Hands On - Day 06 - Inheritance, Polymorphism, Abstraction

1. Question 1

Problem Statement:

Suppose you are working on a project for a car dealership that wants to keep track of their inventory. They have various types of vehicles in their inventory, including cars, trucks, and motorcycles. To manage this inventory efficiently, you are tasked with creating a program using Java that utilizes single inheritance.

Design a Java program to help the car dealership manage their inventory. Your program should include the following classes:

1. Vehicle class:
    - Attributes:
        - brand (String): to store the brand name of the vehicle.
        - year (int): to store the manufacturing year of the vehicle.
    - Methods:
        - displayInfo(): a method to display the brand name and manufacturing year of the vehicle.
2. Car class (subclass of Vehicle):
    - Additional Attributes:
        - model (String): to store the model of the car.
    - Additional Method:
        - displayCarInfo(): a method to display the brand name, model, and manufacturing year of the car.

Your program should allow the user to create an instance of a car, set its attributes (brand, year, model), and then display information about the car using both displayInfo() and displayCarInfo() methods.

Analysis

1. Create a Vehicle class with attributes brand and year.
2. Create a Car class that inherits from the Vehicle class and adds an attribute model.
3. Create a displayInfo() method in the Vehicle class to display brand and year.
4. Create a displayCarInfo() method in the Car class to display brand, model, and year.
5. Create an instance of the Car class, set its attributes, and display information using both methods.

Code:

```java
class Vehicle {
    String brand;
    int year;

    void displayInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Year: " + year);
    }
}

class Car extends Vehicle {
    String model;

    void displayCarInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
        System.out.println("Year: " + year);
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car();
        car.brand = "Toyota";
        car.year = 2021;
        car.model = "Corolla";
        car.displayInfo();
        car.displayCarInfo();
    }
}
```

Output:

```
Brand: Toyota
Year: 2021
Brand: Toyota
Model: Corolla
Year: 2021
```

---

2. Question 2

Problem Statement:

You're tasked with developing a program for a company to manage information about its employees and calculate their salaries. The company has different types of employees: fulltime employees, part-time employees, and contract employees. Each type of employee has specific attributes and salary calcu Design a Java program t assist the company in managing employee information and calculating their salaries. Your program should include the following classes:

1. Employee class:
   - Attributes:
     - employeeID (int): to store the employee ID.
     - name (String): to store the name of the employee.
   - Methods:
     - displayInfo(): a method to display the employee ID and name.
     - calculateSalary(): an abstract method to calculate the salary of the employee.
2. FullTimeEmployee class (subclass of Employee):
   - Additional Attribute:
     - salary (double): to store the monthly salary of the full-time employee.
   - Methods:
     - calculateSalary(): implements the method to calculate the salary based on the formula: salary = monthly salary.
3. PartTimeEmployee class (subclass of Employee):
   - Additional Attributes:
     - hourlyRate (double): to store the hourly rate of the part-time employee.
     - hoursWorked (int): to store the number of hours worked by the parttime employee.
   - Methods:
     - calculateSalary(): implements the method to calculate the salary based on the formula: salary = hourly rate * hours worked.
4. ContractEmployee class (subclass of Employee):
   - Additional Attribute:
     - contractDuration (int): to store the duration of the contract in months.
   - Methods:
     - calculateSalary(): implements the method to calculate the salary based on the formula: salary = contract duration * fixed monthly payment.

Your program should allow the user to create instances of full-time, part-time, and contract employees, set their attributes, and then calculate and display their salarie using the appropriate methods (displayInfo() and calculateSalary()).

Provide the Java implementation of the above scenario, ensuring proper hierarchical inheritance between the Employee, FullTimeEmployee, PartTimeEmployee, and ContractEmployee classes.

Analysis

1. Create an Employee class with attributes employeeID and name, and abstract method calculateSalary().
2. Create a FullTimeEmployee class that inherits from the Employee class and adds an attribute salary.
3. Create a PartTimeEmployee class that inherits from the Employee class and adds attributes hourlyRate and hoursWorked.
4. Create a ContractEmployee class that inherits from the Employee class and adds an attribute contractDuration.
5. Implement the calculateSalary() method in each subclass based on the specific salary calculation formula.
6. Create instances of each type of employee, set their attributes, and calculate and display their salaries.

Code:

```java
abstract class Employee {
    int employeeID;
    String name;

    void displayInfo() {
        System.out.println("Employee ID: " + employeeID);
        System.out.println("Name: " + name);
    }

    abstract void calculateSalary();
}

class FullTimeEmployee extends Employee {
    double salary;

    void calculateSalary() {
        System.out.println("Salary: " + salary);
    }
}

class PartTimeEmployee extends Employee {
    double hourlyRate;
    int hoursWorked;

    void calculateSalary() {
        System.out.println("Salary: " + hourlyRate * hoursWorked);
    }
}

class ContractEmployee extends Employee {
    int contractDuration;

    void calculateSalary() {
        System.out.println("Salary: " + contractDuration * 1000);
    }
}

public class Main {
    public static void main(String[] args) {
        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee();
        fullTimeEmployee.employeeID = 101;
        fullTimeEmployee.name = "Shabari";
        fullTimeEmployee.salary = 100000;
        fullTimeEmployee.displayInfo();
        fullTimeEmployee.calculateSalary();
        PartTimeEmployee partTimeEmployee = new PartTimeEmployee();
        partTimeEmployee.employeeID = 102;
        partTimeEmployee.name = "Sathyaram";
        partTimeEmployee.hourlyRate = 20;
        partTimeEmployee.hoursWorked = 40;
        partTimeEmployee.displayInfo();
        partTimeEmployee.calculateSalary();

        ContractEmployee contractEmployee = new ContractEmployee();
        contractEmployee.employeeID = 103;
        contractEmployee.name = "Naveth";
        contractEmployee.contractDuration = 6;
        contractEmployee.displayInfo();
        contractEmployee.calculateSalary();
    }
}
```

Output:

```
Employee ID: 101
Name: Shabari
Salary: 100000.0
Employee ID: 102
Name: Sathyaram
Salary: 800.0
Employee ID: 103
Name: Naveth
Salary: 6000
```

---

3. Question 3

Problem Statement:

You are building a restaurant ordering system. Create an abstract class called MenuItem with properties for Name (string) and Price (decimal). Implement an abstract method named Cook() that simulates cooking the menu item. Derive two classes from MenuItem called Burger and Pizza. Implement the Cook() method in both derive classes to display the name and cooking instructions for the menu item.

Write a program that creates instances of Burger and Pizza and calls their Cook() methods.

Analysis

1. Create an abstract class MenuItem with properties Name and Price, and an abstract method Cook().
2. Create a Burger class that inherits from MenuItem and implements the Cook() method to display cooking instructions for a burger.
3. Create a Pizza class that inherits from MenuItem and implements the Cook() method to display cooking instructions for a pizza.
4. Create instances of Burger and Pizza and call their Cook() methods.

Code:

```
abstract class MenuItem {
    String Name;
    double Price;

    abstract void Cook();
}

class Burger extends MenuItem {
    void Cook() {
        System.out.println("Cooking instructions for Burger:");
        System.out.println("1. Grill the patty.");
        System.out.println("2. Toast the bun.");
        System.out.println("3. Assemble the burger.");
    }
}

class Pizza extends MenuItem {
    void Cook() {
        System.out.println("Cooking instructions for Pizza:");
        System.out.println("1. Roll out the dough.");
        System.out.println("2. Add toppings.");
        System.out.println("3. Bake in the oven.");
    }
}

public class Main {
    public static void main(String[] args) {
        Burger burger = new Burger();
        burger.Name = "Cheeseburger";
        burger.Price = 10.99;
        burger.Cook();

        Pizza pizza = new Pizza();
        pizza.Name = "Margherita Pizza";
        pizza.Price = 12.99;
        pizza.Cook();
    }
}
```

Output:

```
Cooking instructions for Burger:
1. Grill the patty.
2. Toast the bun.
3. Assemble the burger.
Cooking instructions for Pizza:
1. Roll out the dough.
2. Add toppings.
3. Bake in the oven.
```

---

5. Question 5

Problem Statement:

You are building an e-commerce platform. Create an abstract class called Product with properties for Name (string) and Price (decimal). Implement an abstract metho named AddToCart() that adds the product to the user's shopping cart. Additionally, implement a non-abstract method named GetDiscountedPrice() that calculates an returns the discounted price for the product based on certain conditions. Derive two classes from Product called Book and Electronics. Implement the AddToCart() method in both derived classes to add the respective products to the shopping cart. Implement the GetDiscountedPrice() method to apply specific discount calculations for books and electronics. Write a program that creates instances of Book and Electronics and calls their AddToCart() and GetDiscountedPrice() methods.

Analysis

1. Create an abstract class Product with properties Name and Price, and abstract method AddToCart().
2. Implement a non-abstract method GetDiscountedPrice() in the Product class to calculate and return the discounted price.
3. Create a Book class that inherits from Product and implements the AddToCart() method to add a book to the shopping cart.
4. Create an Electronics class that inherits from Product and implements the AddToCart() method to add an electronic product to the shopping cart.
5. Implement the GetDiscountedPrice() method in both derived classes to apply specific discount calculations.
6. Create instances of Book and Electronics and call their AddToCart() and GetDiscountedPrice() methods.

Code:

```
abstract class Product {
    String Name;
    double Price;

    abstract void AddToCart();

    double GetDiscountedPrice() {
        return Price;
    }
}

class Book extends Product {
    void AddToCart() {
        System.out.println("Book added to cart: " + Name);
    }

    double GetDiscountedPrice() {
        return Price * 0.9; // 10% discount on books
    }
}

class Electronics extends Product {
    void AddToCart() {
        System.out.println("Electronics added to cart: " + Name);
    }

    double GetDiscountedPrice() {
        return Price * 0.8; // 20% discount on electronics
    }
}

public class Main {
    public static void main(String[] args) {
        Book book = new Book();
        book.Name = "Java Programming";
        book.Price = 29.99;
        book.AddToCart();
        System.out.println("Discounted Price: $" + book.GetDiscountedPrice());

        Electronics electronics = new Electronics();
        electronics.Name = "Smartphone";
        electronics.Price = 499.99;
        electronics.AddToCart();
        System.out.println("Discounted Price: $" + electronics.GetDiscountedPrice());
    }
}
```

Output:

```
Book added to cart: Java Programming
Discounted Price: $26.991
Electronics added to cart: Smartphone
Discounted Price: $399.992
```

---

6. Question 6

Problem Statement:

You are building a shape calculator application. Create an interface called IShape with the following methods: CalculateArea(): This method should calculate and retur the area of the shape. CalculatePerimeter(): This method should calculate and return the perimeter of the shape. Write two classes, Rectangle and Circle, that implement the IShape interface. In each class, implement the methods to calculate and return the area and perimeter of the shape. Write a program that creates instances of both Rectangle and Circle, sets their dimensions, and calls their respective methods to calculate the area and perimeter.

Analysis

1. Create an interface IShape with methods CalculateArea() and CalculatePerimeter().
2. Create a Rectangle class that implements the IShape interface and calculates the area and perimeter of a rectangle.
3. Create a Circle class that implements the IShape interface and calculates the area and perimeter of a circle.
4. Create instances of Rectangle and Circle, set their dimensions, and call the CalculateArea() and CalculatePerimeter() methods.

Code:

```java
interface IShape {
    double CalculateArea();
    double CalculatePerimeter();
}

class Rectangle implements IShape {
    double width;
    double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public double CalculateArea() {
        return width * height;
    }

    public double CalculatePerimeter() {
        return 2 * (width + height);
    }
}

class Circle implements IShape {
    double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double CalculateArea() {
        return Math.PI * radius * radius;
    }

    public double CalculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}

public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        System.out.println("Rectangle Area: " + rectangle.CalculateArea());
        System.out.println("Rectangle Perimeter: " + rectangle.CalculatePerimeter());

        Circle circle = new Circle(3);
        System.out.println("Circle Area: " + circle.CalculateArea());
        System.out.println("Circle Perimeter: " + circle.CalculatePerimeter());
    }
}
```

Output:

```
Rectangle Area: 50.0
Rectangle Perimeter: 30.0
Circle Area: 28.274333882308138
Circle Perimeter: 18.84955592153876
```

---

7. Question 7

Problem Statement:

Write a JAVA program that uses method overloading to perform different operations on a string. Implement a method called ProcessString that can perform different operations like converting the string to uppercase, reversing the string, and calculating the length of the string. The method should be overloaded to accept different numbers and types of parameters.

Analysis

1. Create a method ProcessString that overloads different operations on a string.
2. Implement the following operations:
    - Convert the string to uppercase.
    - Reverse the string.
    - Calculate the length of the string.
3. Overload the method to accept different numbers and types of parameters.

Code:

```
class StringOverloading{
    void ProcessString(String str){
        System.out.println("Original String: " + str);
    }

    void ProcessString(String str, int option){
        switch(option){
            case 1:
                System.out.println("Uppercase String: " + str.toUpperCase());
                break;
            case 2:
                StringBuilder sb = new StringBuilder(str);
                System.out.println("Reversed String: " + sb.reverse());
                break;
            case 3:
                System.out.println("Length of String: " + str.length());
                break;
            default:
                System.out.println("Invalid option");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        DemoOverloading demo = new DemoOverloading();
        demo.ProcessString("hello");
        demo.ProcessString("world", 1);
        demo.ProcessString("java", 2);
        demo.ProcessString("programming", 3);
    }
}
```

Output:

```
Original String: i am
Uppercase String: SHABARI
Reversed String: avaj
Length of String: 11
```

---

8. Question 8

Problem Statement:

Write a JAVA program that demonstrates method overloading by creating multiple methods with the same name but different numbers of parameters. Implement a method called CalculateArea that can calculate the area of a rectangle, a circle, and a triangle. The method should be overloaded to accept different numbers and type of parameters

Analysis

1. Create a method CalculateArea that overloads to calculate the area of a rectangle, circle, and triangle.
2. Implement the following operations:
    - Calculate the area of a rectangle using length and breadth.
    - Calculate the area of a circle using the radius.
    - Calculate the area of a triangle using the base and height.
3. Overload the method to accept different numbers and types of parameters.

Code:

```
class AreaOverloading {
    void calculateArea(double breadth, double height){
        System.out.println(breadth * height);
    }

    void calculateArea(double radius){
        System.out.println(Math.PI * radius * radius);
    }

    void  calculateArea(int base, int height){
        System.out.println(0.5 * base * height);
    }
}

public class Main {
    public static void main(String[] args) {
        AreaOverloading area = new AreaOverloading();
        area.calculateArea(5.5, 10.5);
        area.calculateArea(3.5);
        area.calculateArea(4, 6);
    }
}
```

Output:

```
57.75
38.48451000647496
12.0
```

9. Question 9

Problem Statement:

Create a JAVA program that uses method overloading to perform different operations on an array of integers. Implement a method called ProcessArray that can perfor different operations like finding the sum of the array elements, finding the maximum value in the array, and calculating the average of the array elements. The method should be overloaded to accept different numbers and types of parameters.

Analysis

1. Create a method ProcessArray that overloads different operations on an array of integers.
2. Implement the following operations:
    - Find the sum of the array elements.
    - Find the maximum value in the array.
    - Calculate the average of the array elements.
3. Overload the method to accept different numbers and types of parameters.

Code:

```java
class ArrayOverloading {
    void ProcessArray(int[] arr){
        int sum = 0;
        for(int i : arr){
            sum += i;
        }
        System.out.println("Sum of array elements: " + sum);
    }

    void ProcessArray(int[] arr, int option){
        switch(option){
            case 1:
                int max = arr[0];
                for(int i : arr){
                    if(i > max){
                        max = i;
                    }
                }
                System.out.println("Maximum value in the array: " + max);
                break;
            case 2:
                int sum = 0;
                for(int i : arr){
                    sum += i;
                }
                System.out.println("Average of array elements: " + (double)sum / arr.length);
                break;
            default:
                System.out.println("Invalid option");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        ArrayOverloading array = new ArrayOverloading();
        int[] arr = {1, 2, 3, 4, 5};
        array.ProcessArray(arr);
        array.ProcessArray(arr, 1);
        array.ProcessArray(arr, 2);
        array.ProcessArray(arr, 5);
    }
}
```

Output:

```
The sum of array elements is 15
Tha greatest element in the arrays is 5
The average elements of the array is 3.0
Invalid Option
```

---

10. Question 10

Problem Statement:

Create a base class called Shape with a virtual method named CalculateArea() that returns a double value. Derive two classes from Shape called Rectangle and Circle. Override the CalculateArea() method in both derived classes to calculate and return the area of a rectangle and a circle, respectively. Finally, write a program that creates instances of Rectangle and Circle and calls their CalculateArea() methods.

Analysis

1. Create a base class Shape with a virtual method CalculateArea() that returns a double value.
2. Derive two classes Rectangle and Circle from Shape.
3. Override the CalculateArea() method in both derived classes to calculate and return the area of a rectangle and a circle, respectively.
4. Create instances of Rectangle and Circle and call their CalculateArea() methods.

Code:

```
class Shape {
    public double calculateArea(){
        return 0;
    }
}

class Rectangle extends Shape {
    double breadth;
    double length;

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

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

class Circle extends Shape{
       double radius;

       public Circle(double radius){
           this.radius = radius;
        }

        public double calculateArea(){
            return Math.PI * radius * radius;
        }
}


public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        System.out.println("The Area of Rectangle : "+ rectangle.CalculateArea());

        Circle circle = new Circle(4);
        System.out.println("The Area of Circle: "+ circle.calculateArea());
    }
}
```

Output:

```
The Area of Rectangle : 50.0
The Area of Circle: 50.26548245743669
```

---

11. Question 11

Problem Statement:

You are tasked with designing a Java program to model different types of vehicles with various features.

Each vehicle can have common functionality such as starting and stopping, as well as specific features like electric or gaspowered engines. Design a program using interfaces to represent these features and implement classes for specific types of vehicles.

Define an interface named Vehicle with two methods:

1. start(): a method to start the vehicle.
2. stop(): a method to stop the vehicle.
3. Define two additional interfaces:
   - ElectricVehicle with a method charge() for vehicles with electric engines.
   - GasVehicle with a method refuel() for vehicles with gas engines.
4. Implement classes for specific types of vehicles:
   - Implement a class named ElectricCar for electric cars. This class should implement the Vehicle and ElectricVehicle interfaces and provide implementations fo all required methods.
   - Implement a class named GasMotorcycle for gas-powered motorcycles. This class should implement the Vehicle and GasVehicle interfaces and provide implementations for all required methods.
5. In the Main class, create instances of ElectricCar and GasMotorcycle, and test their functionality by calling their methods to start, stop, charge, and refuel.
6. Ensure that each class provides meaningful output messages when their methods are called to demonstrate the functionality of the vehicles.

Analysis

1. Create an interface Vehicle with methods start() and stop().
2. Create an interface ElectricVehicle with a method charge().
3. Create an interface GasVehicle with a method refuel().
4. Implement a class ElectricCar that implements the Vehicle and ElectricVehicle interfaces.
5. Implement a class GasMotorcycle that implements the Vehicle and GasVehicle interfaces.
6. Create instances of ElectricCar and GasMotorcycle and test their functionality by calling their methods.

Code:

```java
interface Vehicle {
    void start();
    void stop();
}

interface ElectricVehicle {
    void charge();
}

interface GasVehicle {
    void refuel();
}

class ElectricCar implements Vehicle, ElectricVehicle {
    @Override
    public void charge() {
        System.out.println("The Electric car is charging");
    }

    @Override
    public void start() {
        System.out.println("The Electric car is started");
    }

    @Override
    public void stop() {
        System.out.println("The Electric car is stopped");
    }
}


class GasMotorcycle implements Vehicle, GasVehicle {
    public void start() {
        System.out.println("Gas motorcycle started");
    }

    public void stop() {
        System.out.println("Gas motorcycle stopped");
    }

    public void refuel() {
        System.out.println("Gas motorcycle refueling");
    }
}

public class Main {
    public static void main(String[] args) {
        ElectricCar electricCar = new ElectricCar();
        GasMotorcycle gasMotorcycle = new GasMotorcycle();

        electricCar.start();
        electricCar.charge();
        electricCar.stop();

        gasMotorcycle.start();
        gasMotorcycle.refuel();
        gasMotorcycle.stop();
    }
}
```

Output:

```
The Electric car is started
The Electric car is charging
The Electric car is stopped
The Gas Vehicle is started
The Gas Vehicle is refueling
The Gas Vehicle is stopped
```
