**1.Simple Inheritance Example**

**PSEUDO CODE:**

\*Animal (Base Class)\*

PROCEDURE Animal(name)

SET name TO name

END PROCEDURE

PROCEDURE makeSound()

PRINT "The animal makes a sound"

END PROCEDURE

\*Dog (Derived Class)\*

PROCEDURE Dog(name)

INHERIT FROM Animal(name)

END PROCEDURE

PROCEDURE makeSound()

OVERRIDE

PRINT "The dog barks"

END PROCEDURE

\*Main (Test Class)\*

PROCEDURE Main()

DECLARE animal AS Animal

SET animal TO NEW Animal("Generic Animal")

CALL animal.makeSound()

DECLARE dog AS Dog

SET dog TO NEW Dog("Fido")

CALL dog.makeSound()

END PROCEDURE

**JAVA CODE:**

// Base class Animal

class Animal {

String name;

// Base class method

public void makeSound() {

System.out.println("The animal makes a sound");

}

}

// Derived class Dog that extends Animal

class Dog extends Animal {

// Overriding the makeSound method in the Dog class

@Override

public void makeSound() {

System.out.println("The dog barks");

}

}

// Main class to test inheritance

public class Main {

public static void main(String[] args) {

// Creating an object of Dog class

Dog dog = new Dog();

// Calling the makeSound method

dog.makeSound();

}

}

**2. Constructor Inheritance**

**Pseudo code:**

START

\*Person (Base Class)\*

PROCEDURE Person(name, age)

SET name, age

END PROCEDURE

PROCEDURE display()

PRINT name, age

END PROCEDURE

\*Student (Derived Class)\*

PROCEDURE Student(name, age, grade)

CALL super(name, age)

SET grade

END PROCEDURE

PROCEDURE display()

CALL super.display()

PRINT grade

END PROCEDURE

\*Main (Test Class)\*

PROCEDURE Main()

DECLARE student AS Student

SET student TO NEW Student("Alice", 20, "A")

CALL student.display()

END PROCEDURE

END

**Java code:**

// Base class Person

class Person {

String name;

int age;

// Constructor to initialize name and age

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Method to display person information

public void displayInfo() {

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

System.out.println("Age: " + age);

}

}

// Derived class Student that extends Person

class Student extends Person {

String grade;

// Constructor to initialize name, age, and grade

public Student(String name, int age, String grade) {

super(name, age); // Call the base class constructor

this.grade = grade;

}

// Override the displayInfo method to add grade information

@Override

public void displayInfo() {

super.displayInfo(); // Call the base class displayInfo method

System.out.println("Grade: " + grade);

}

}

// Main class to test constructor inheritance

public class Main {

public static void main(String[] args) {

// Create a Student object

Student student = new Student("John", 20, "A");

// Display student information

student.displayInfo();

}

}

**3.Multilevel Inheritance**

**Pseudo code:**

START

\*Vehicle (Base Class)\*

PROCEDURE Vehicle(speed, fuelType)

SET speed, fuelType

END PROCEDURE

PROCEDURE display()

PRINT speed, fuelType

END PROCEDURE

\*Car (Derived Class)\*

PROCEDURE Car(speed, fuelType, brand)

CALL super(speed, fuelType)

SET brand

END PROCEDURE

PROCEDURE display()

CALL super.display()

PRINT brand

END PROCEDURE

\*ElectricCar (Derived Class)\*

PROCEDURE ElectricCar(speed, fuelType, brand, batteryCapacity)

CALL super(speed, fuelType, brand)

SET batteryCapacity

END PROCEDURE

PROCEDURE display()

CALL super.display()

PRINT batteryCapacity

END PROCEDURE

\*Main (Test Class)\*

PROCEDURE Main()

DECLARE electricCar AS ElectricCar

SET electricCar TO NEW ElectricCar(120, "Electric", "Tesla", 100)

CALL electricCar.display()

END PROCEDURE

END

**Java code:**

// Base class Vehicle

class Vehicle {

int speed;

String fuelType;

// Constructor to initialize speed and fuelType

public Vehicle(int speed, String fuelType) {

this.speed = speed;

this.fuelType = fuelType;

}

// Method to display vehicle information

public void display() {

System.out.println("Speed: " + speed + " km/h");

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

}

}

// Derived class Car that extends Vehicle

class Car extends Vehicle {

String brand;

// Constructor to initialize speed, fuelType, and brand

public Car(int speed, String fuelType, String brand) {

super(speed, fuelType); // Call the Vehicle constructor

this.brand = brand;

}

// Method to display car information

@Override

public void display() {

super.display(); // Call Vehicle display method

System.out.println("Brand: " + brand);

}

}

// Derived class ElectricCar that extends Car

class ElectricCar extends Car {

int batteryCapacity;

// Constructor to initialize speed, fuelType, brand, and batteryCapacity

public ElectricCar(int speed, String fuelType, String brand, int batteryCapacity) {

super(speed, fuelType, brand); // Call Car constructor

this.batteryCapacity = batteryCapacity;

}

// Method to display electric car information

@Override

public void display() {

super.display(); // Call Car display method

System.out.println("Battery Capacity: " + batteryCapacity + " kWh");

}

}

// Main class to test multilevel inheritance

public class Main {

public static void main(String[] args) {

// Create an ElectricCar object

ElectricCar electricCar = new ElectricCar(120, "Electric", "Tesla", 100);

// Display all properties of the ElectricCar

electricCar.display();

}

}

**4. Method overriding in inheritance**

**Pseudo code:**

START

\*Shape (Base Class)\*

PROCEDURE Shape()

END PROCEDURE

PROCEDURE draw()

PRINT "Drawing a shape"

END PROCEDURE

\*Circle (Derived Class)\*

PROCEDURE Circle()

CALL super() // Call base class constructor

END PROCEDURE

PROCEDURE draw()

PRINT "Drawing a circle" // Override draw method

END PROCEDURE

\*Rectangle (Derived Class)\*

PROCEDURE Rectangle()

CALL super() // Call base class constructor

END PROCEDURE

PROCEDURE draw()

PRINT "Drawing a rectangle" // Override draw method

END PROCEDURE

\*Main (Test Class)\*

PROCEDURE Main()

DECLARE circle AS Circle // Declare Circle object

SET circle TO NEW Circle() // Initialize Circle object

CALL circle.draw() // Call draw method for Circle

DECLARE rectangle AS Rectangle // Declare Rectangle object

SET rectangle TO NEW Rectangle() // Initialize Rectangle object

CALL rectangle.draw() // Call draw method for Rectangle

END PROCEDURE

END

**Java code:**

// Base class Shape

class Shape {

// Method to draw a shape

public void draw() {

System.out.println("Drawing a shape");

}

}

// Derived class Circle that extends Shape

class Circle extends Shape {

// Overriding the draw method

@Override

public void draw() {

System.out.println("Drawing a circle");

}

}

// Derived class Rectangle that extends Shape

class Rectangle extends Shape {

// Overriding the draw method

@Override

public void draw() {

System.out.println("Drawing a rectangle");

}

}

// Main class to test method overriding

public class Main {

public static void main(String[] args) {

// Create a Circle object and call draw method

Shape circle = new Circle();

circle.draw(); // Output: "Drawing a circle"

// Create a Rectangle object and call draw method

Shape rectangle = new Rectangle();

rectangle.draw(); // Output: "Drawing a rectangle"

}

}

**5.Inheritance and access modifiers**

**Pseudo code:**

START

\*Employee (Base Class)\*

PRIVATE VARIABLE privateField

PROTECTED VARIABLE protectedField

PUBLIC VARIABLE publicField

PROCEDURE Employee(privateValue, protectedValue, publicValue)

SET privateField TO privateValue

SET protectedField TO protectedValue

SET publicField TO publicValue

END PROCEDURE

\*Manager (Derived Class)\*

PROCEDURE Manager()

CALL super(1, 2, 3) // Example values for fields

END PROCEDURE

PROCEDURE accessFields()

// Attempt to access fields

PRINT "Accessing fields from Manager:"

// PRINT privateField // This will cause a compilation error

PRINT "Protected Field: " + protectedField // Successful access

PRINT "Public Field: " + publicField // Successful access

END PROCEDURE

\*Main (Test Class)\*

PROCEDURE Main()

DECLARE manager AS Manager

SET manager TO NEW Manager()

CALL manager.accessFields() // Access fields from Manager

END PROCEDURE

END

**Java code:**

// Base class Employee

class Employee {

private int privateField; // Private field

protected int protectedField; // Protected field

public int publicField; // Public field

// Constructor for Employee

public Employee(int privateValue, int protectedValue, int publicValue) {

this.privateField = privateValue;

this.protectedField = protectedValue;

this.publicField = publicValue;

}

}

// Derived class Manager that extends Employee

class Manager extends Employee {

// Constructor for Manager

public Manager() {

super(1, 2, 3); // Example values for fields

}

// Method to access fields

public void accessFields() {

System.out.println("Accessing fields from Manager:");

// System.out.println("Private Field: " + privateField); // Compilation error

System.out.println("Protected Field: " + protectedField); // Successful access

System.out.println("Public Field: " + publicField); // Successful access

}

}

// Main class to test access modifiers

public class Main {

public static void main(String[] args) {

// Create a Manager object

Manager manager = new Manager();

// Access fields from Manager

manager.accessFields();

}

}