**Single Inheritance**

1. **Single Inheritance Basics**
   * **Question:** Write a Java program to demonstrate single inheritance. Create a base class Animal with a method makeSound(), and a derived class Dog that overrides makeSound() to print "Bark". Instantiate Dog and call makeSound().
2. **Single Inheritance with Constructor**
   * **Question:** Implement a Java program where a base class Vehicle has a constructor that initializes the make of the vehicle. The derived class Car should call the constructor of Vehicle using the super keyword and add an additional model property. Print the make and model of the car.

**Multilevel Inheritance**

1. **Multilevel Inheritance Basics**
   * **Question:** Create a base class Person with a method showName(). Derive a class Employee from Person with an additional method showJob(). Further derive a class Manager from Employee and add a method showDepartment(). Instantiate Manager and call all three methods.
2. **Multilevel Inheritance with Constructor**
   * **Question:** Implement a Java program where Shape is the base class with a constructor that initializes type. Circle is derived from Shape with an additional radius property, and ColoredCircle extends Circle with an additional color property. Use constructors to initialize all properties and display them.

**Hierarchical Inheritance**

1. **Hierarchical Inheritance Basics**
   * **Question:** Create a base class Animal with a method eat(). Derive two classes Cat and Bird from Animal, each with an additional method (meow() for Cat and fly() for Bird). Instantiate both Cat and Bird and call their respective methods along with the eat() method.
2. **Hierarchical Inheritance with Method Overriding**
   * **Question:** Implement a hierarchical inheritance where Vehicle is the base class with a method start(). Derive Bike and Car from Vehicle, each overriding start() to print different messages. Instantiate both Bike and Car and call start() on both.

**Super Keyword**

1. **Using Super to Access Superclass Methods**
   * **Question:** Write a Java program with a base class Parent having a method display() and a derived class Child that overrides display(). In the Child class, use the super keyword to call the display() method from Parent and add additional functionality. Instantiate Child and call display().
2. **Super Keyword in Constructor**
   * **Question:** Create a base class Animal with a constructor that initializes the name. Derive a class Dog from Animal and use the super keyword in the Dog constructor to call the Animal constructor. Print the name from the Dog class.
3. **Super Keyword with Variable Shadowing**
   * **Question:** Implement a Java program where a base class Base has a variable value. Derive a class Derived from Base, and in Derived, define a variable value that shadows the value in Base. Use the super keyword to access the value from Base and print both values.
4. **Super Keyword in Method Overriding**
   * **Question:** Write a program with a base class Calculator having a method add(int a, int b). Derive a class ScientificCalculator that overrides add() to perform additional operations. Use the super keyword in ScientificCalculator to call the add() method from Calculator and display both results.

**Single Inheritance**

**Single Inheritance Basics**

java

Copy code

// Base class

class Animal {

void makeSound() {

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

}

}

// Derived class

class Dog extends Animal {

@Override

void makeSound() {

System.out.println("Bark");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog();

dog.makeSound(); // Output: Bark

}

}

**Single Inheritance with Constructor**

java

Copy code

// Base class

class Vehicle {

String make;

Vehicle(String make) {

this.make = make;

}

}

// Derived class

class Car extends Vehicle {

String model;

Car(String make, String model) {

super(make);

this.model = model;

}

void displayInfo() {

System.out.println("Make: " + make);

System.out.println("Model: " + model);

}

}

public class Main {

public static void main(String[] args) {

Car car = new Car("Toyota", "Corolla");

car.displayInfo(); // Output: Make: Toyota Model: Corolla

}

}

**Multilevel Inheritance**

**Multilevel Inheritance Basics**

java

Copy code

// Base class

class Person {

void showName() {

System.out.println("Person");

}

}

// Intermediate class

class Employee extends Person {

void showJob() {

System.out.println("Employee");

}

}

// Derived class

class Manager extends Employee {

void showDepartment() {

System.out.println("Manager");

}

}

public class Main {

public static void main(String[] args) {

Manager manager = new Manager();

manager.showName(); // Output: Person

manager.showJob(); // Output: Employee

manager.showDepartment(); // Output: Manager

}

}

**Multilevel Inheritance with Constructor**

java

Copy code

// Base class

class Shape {

String type;

Shape(String type) {

this.type = type;

}

}

// Intermediate class

class Circle extends Shape {

double radius;

Circle(String type, double radius) {

super(type);

this.radius = radius;

}

}

// Derived class

class ColoredCircle extends Circle {

String color;

ColoredCircle(String type, double radius, String color) {

super(type, radius);

this.color = color;

}

void displayInfo() {

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

System.out.println("Radius: " + radius);

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

}

}

public class Main {

public static void main(String[] args) {

ColoredCircle coloredCircle = new ColoredCircle("Circle", 5.0, "Red");

coloredCircle.displayInfo(); // Output: Type: Circle Radius: 5.0 Color: Red

}

}

**Hierarchical Inheritance**

**Hierarchical Inheritance Basics**

java

Copy code

// Base class

class Animal {

void eat() {

System.out.println("Eating");

}

}

// Derived class 1

class Cat extends Animal {

void meow() {

System.out.println("Meow");

}

}

// Derived class 2

class Bird extends Animal {

void fly() {

System.out.println("Fly");

}

}

public class Main {

public static void main(String[] args) {

Cat cat = new Cat();

Bird bird = new Bird();

cat.eat(); // Output: Eating

cat.meow(); // Output: Meow

bird.eat(); // Output: Eating

bird.fly(); // Output: Fly

}

}

**Hierarchical Inheritance with Method Overriding**

java

Copy code

// Base class

class Vehicle {

void start() {

System.out.println("Vehicle starting");

}

}

// Derived class 1

class Bike extends Vehicle {

@Override

void start() {

System.out.println("Bike starting");

}

}

// Derived class 2

class Car extends Vehicle {

@Override

void start() {

System.out.println("Car starting");

}

}

public class Main {

public static void main(String[] args) {

Bike bike = new Bike();

Car car = new Car();

bike.start(); // Output: Bike starting

car.start(); // Output: Car starting

}

}

**Super Keyword**

**Using Super to Access Superclass Methods**

java

Copy code

// Base class

class Parent {

void display() {

System.out.println("Parent display");

}

}

// Derived class

class Child extends Parent {

@Override

void display() {

super.display(); // Calls Parent's display()

System.out.println("Child display");

}

}

public class Main {

public static void main(String[] args) {

Child child = new Child();

child.display(); // Output: Parent display Child display

}

}

**Super Keyword in Constructor**

java

Copy code

// Base class

class Animal {

String name;

Animal(String name) {

this.name = name;

}

}

// Derived class

class Dog extends Animal {

Dog(String name) {

super(name);

}

void displayName() {

System.out.println("Name: " + name);

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog("Buddy");

dog.displayName(); // Output: Name: Buddy

}

}

**Super Keyword with Variable Shadowing**

java

Copy code

// Base class

class Base {

int value = 10;

}

// Derived class

class Derived extends Base {

int value = 20;

void showValues() {

System.out.println("Base value: " + super.value);

System.out.println("Derived value: " + value);

}

}

public class Main {

public static void main(String[] args) {

Derived derived = new Derived();

derived.showValues(); // Output: Base value: 10 Derived value: 20

}

}

**Super Keyword in Method Overriding**

java

Copy code

// Base class

class Calculator {

int add(int a, int b) {

return a + b;

}

}

// Derived class

class ScientificCalculator extends Calculator {

@Override

int add(int a, int b) {

int sum = super.add(a, b); // Calls Calculator's add()

return sum + 10; // Adds extra functionality

}

}

public class Main {

public static void main(String[] args) {

ScientificCalculator sciCalc = new ScientificCalculator();

System.out.println("Result: " + sciCalc.add(5, 3)); // Output: Result: 18

}

}