Day - 06 - Hands On - Shabari

1. QuestionNo1

Problem Statement:

Design the Circle class as given below and test the methods of the class by creating objects of type Circle in the driver class.

Analysis:

- The Circle class has the following attributes:
  - radius: a float representing the radius of the circle.
  - color: a string representing the color of the circle.
- The Circle class has the following methods:
  - Circle(): a constructor that creates a Circle instance with default radius and color.
  - Circle(radius: double): a constructor that creates a Circle instance with the given radius and default color.
  - Circle(radius: double, color: String): a constructor that creates a Circle instance with the given radius and color.
  - setRadius(radius: double): a method that sets the radius of the circle.
  - setColor(color: String): a method that sets the color of the circle.
  - getRadius(): a method that returns the radius of the circle.
  - getColor(): a method that returns the color of the circle.
- The driver class QuestionNo1 creates three Circle objects using the default constructor, the single parameterized constructor, and the multi-parameterized constructor. The driver class then prints
  - the color and radius of the first Circle object.
  - the color and radius of the second Circle object.
  - the color and radius of the third Circle object.

Code:

```
package com.hands_on;

class Circle{
    private double radius = 1.0;
    private String color = "red";

    Circle(){
        System.out.println("Calling default Constructor");
    }

    Circle(double radius){
        this.radius = radius;
        System.out.println("Calling single parameterized Constructor");
    }

    Circle(double radius, String color){
```

```
        this.radius = radius;
        this.color = color;
        System.out.println("Calling multi parameterized Constructor");
    }

    public void setRadius(double val){
        radius = val;
    }

    public void  setColor(String val){
        color = val;
    }

    public double getRadius(){
        return this.radius;
    }

    public String getColor(){
        return this.color;
    }

    public void display(){
        System.out.println("The object data is ");
        System.out.println("Radius = "+this.radius);
        System.out.println("Color = "+this.color);
    }
}

public class QuestionNo1 {
    public static void main(String[] args) {
        Circle a = new Circle();
        Circle b = new Circle(2.15);
        Circle c = new Circle(3.14, "Black");
        a.display();
        b.display();
        c.display();
    }
}
```

Output:

```
Calling default Constructor
Calling single parameterized Constructor
Calling multi parameterized Constructor
The object data is
Radius = 1.0
Color = red
The object data is
Radius = 2.15
Color = red
The object data is
Radius = 3.14
Color = Black
```

2. QuestionNo2

Problem Statement:

Write a program to print the area of a rectangle by creating a class named 'Area' having two methods. First method named as 'setDim' takes length and breadth of rectangle as parameters and the second method named as 'getArea' returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

Analysis:

- The Area class has the following attributes:
  - length: a float representing the length of the rectangle.
  - breadth: a float representing the breadth of the rectangle.
- The Area class has the following methods:
  - setDim(length: double, breadth: double): a method that sets the length and breadth of the rectangle.
  - getArea(): a method that returns the area of the rectangle.
- The driver class QuestionNo2 creates an Area object and calls the setDim method to set the length and breadth of the rectangle. The driver class then calls the getArea method to print the area of the rectangle.

Code:

```java
package com.hands_on;

import java.util.Scanner;

class Area{
    private double length;
    private double breadth;

    public void setDim(double length, double breadth){
        this.length = length;
        this.breadth = breadth;
    }

    public double getArea(){
        return length * breadth;
    }
}

public class QuestionNo2 {
    public static void main(String[] args) {
        Area a = new Area();
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the length of the rectangle: ");
        double length = sc.nextDouble();
        System.out.print("Enter the breadth of the rectangle: ");
        double breadth = sc.nextDouble();
        a.setDim(length, breadth);
        System.out.println("Area of the rectangle is: " + a.getArea());
```

```
    }
}
```

Output:

```
Enter the length of the rectangle: 20
Enter the breadth of the rectangle: 30
Area of the rectangle is: 600.0
```

---

3. QuestionNo3

Problem Statement:

Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.

Analysis:

- The Triangle class has the following attributes:
  - side1: a float representing the first side of the triangle.
  - side2: a float representing the second side of the triangle.
  - side3: a float representing the third side of the triangle.
- The Triangle class has the following methods:
  - getPerimeter(): a method that returns the perimeter of the triangle.
  - getArea(): a method that returns the area of the triangle.
- The driver class QuestionNo3 creates a Triangle object and calls the getPerimeter and getArea methods to print the perimeter and area of the triangle.

Code:

```
package com.hands_on;

import java.lang.Math;

class Triangle{
    private double side1 = 3;
    private double side2 = 4;
    private double side3 = 5;

    public double getPerimeter(){
        return side1 + side2 + side3;
    }

    public double getArea(){
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
}
```

```
public class QuestionNo3 {
    public static void main(String[] args) {
        Triangle a = new Triangle();
        System.out.println("Perimeter of the triangle is: " + a.getPerimeter());
        System.out.println("Area of the triangle is: " + a.getArea());
    }
}
```

Output:

```
Perimeter of the triangle is: 12.0
Area of the triangle is: 6.0
```

---

4. QuestionNo4

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:

- The Employee class has the following attributes:
  - salary: a float representing the salary of the employee.
  - hoursWorked: an integer representing the number of hours worked by the employee.
- The Employee class has the following methods:
  - getInfo(salary: double, hoursWorked: int): a method that sets the salary and hoursWorked of the employee.
  - AddSal(): a method that adds $10 to the salary of the employee if it is less than 500.
  - AddWork(): a method that adds $5 to the salary of the employee if the number of hours worked per day is more than 6 hours.
- The driver class QuestionNo4 creates an Employee object and calls the getInfo method to set the salary and hoursWorked of the employee. The driver class then calls the AddSal and AddWork methods to add $10 and 5 to the salary of the employee. The driver class then prints

Code:

```
package com.hands_on;

class Employee{
    private double salary;
    private int hoursWorked;

    public void getInfo(double salary, int hoursWorked){
        this.salary = salary;
```

```
        this.hoursWorked = hoursWorked;
    }

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

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

    public double getSalary(){
        return salary;
    }
}

public class QuestionNo4 {
    public static void main(String[] args) {
        Employee a = new Employee();
        a.getInfo(450, 7);
        a.AddSal();
        a.AddWork();
        System.out.println("Final Salary: " + a.getSalary());
    }
}
```

Output:

```
Final Salary: 465.0
```

---

5. QuestionNo5

Problem Statement:

Suppose you are designing a simple class to keep track of the number of students enrolled in a school. Each time a new student is enrolled, the total count of students should increase. Additionally, you want to define a constant variable for the maximum capacity of students that the school can accommodate.

Define a Java class named `School` with the following requirements:

1. Implement a static variable named `totalStudents` to keep track of the total number of students enrolled in the school.
2. Define a constant variable named `MAX_CAPACITY` to represent the maximum capacity of students that the school can accommodate. Set its value to 500.
3. Implement a method named `enrollStudent()` that increments the `totalStudents` count each time a new student is enrolled.
4. Implement a method named `getTotalStudents()` that returns the current count of total students.
5. Ensure that the `MAX_CAPACITY` variable cannot be modified after initialization.

Write the Java class `School` with the above requirements and demonstrate its usage in the `Main` class by enrolling students and retrieving the total count.

Analysis:

- The School class has the following attributes:
  - totalStudents: an integer representing the total number of students enrolled in the school.
  - MAX_CAPACITY: a constant integer representing the maximum capacity of students that the school can accommodate.
- The School class has the following methods:
  - enrollStudent(): a method that increments the totalStudents count each time a new student is enrolled.
  - getTotalStudents(): a method that returns the current count of total students.
- The driver class QuestionNo5 creates a School object and calls the enrollStudent method to increment the totalStudents count each time a new student is enrolled. The driver class then calls the getTotalStudents method to print the current count of total students.

Code:

```java
package com.hands_on;

class School{
    private static int totalStudents;
    private static final int MAX_CAPACITY = 500;

    public static void enrollStudent(){
        totalStudents++;
    }

    public static int getTotalStudents(){
        return totalStudents;
    }
}

public class QuestionNo5 {
    public static void main(String[] args) {
        School.enrollStudent();
        School.enrollStudent();
        School.enrollStudent();
        System.out.println("Total Students: " + School.getTotalStudents());
    }
}
```

Output:

```
Total Students: 3
```

---

6. QuestionNo6

Problem Statement:

You are developing a simple program to convert temperature measurements between Celsius and Fahrenheit scales. To achieve this, you want to define a Java class with static methods for temperature conversion.

Define a Java class named `TemperatureConverter` with the following specifications:

1. Implement a static method named `celsiusToFahrenheit()` that takes a temperature value in Celsius as input and returns the equivalent temperature in Fahrenheit using the formula: `Fahrenheit (°F) = (Temperature in degrees Celsius (°C) * 9/5) + 32`.
2. Implement another static method named `fahrenheitToCelsius()` that takes a temperature value in Fahrenheit as input and returns the equivalent temperature in Celsius using the formula: `Celsius (°C) = (°F - 32) × 5/9)`.

Write the Java class `TemperatureConverter` with the given requirements and demonstrate its usage in the Main class by converting temperatures between Celsius and Fahrenheit scales.

Analysis:

- The TemperatureConverter class has the following methods:
  - celsiusToFahrenheit(celsius: double): a static method that takes a temperature value in Celsius as input and returns the equivalent temperature in Fahrenheit.
  - fahrenheitToCelsius(fahrenheit: double): a static method that takes a temperature value in Fahrenheit as input and returns the equivalent temperature in Celsius.
- The driver class QuestionNo6 calls the celsiusToFahrenheit and fahrenheitToCelsius methods to convert temperatures between Celsius and Fahrenheit scales.

Code:

```
package com.hands_on;

import java.util.Scanner;

class Temperature{
    public static double celsiusToFahrenheit(double celsius){
        return (celsius * 9/5) + 32;
    }

    public static double fahrenheitToCelsius(double fahrenheit){
        return (fahrenheit - 32) * 5/9;
    }
}

public class QuestionNo6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the temperature in Celsius: ");
        double celsius = sc.nextDouble();
        System.out.println("Celsius to Fahrenheit: " + Temperature.celsiusToFahrenheit(celsius));
        System.out.println("Enter the temperature in Fahrenheit: ");
        double fahrenheit = sc.nextDouble();
        System.out.println("Fahrenheit to Celsius: " + Temperature.fahrenheitToCelsius(fahrenheit));
    }
}
```

Output:

```
Enter the temperature in Celsius:
37.5
Celsius to Fahrenheit: 99.5
Enter the temperature in Fahrenheit:
108.10
Fahrenheit to Celsius: 42.27777777777778
```

---

7. QuestionNo7

Problem Statement:

Write the java implementation for a class named Theatre to display the details of theater. The class Theatre is described as follows:

Attributes:

```
theatreId : int
theatreName : String
totalTheatreScreens
theatreLocation
```

Methods:

- A Theatre instance can be created by supplying the value for all the attributes of class.
- A method to display the details of a theatre

Write class TestTheatre.java which Create n instance of Theatre and display the n instance theatre details.

Analysis:

- The Theatre class has the following attributes:
  - theatreId: an integer representing the ID of the theater.
  - theatreName: a string representing the name of the theater.
  - totalTheatreScreens: an integer representing the total number of screens in the theater.
  - theatreLocation: a string representing the location of the theater.
- The Theatre class has the following methods:
  - Theatre(theatreId: int, theatreName: String, totalTheatreScreens: int, theatreLocation: String): a constructor that creates a Theater instance by supplying the value for all the attributes of the class.
  - display(): a method that displays the details of a theater.
- The driver class TestTheatre creates n instances of the Theatre class and displays the details of each theater instance.

Code:

```java
package com.hands_on;

class Theatre{
    private int theatreId;
```

```
    private String theatreName;
    private int totalTheatreScreens;
    private String theatreLocation;

    Theatre(int theatreId, String theatreName, int totalTheatreScreens, String theatreLocation){
        this.theatreId = theatreId;
        this.theatreName = theatreName;
        this.totalTheatreScreens = totalTheatreScreens;
        this.theatreLocation = theatreLocation;
    }

    public void display(){
        System.out.println("---------------------  Theater Details  ---------------------");
        System.out.println("Theater ID: " + theatreId);
        System.out.println("Theater Name: " + theatreName);
        System.out.println("Total Screens: " + totalTheatreScreens);
        System.out.println("Theater Location: " + theatreLocation);
        System.out.println("------------------------------------------------------------");
    }
}

public class TestTheatre {
    public static void main(String[] args) {
        Theatre t1 = new Theatre(1, "PVR", 5, "Hyderabad");
        Theatre t2 = new Theatre(2, "INOX", 3, "Bangalore");
        Theatre t3 = new Theatre(3, "Cinepolis", 4, "Chennai");
        t1.display();
        t2.display();
        t3.display();
    }
}
```

Output

```
---------------------  Theater Details  ---------------------
Theater ID: 1
Theater Name: PVR
Total Screens: 5
Theater Location: Hyderabad
------------------------------------------------------------
---------------------  Theater Details  ---------------------
Theater ID: 2
Theater Name: INOX
Total Screens: 3
Theater Location: Bangalore
------------------------------------------------------------
---------------------  Theater Details  ---------------------
Theater ID: 3
Theater Name: Cinepolis
Total Screens: 4
Theater Location: Chennai
------------------------------------------------------------
```

8. QuestionNo8

Problem Statement:

You are tasked with creating a Java class to represent a student in a school management system. To allow easy access to the student's details, you need to implement public instance variables in the class.

Define a Java class named Student with the following specifications:

1. Public instance variables to store the student's ID, name, age, and grade.
2. Implement a default constructor that initializes the student's details as follows:
    - Student ID: 0
    - Student name: "Unknown"
    - Age: 0
    - Grade: "Unknown"
3. Implement another constructor that takes parameters for student ID, name, age, and grade, and initializes the corresponding instance variables with the provided values.

Analysis:

- The Student class has the following attributes:
    - studentId: an integer representing the ID of the student.
    - studentName: a string representing the name of the student.
    - age: an integer representing the age of the student.
    - grade: a string representing the grade of the student.
- The Student class has the following constructors:
    - Student(): a default constructor that initializes the student's details with default values.
    - Student(studentId: int, studentName: String, age: int, grade: String): a constructor that initializes the student's details with the provided values.
- The driver class QuestionNo8 creates a Student object using the default constructor and another Student object using the parameterized constructor. The driver class then prints
    - the student details of the first Student object.
    - the student details of the second Student object.

Code:

```
package com.hands_on;

class Student{
    public int studentId;
    public String studentName;
    public int age;
    public String grade;

    Student(){
        studentId = 0;
        studentName = "Unknown";
        age = 0;
        grade = "Unknown";
```

```
    }

    Student(int studentId, String studentName, int age, String grade){
        this.studentId = studentId;
        this.studentName = studentName;
        this.age = age;
        this.grade = grade;
    }
}

public class QuestionNo8 {
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2 = new Student(101, "John", 15, "10th Grade");
        System.out.println("Student 1 Details:");
        System.out.println("Student ID: " + s1.studentId);
        System.out.println("Student Name: " + s1.studentName);
        System.out.println("Age: " + s1.age);
        System.out.println("Grade: " + s1.grade);
        System.out.println("Student 2 Details:");
        System.out.println("Student ID: " + s2.studentId);
        System.out.println("Student Name: " + s2.studentName);
        System.out.println("Age: " + s2.age);
        System.out.println("Grade: " + s2.grade);
    }
}
```

Output:

```
Student 1 Details:
Student ID: 0
Student Name: Unknown
Age: 0
Grade: Unknown
Student 2 Details:
Student ID: 101
Student Name: John
Age: 15
Grade: 10th Grade
```

---

9. QuestionNo9

Problem Statement:

Define a class in JAVA with following description:

Private Members

- A data member Flight number of type integer
- A data member Destination of type string
- A data member Distance of type float

- A data member Fuel of type float
- A member function CALFUEL() to calculate the value of Fuel as per the following criteria

Distance	Fuel
<=1000	500
more than 1000 and <=2000	1100
more than 2000	2200

Public Members

- A function FEEDINFO() to allow user to enter values for Flight Number, Destination, Distance & call function CALFUEL() to calculate the quantity of Fuel
- A function SHOWINFO() to allow user to view the content of all the data members

Analysis:

- The Flight class has the following attributes:
  - flightNumber: an integer representing the flight number.
  - destination: a string representing the destination of the flight.
  - distance: a float representing the distance of the flight.
  - fuel: a float representing the quantity of fuel required for the flight.
- The Flight class has the following methods:
  - CALFUEL(): a method that calculates the value of fuel based on the distance of the flight.
  - FEEDINFO(): a method that allows the user to enter values for flight number, destination, distance, and calls the CALFUEL() method to calculate the quantity of fuel.
  - SHOWINFO(): a method that allows the user to view the content of all the data members.
- The driver class QuestionNo9 creates a Flight object and calls the FEEDINFO() method to enter values for the flight details and calculate the quantity of fuel. The driver class then calls the SHOWINFO() method to display the flight details.

Code:

```
package com.hands_on;

import java.util.Scanner;

class Flight{
    private int flightNumber;
    private String destination;
    private float distance;
    private float fuel;

    private void CALFUEL(){
        if(distance <= 1000){
            fuel = 500;
        }else if(distance > 1000 && distance <= 2000){
            fuel = 1100;
        }else{
            fuel = 2200;
        }
```

```
    }

    public void FEEDINFO(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the Flight Number: ");
        flightNumber = sc.nextInt();
        System.out.print("Enter the Destination: ");
        destination = sc.next();
        System.out.print("Enter the Distance: ");
        distance = sc.nextFloat();
        CALFUEL();
    }

    public void SHOWINFO(){
        System.out.println("---------------------  Flight Details  ---------------------");
        System.out.println("Flight Number: " + flightNumber);
        System.out.println("Destination: " + destination);
        System.out.println("Distance: " + distance);
        System.out.println("Fuel: " + fuel);
        System.out.println("------------------------------------------------------------");
    }
}

public class QuestionNo9 {
    public static void main(String[] args) {
        Flight f = new Flight();
        f.FEEDINFO();
        f.SHOWINFO();
    }
}
```

Output:

```
Enter the Flight Number: 137
Enter the Destination: Salem
Enter the Distance: 1500
---------------------  Flight Details  ---------------------
Flight Number: 137
Destination: Salem
Distance: 1500.0
Fuel: 1100.0
------------------------------------------------------------
```

---

10. QuestionNo10

Problem Statement:

You are designing a simple banking system where you need to create a class to represent a bank account. To ensure data security and prevent unauthorized access to sensitive information, you want to encapsulate the account details by providing access only through getter and setter methods.

Define a Java class named `BankAccount` with the following specifications:

1. Private instance variables to store the account number, account holder's name, and account balance.
2. Public getter and setter methods for each instance variable to provide controlled access to the account details. Ensure that the setter methods validate the input before assigning it to the instance variables:
    - The account number must be a positive integer.
    - The account holder's name cannot be empty.
    - The account balance must be a non-negative value.
3. Implement a method named `deposit()` that takes an amount as input and adds it to the account balance.
4. Implement a method named `withdraw()` that takes an amount as input and subtracts it from the account balance, ensuring that the account balance never becomes negative.

Write the Java class BankAccount with the given requirements and demonstrate its usage in the Main class by creating an account, performing deposits and withdrawals, and retrieving account details.

Analysis:

- The BankAccount class has the following attributes:
    - accountNumber: an integer representing the account number.
    - accountHolderName: a string representing the account holder's name.
    - accountBalance: a double representing the account balance.
- The BankAccount class has the following methods:
    - getAccountNumber(): a method that returns the account number.
    - setAccountNumber(accountNumber: int): a method that sets the account number.
    - getAccountHolderName(): a method that returns the account holder's name.
    - setAccountHolderName(accountHolderName: String): a method that sets the account holder's name.
    - getAccountBalance(): a method that returns the account balance.
    - setAccountBalance(accountBalance: double): a method that sets the account balance.
    - deposit(amount: double): a method that takes an amount as input and adds it to the account balance.
    - withdraw(amount: double): a method that takes an amount as input and subtracts it from the account balance, ensuring that the account balance never becomes negative.
- The driver class QuestionNo10 creates a BankAccount object, performs deposits and withdrawals, and retrieves account details.

Code:

```java
package com.hands_on;

import java.util.Scanner;

class BankAccount{
    private int accountNumber;
    private String accountHolderName;
    private double accountBalance;

    public int getAccountNumber(){
        return accountNumber;
    }

    public void setAccountNumber(int accountNumber){
        if(accountNumber > 0){
            this.accountNumber = accountNumber;
```

```
        }
    }

    public String getAccountHolderName(){
        return accountHolderName;
    }

    public void setAccountHolderName(String accountHolderName){
        if(!accountHolderName.isEmpty()){
            this.accountHolderName = accountHolderName;
        }
    }

    public double getAccountBalance(){
        return accountBalance;
    }

    public void setAccountBalance(double accountBalance){
        if(accountBalance >= 0){
            this.accountBalance = accountBalance;
        }
    }

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

    public void withdraw(double amount){
        if(accountBalance - amount >= 0){
            accountBalance -= amount;
        }
    }

    public void display(){
        System.out.println("---------------------  Account Details  ----------------------");
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder Name: " + accountHolderName);
        System.out.println("Account Balance: " + accountBalance);
        System.out.println("------------------------------------------------------------");
    }
}

public class QuestionNo10 {
    public static void main(String[] args) {
        BankAccount b = new BankAccount();
        b.setAccountNumber(137);
        b.setAccountHolderName("Shabari");
        b.setAccountBalance(5000);
        b.deposit(1000);
        b.withdraw(2000);
        b.display();
    }
}
```

Output:

```
----------------------  Account Details  ----------------------
Account Number: 137
Account Holder Name: Shabari
Account Balance: 4000.0
------------------------------------------------------------
```