Exp 4.Inheritance in java.

1. **Write a java program that creates a parent class called vehicles and 2 child classes car and motorcycle. The parent class should have attribute for color and speed and methods to display these attributes. The child class should inherit these attributes and methods also have their own unique attribute and methods.**

class Vehicle{

String color;

int speed;

Vehicle(String color, int speed){

this.color = color;

this.speed = speed;

}

void display(){

System.out.println("Color: " +color+", speed: "+speed);

}

}

class Car extends Vehicle{

String fuelType;

Car(String color, int speed, String fuelType){

super(color, speed);

this.fuelType = fuelType;

}

void displayFuelType(){

System.out.println("Fuel Type: "+ fuelType);

}

}

class Motorcycle extends Vehicle{

String type;

Motorcycle(String color, int speed, String type){

super(color, speed);

this.type = type;

}

void displayMType(){

System.out.println("Motorcycle Type: "+ type);;

}

}

public class InheritanceVehicle {

public static void main(String[] args) {

Vehicle v = new Vehicle("White", 70);

v.display();

Car c = new Car("Black", 90, "Petrol");

c.display();

c.displayFuelType();

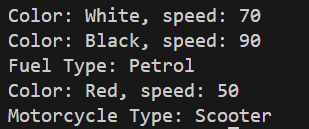
Motorcycle m = new Motorcycle("Red", 50,"Scooter" );

m.display();

m.displayMType();

}

}



1. **Create a parent class called shape with the following** 
   1. **A private string field for color.**
   2. **A public constructor that takes a string parameter for color.**
   3. **A public method called getColor that return the color.**

**Create 2 child classes first one is circle with a private double field for radius. A constructor that takes color and radius and a method called calculate area that return the area of circle.**

**Rectangle with a private double field for length and width, a constructor that takes color length and width and a method called calculate area that return the area of rectangle both child classes should inherit the color field and getColor method.**

class Shape{

private String color;

Shape(String color){

this.color = color;

}

String getColor(){

return color;

}

}

class Circle extends Shape{

private double radius;

Circle(String color, double radius){

super(color);

this.radius = radius;

}

double calculateArea(){

return 3.14\*radius\*radius;

}

}

class Rectangle extends Shape{

private double length;

private double width;

Rectangle(String color, double length, double width){

super(color);

this.length = length;

this.width = width;

}

double calculateArea(){

return length \* width;

}

}

public class InheritanceShape {

public static void main(String[] args) {

Circle c = new Circle("Red", 9);

System.out.println("Circle color: "+ c.getColor()+ " with area: "+c.calculateArea());

Rectangle r = new Rectangle("Yellow", 5, 8);

System.out.println("Rectangle color: "+ r.getColor()+ " with area: "+r.calculateArea());

}

}



**Overriding Questions**

1. **Question 1: Overriding with an Animal Hierarchy**  
   Create a superclass **Animal** with a method makeSound() that prints "Some generic sound".
   * Create subclasses **Dog**, **Cat**, and **Cow**, each overriding the makeSound() method to print their specific sounds ("Bark", "Meow", "Moo", respectively).  
     In the main method, create objects of the subclasses and call their makeSound() method to demonstrate overriding.

class Animal{

void makeSound(){

System.out.println("Some generic sound");

}

}

class Dog extends Animal{

void makeSound(){

System.out.println("Bark");

}

}

class Cat extends Animal{

void makeSound(){

System.out.println("Meow");

}

}

class Cow extends Animal{

void makeSound(){

System.out.println("Moo");

}

}

public class OverridingAnimal {

public static void main(String[] args) {

Dog d = new Dog();

d.makeSound();

Cat c = new Cat();

c.makeSound();

Cow co = new Cow();

co.makeSound();

}

}



1. **Question 2: Overriding with Super Keyword**
   * Write a Java program with a superclass Employee having a method displayDetails() that prints basic employee information.
   * Create a subclass Manager that overrides displayDetails() to include additional manager-specific information. Use the super keyword to call the superclass method from the overridden method..

class Employee{

void displayDetails(){

System.out.println("Name, Mobile Number, Project Name, Project Details");

}

}

class Manager extends Employee{

void displayDetails(){

super.displayDetails();

System.out.println("Manages, Work hours");

}

}

public class EmployeeOverride {

public static void main(String[] args) {

Employee e = new Employee();

System.out.println("Employeer");

e.displayDetails();

System.out.println("Manager");

Manager m = new Manager();

m.displayDetails();

}

}

