**Task 1 –**

**Algorithm**1. Create a base class Employee with attributes: name, employeeId, salary.

2. Define a method getDetails() in Employee that returns basic info.

3. Create a subclass Manager that adds department attribute.

4. Override getDetails() in Manager to include department.

5. Create a subclass Developer that adds programmingLanguage attribute.

6. Override getDetails() in Developer to include programmingLanguage.

7. In the main method, create and test all three types of objects.

**Psudocode**

CLASS Employee

ATTRIBUTES: name, employeeId, salary

METHOD getDetails() → return name, id, salary

SUBCLASS Manager extends Employee

ATTRIBUTE: department

OVERRIDE getDetails() → include department

SUBCLASS Developer extends Employee

ATTRIBUTE: programmingLanguage

OVERRIDE getDetails() → include programmingLanguage

MAIN METHOD

CREATE Manager and Developer objects

CALL getDetails() on each

**Code(java)**class Employee {

    String name;

    String employeeId;

    double salary;

    public Employee(String name, String employeeId, double salary) {

        this.name = name;

        this.employeeId = employeeId;

        this.salary = salary;

    }

    public String getDetails() {

        return "Name: " + name + ", ID: " + employeeId + ", Salary: " + salary;

    }

}

class Manager extends Employee {

    String department;

    public Manager(String name, String employeeId, double salary, String department) {

        super(name, employeeId, salary);

        this.department = department;

    }

    @Override

    public String getDetails() {

        return super.getDetails() + ", Department: " + department;

    }

}

class Developer extends Employee {

    String programmingLanguage;

    public Developer(String name, String employeeId, double salary, String programmingLanguage) {

        super(name, employeeId, salary);

        this.programmingLanguage = programmingLanguage;

    }

    @Override

    public String getDetails() {

        return super.getDetails() + ", Programming Language: " + programmingLanguage;

    }

}

public class Company {

    public static void main(String[] args) {

        Manager m = new Manager("Priya", "EMP001", 80000, "Finance");

        Developer d = new Developer("Karan", "EMP002", 75000, "Java");

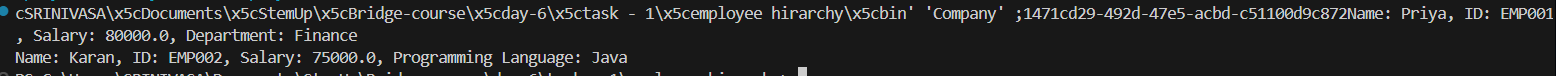
        System.out.println(m.getDetails());

        System.out.println(d.getDetails());

    }

}

**Output (test case 1)**

**Observations**

- Demonstrates inheritance and method overriding in Java.

- Each subclass extends the base behavior and adds specific data.

- getDetails() is reused and extended, showing polymorphism.

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)  
class Animal {**

**public void makeSound() {**

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

**}**

**}**

**class Dog extends Animal {**

**@Override**

**public void makeSound() {**

**System.out.println("Woof!");**

**}**

**}**

**class Cat extends Animal {**

**@Override**

**public void makeSound() {**

**System.out.println("Meow!");**

**}**

**}**

**public class AnimalKingdom {**

**public static void main(String[] args) {**

**Dog dog = new Dog();**

**Cat cat = new Cat();**

**dog.makeSound();**

**cat.makeSound();**

**}**

**}**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**

**Task 1 –**

**Algorithm**

**Psudocode**

**Code(java)**

**Output (test case 1)**

**Output (test case 2)**

**Output (test case 3)**

**Observations**