the provided values.

Write the Java class Product with the given requirements and demonstrate its usage in the Main class by creating instances of the Product class using both default and parameterized constructors.

Analysis:

- Create a class named 'Product'.
- Create instance variables to store the product ID, name, price, and quantity.
- Create a default constructor to initialize the product details.
- Create another constructor to initialize the product details with the provided values.
- Create a method to display the product details.

Code

```java
class Product{
    private int productId;
    private String name;
    private double price;
    private int quantity;

    public Product(){
        productId = 0;
        name = "Unknown";
        price = 0.0;
        quantity = 0;
    }

    public Product(int productId, String name, double price, int quantity){
        this.productId = productId;
```

```
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public void display(){
        System.out.println("Product ID: "+productId);
        System.out.println("Name: "+name);
        System.out.println("Price: "+price);
        System.out.println("Quantity: "+quantity);
    }
}

public class Question3{
    public static void main(String[] args){
        Product product1 = new Product();
        Product product2 = new Product(123456, "The Logic God", 100.0, 10);
        product1.display();
        product2.display();
    }
}
```

Output

```
Product ID: 0
Name: Unknown
Price: 0.0
Quantity: 0
Product ID: 123456
Name: The Logic God
Price: 100.0
Quantity: 10
```

---

4. Question 4

Problem Statement:

Write a Java class called crop and help the farmer to separate the vegetables in his farm.The farm contains vegetables like Carrot,Brinjal,Potato .Seperate the vegetable and print the result in following format using constructor overloading C 15 P 25 B 30.

Analysis:

- Create a class named 'Crop'.
- Create constructors with different parameters.
- Print the result in the required format.

Code

```
class Crop{
    public Crop(){
        System.out.println("C 15 P 25 B 30");
    }

    public Crop(int carrot, int potato, int brinjal){
        System.out.println("C "+carrot+" P "+potato+" B "+brinjal);
    }
}
```

```
public class Question4{
    public static void main(String[] args){
        Crop crop1 = new Crop();
        Crop crop2 = new Crop(15, 25, 30);
    }
}
```

Output

```
C 15 P 25 B 30
C 15 P 25 B 30
```

---

5. Question 5

Problem Statement:

Write a program by creating an 'Employee' class having the following methods and print the final salary.

- 'getInfo()' which takes the salary, number of hours of work per day of employee as parameter
- 'AddSal()' which adds $10 to salary of the employee if it is less than 500.
- 'AddWork()' which adds $5 to salary of employee if the number of hours of work per day is more than 6 hours

Analysis:

- Create a class named 'Employee'.
- Create methods 'getInfo()', 'AddSal()', and 'AddWork()'.
- Calculate the final salary based on the given conditions.

Code

```
class Employee{
    double salary;
    int hours;

    public void getInfo(double salary, int hours){
        this.salary = salary;
        this.hours = hours;
    }

    public void AddSal(){
        if(salary<500){
            salary += 10;
        }
    }

    public void AddWork(){
        if(hours>6){
            salary += 5;
        }
    }

    public void display(){
        System.out.println("Final Salary: "+salary);
    }
}

public class Question5{
    public static void main(String[] args){
```

```
        Employee emp = new Employee();
        emp.getInfo(400, 8);
        emp.AddSal();
        emp.AddWork();
        emp.display();
    }
}
```

Output

```
Final Salary: 405.0
```

---

6. Question 6

Problem Statement:

The machine has two main components: A built-in cash register and several dispensers to hold and release the products.

Define class cashRegister in C# with the following descriptions :

- Private Members: cashOnHand of type integer
- Public Members: A default constructor cashRegister() sets the cash in the register to 100.
- A constructor cashRegister(int) sets the cash in the register to a specific amount.
- A function getCurrentBalance() which returns value of cashOnHand
- A function acceptAmount(int) to receive the amount deposited by the customer and update the amount in the register.

Analysis:

- Create a class named 'CashRegister'.
- Create instance variable 'cashOnHand'.
- Create default and parameterized constructors.
- Create methods to get the current balance and accept the amount.

Code

```
class CashRegister{
    private int cashOnHand;

    public CashRegister(){
        cashOnHand = 100;
    }

    public CashRegister(int cash){
        cashOnHand = cash;
    }

    public int getCurrentBalance(){
        return cashOnHand;
    }

    public void acceptAmount(int amount){
        cashOnHand += amount;
    }
}

public class Question6{
    public static void main(String[] args){
        CashRegister cr = new CashRegister();
```

```
        System.out.println("Current Balance: "+cr.getCurrentBalance());
        cr.acceptAmount(100);
        System.out.println("Current Balance: "+cr.getCurrentBalance());
    }
}
```

Output

```
Current Balance: 100
Current Balance: 200
```

---

7. Question 7

Problem Statement:

You are developing a simple program to manage a library's collection of books. Each book in the library has a unique identification number assigned to it. Additionally you want to define a constant variable to represent the maximum number of books that the library can hold.

Define a Java class named Library with the following specifications:

1. Implement a static variable named totalBooks to keep track of the total number of books in the library.
2. Define a constant variable named MAX_CAPACITY to represent the maximum number of books that the library can hold. Set its value to 1000.
3. Implement a method named addBook() that increments the totalBooks count each time a new book is added to the library.
4. Implement a method named getTotalBooks() that returns the current count of total books in the library.
5. Ensure that the MAX_CAPACITY variable cannot be modified after initialization.

Write the Java class Library with the given requirements and demonstrate its usage in the Main class by adding books to the library and retrieving the total count.

Analysis:

- Create a class named 'Library'.
- Create a static variable 'totalBooks'.
- Create a constant variable 'MAX_CAPACITY'.
- Create methods 'addBook()' and 'getTotalBooks()'.
- Ensure that the 'MAX_CAPACITY' variable cannot be modified after initialization.

Code

```
class Library{
    static int totalBooks;
    final static int MAX_CAPACITY = 1000;

    public void addBook(){
        totalBooks++;
    }

    public int getTotalBooks(){
        return totalBooks;
    }
}

public class Question7{
    public static void main(String[] args){
        Library library = new Library();
        library.addBook();
        library.addBook();
        library.addBook();
```

```
        System.out.println("Total Books: "+library.getTotalBooks());
    }
}
```

Output

```
Total Books: 3
```

---

8. Question 8

Problem Statement:

You are developing a simple banking system where you need to calculate the compound interest earned on a savings account. To accomplish this, you want to define a Java class with static methods for calculating compound interest.

Define a Java class named InterestCalculator with the following specifications:

1. Implement a static method named calculateCompoundInterest() that takes three parameters: the principal amount (initial investment), the annual interest rate (in percentage), and the number of years for which the interest is compounded. This method should return the total amount including the compound interest, calculated using the formula: Total Amount=Principal×(1+Annual Interest Rate100)Number of YearsTotal Amount=Principal×(1+100Annual Interest Rate)Number o Years
2. Implement another static method named calculateInterestOnly() that takes the same parameters as calculateCompoundInterest(), but it only calculates the compound interest earned over the given number of years. This method should return the compound interest amount, calculated as: Compound Interest=Principal×((1+Annual Interest Rate100)Number of Years−1)Compound Interest=Principal×((1+100Annual Interest Rate)Number of Years−1)

Write the Java class InterestCalculator with the given requirements and demonstrate its usage in the Main class by calculating compound interest and compound interest only for different scenarios

Analysis:

- Create a class named 'InterestCalculator'.
- Create static methods 'calculateCompoundInterest()' and 'calculateInterestOnly()'.
- Calculate the total amount and compound interest based on the given formula.

Code

```
class InterestCalculator{
    public static double calculateCompoundInterest(double principal, double rate, int years){
        return principal * Math.pow(1 + rate/100, years);
    }

    public static double calculateInterestOnly(double principal, double rate, int years){
        return principal * (Math.pow(1 + rate/100, years) - 1);
    }
}

public class Question8{
    public static void main(String[] args){
        System.out.println("Total Amount: "+InterestCalculator.calculateCompoundInterest(1000, 5, 2));
        System.out.println("Compound Interest: "+InterestCalculator.calculateInterestOnly(1000, 5, 2));
    }
}
```

Output

```
Total Amount: 1102.5
Compound Interest: 102.5
```

---

9. Question 9

Problem Statement:

You are tasked with developing a simple application to manage employee information for a company. To maintain data integrity and ensure security, you need to encapsulate the employee details within a Java class and provide controlled access using getter and setter methods.

Define a Java class named Employee with the following specifications: Private instance variables to store the employee's ID, name, age, and salary.

Public getter and setter methods for each instance variable to provide controlled access to the employee details. Ensure that the setter methods validate the input before assigning it to the instance variables:

- The employee ID must be a positive integer.
- The employee's name cannot be empty.
- The employee's age must be a positive integer.
- The employee's salary must be a non-negative value.

Implement a method named raiseSalary() that takes a percentage increase as input and raises the employee's salary accordingly. Implement a method named displayInfo() that displays all the employee details.

Write the Java class Employee with the given requirements and demonstrate its usage in the Main class by creating an employee, updating their details, giving them a salary raise, and displaying their information.

Analysis:

- Create a class named 'Employee'.
- Create instance variables to store the employee's ID, name, age, and salary.
- Create getter and setter methods for each instance variable.
- Validate the input before assigning it to the instance variables.
- Create a method 'raiseSalary()' to raise the employee's salary based on the percentage increase.
- Create a method 'displayInfo()' to display all the employee details.

Code

```java
class Employee1 {
    private int id;
    private String name;
    private int age;
    private double salary;

    public int getId(){
        return id;
    }

    public void setId(int id){
        if(id>0){
            this.id = id;
        }
    }

    public String getName(){
        return name;
    }

    public void setName(String name){
```

```
        if(!name.isEmpty()){
            this.name = name;
        }
    }

    public int getAge(){
        return age;
    }

    public void setAge(int age){
        if(age>0){
            this.age = age;
        }
    }

    public double getSalary(){
        return salary;
    }

    public void setSalary(double salary){
        if(salary>=0){
            this.salary = salary;
        }
    }

    public void raiseSalary(double percentage){
        salary += salary * percentage / 100;
    }

    public void displayInfo(){
        System.out.println("ID: "+id);
        System.out.println("Name: "+name);
        System.out.println("Age: "+age);
        System.out.println("Salary: "+salary);
    }
}

public class Question9{
    public static void main(String[] args){
        Employee1 emp = new Employee1();
        emp.setId(1);
        emp.setName("Shabari");
        emp.setAge(20);
        emp.setSalary(100000);
        emp.raiseSalary(10);
        emp.displayInfo();
    }
}
```

Output

```
ID: 1
Name: Shabari
Age: 20
Salary: 110000.0
```

11. Question 11

Problem Statement:

Write the Java implementation for a class named ‘TaxOnSalary’ to calculate tax on salary. The class TaxOnSalary is described as follows:

Attributes:

- salary: double // salary to calculate tax
- isPANsubmitted: boolean // PAN submission status

Methods:

The class supplies the operation(s) as per the following specification:

- A TaxOnSalary instance can be created either by supplying the value for the instance field isPANsubmitted OR without supplying value for any field. If TaxOnSalary instance is created by providing the value for isPANsubmitted then the value for salary is initialized with 1000.00.However it can be reinitialized through the methc inputSalary() [which is described below] . If TaxOnSalary instance is created without supplying value for any field, then value for salary and isPANsubmitted is by default initialized to 0.0 and false respectively.
- Accessor methods(s) are provided for every instance field.
- A method for computing the tax based on salary [caculateTax() : double] is supplied. The tax is calculated as per the rules shown below:
    - if salary < 180000 and isPANsubmitted = true, then tax payable is zero
    - if salary < 180000 and isPANsubmitted = false, then tax payable is 5% of the salary
    - if 180000 < salary < 500000, then tax payable is 10% of the salary
    - if 500000 < salary < 1000000, then tax payable is 20% of the salary
    - if 1000000 < salary, then tax payable is 30% of the salary A method named inputSalary() is supplied to read the value for the salary as an input from the user [consider reading this value from keyboard] and to assign the value to the corresponding instance variable salary. Write a Test class named TestTax.java which
- Creates two instances of tax1 and tax2 of the class TaxOnSalary with different initializations [see point (a) in the description of Methods].
- Takes salary as an input from the user [using keyboard] for both the instances tax1 and tax2.
- Calculate and display tax for both the instance tax1 and tax2

Analysis:

- Create a class named 'TaxOnSalary'.
- Create instance variables 'salary' and 'isPANsubmitted'.
- Create a constructor to initialize the instance variables.
- Create accessor methods for each instance variable.
- Create a method 'calculateTax()' to calculate the tax based on the salary.
- Create a method 'inputSalary()' to read the value for the salary from the user.
- Create a Test class to test the 'TaxOnSalary' class.

Code

```
import java.util.Scanner;

class TaxOnSalary{
    double salary;
    boolean isPANsubmitted;

    public TaxOnSalary(){
        salary = 0.0;
        isPANsubmitted = false;
    }

    public TaxOnSalary(boolean isPANsubmitted){
        salary = 1000.0;
        this.isPANsubmitted = isPANsubmitted;
    }

    public double getSalary(){
        return salary;
    }
```

```
    public boolean getIsPANsubmitted(){
        return isPANsubmitted;
    }

    public void inputSalary(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Salary: ");
        salary = sc.nextDouble();
    }

    public double calculateTax(){
        if(salary < 180000 && isPANsubmitted){
            return 0.0;
        }
        else if(salary < 180000 && !isPANsubmitted){
            return salary * 0.05;
        }
        else if(salary < 500000){
            return salary * 0.10;
        }
        else if(salary < 1000000){
            return salary * 0.20;
        }
        else{
            return salary * 0.30;
        }
    }
}

public class Question10{
    public static void main(String[] args){
        TaxOnSalary tax1 = new TaxOnSalary(true);
        tax1.inputSalary();
        System.out.println("Tax: "+tax1.calculateTax());

        TaxOnSalary tax2 = new TaxOnSalary(false);
        tax2.inputSalary();
        System.out.println("Tax: "+tax2.calculateTax());
    }
}
```

Output

```
Enter Salary: 20000
Tax: 0.0
Enter Salary: 100000
Tax: 5000.0
```