# Assignment - 7

## Object-Oriented Programming in Java

**Name:** Kamithkar Vinod
**Course:** PG DAC AUGUST 2025
**Form No:** 250500480
**Date:** 18-09-2025

---

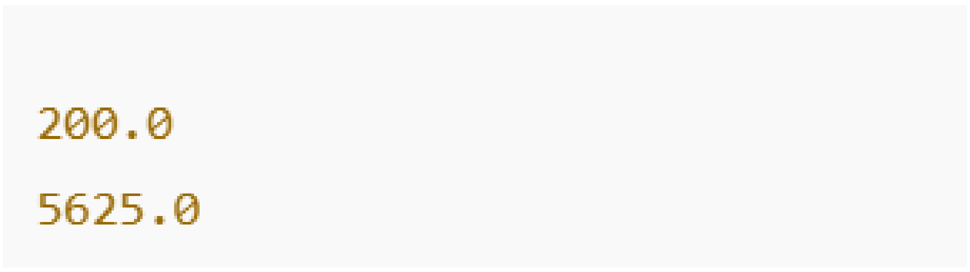
## Problem 1: Inheritance

**Task:** Create class Box and Box3d. Box3d is extended class of Box. The above two classes going to fulfill the following requirement. Include constructor, set value of length, breadth, height. Find out area and volume.

**Code:** —

```java
package Box;

public class Box {
    private double length;
    private double breadth;

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

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

}
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

package Box;

public class Box3d extends Box {

    private double height;

    public Box3d(double length, double breadth, double height) {
        super(length, breadth);
        this.height = height;
    }

    public double volume() {
        return super.area() * height;
    }
}

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

package Box;

public class Main {

    public static void main(String[] args) {
        Box b1 = new Box(10, 20);

        Box3d b2 = new Box3d(15, 15, 25);

        System.out.println(b1.area());

        System.out.println(b2.volume());

    }

}
```

**Output:** —



```
200.0
5625.0
```

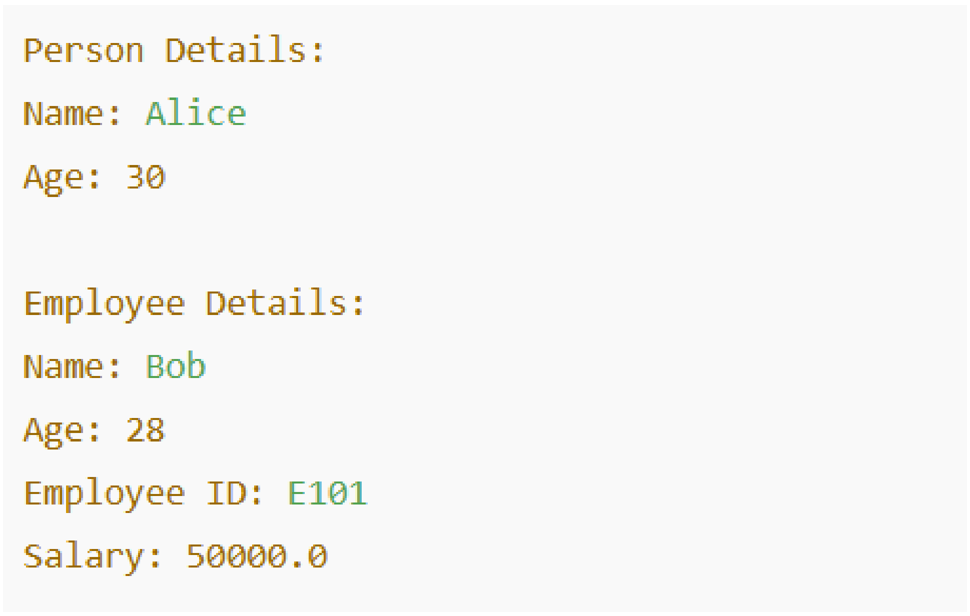
## Problem 2: Inheritance, Overloading, Overriding

**Task:** Define a base class Person and a derived class employee with single inheritance. Define SetData() member functions in each of the class with different signatures to set the data members and demonstrate overloading of member functions. Define GetData() member functions in each of the class with same signatures to display data and demonstrate overriding of member functions.

**Code:** —

```java
// Base class Person
class Person {
    protected String name;
    protected int age;

    // Overloaded SetData() method
    public void setData(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // GetData() method (will be overridden)
    public void getData() {
        System.out.println("Person Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

// Derived class Employee
class Employee extends Person {
    private String employeeId;
    private double salary;

    // Overloaded SetData() method (different signature than in
       Person)
    public void setData(String name, int age, String employeeId,
       double salary) {
        // Call parent setData for common attributes
        super.setData(name, age);
        this.employeeId = employeeId;
        this.salary = salary;
    }

    // Overriding GetData() method
    @Override
    public void getData() {
        System.out.println("Employee Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Employee ID: " + employeeId);
        System.out.println("Salary: " + salary);
    }
}

// Main class
public class OverloadingOverridingExample {
    public static void main(String[] args) {
        // Base class object
        Person p = new Person();
        p.setData("Alice", 30);
        p.getData();  // Calls Person version of getData()

        System.out.println();

        // Derived class object
        Employee e = new Employee();
        e.setData("Bob", 28, "E101", 50000.0); // Overloaded method
        e.getData();  // Calls Employee version (overridden)
    }
}
```

**Output:** —



```
Person Details:
Name: Alice
Age: 30

Employee Details:
Name: Bob
Age: 28
Employee ID: E101
Salary: 50000.0
```

## Problem 3: Multi Level Inheritance

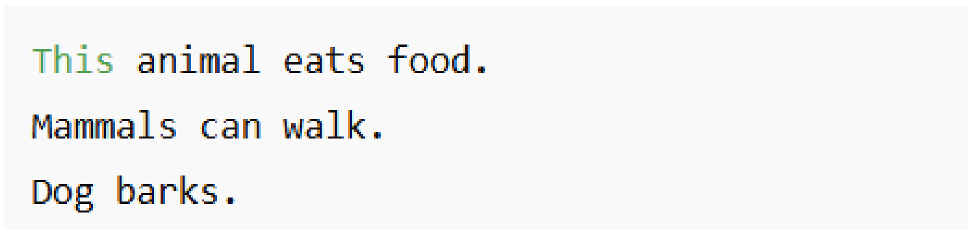
**Task:** Write a program to give example for multilevel inheritance in Java.

**Code:** —

```java
// Example of Multilevel Inheritance in Java

// Base class (Grandparent)
class Animal {
    void eat() {
        System.out.println("This animal eats food.");
    }
}

// Derived class (Parent)
class Mammal extends Animal {
    void walk() {
        System.out.println("Mammals can walk.");
    }
}

```

```java
// Derived class (Child)
class Dog extends Mammal {
    void bark() {
        System.out.println("Dog barks.");
    }
}

// Main class
public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        // Create object of Dog
        Dog dog = new Dog();

        // Methods from all levels of inheritance
        dog.eat();    // from Animal
        dog.walk();   // from Mammal
        dog.bark();   // from Dog
    }
}
```

**Output:** —



```
This animal eats food.
Mammals can walk.
Dog barks.
```

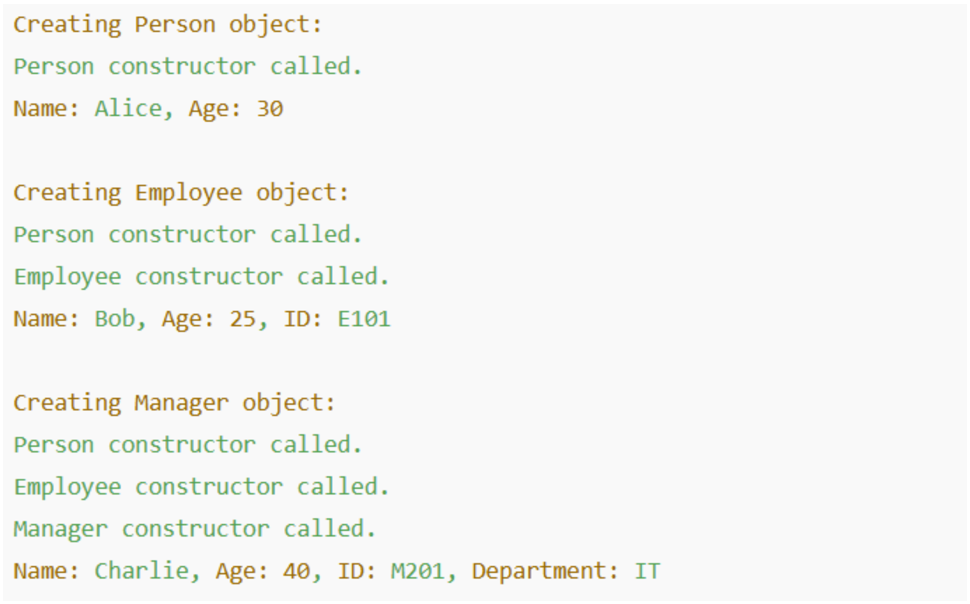
## Problem 4: Multi Level Inheritance

**Task:** Demonstrate calling the constructor of the base class from the constructor of the derived class. Create objects of person and employee classes to show the order of invocation of constructors.

**Code:** —

```java
// Base class
class Person {
    // private fields (encapsulation)
    private String name;
    private int age;

    // Constructor
    Person(String name, int age) {
        this.name = name;
        this.age = age;
        System.out.println("Person constructor called.");
    }

    // Getters & Setters
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}

// Derived class
class Employee extends Person {
    // private field
    private String employeeId;

    // Constructor
    Employee(String name, int age, String employeeId) {
        super(name, age); // calling Person constructor
        this.employeeId = employeeId;
        System.out.println("Employee constructor called.");
    }

    // Getter & Setter
    public String getEmployeeId() {
        return employeeId;
    }
    public void setEmployeeId(String employeeId) {
        this.employeeId = employeeId;
    }
}

// Derived class from Employee
class Manager extends Employee {
    // private field
    private String department;

    // Constructor
    Manager(String name, int age, String employeeId, String
       department) {
        super(name, age, employeeId); // calls Employee
           constructor
        this.department = department;
        System.out.println("Manager constructor called.");
    }

    // Getter & Setter
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
    }
}

// Main class
public class EncapsulationInheritanceExample {
    public static void main(String[] args) {
        System.out.println("Creating Person object:");
        Person p = new Person("Alice", 30);
        System.out.println("Name: " + p.getName() + ", Age: " + p.getAge());

        System.out.println("\nCreating Employee object:");
        Employee e = new Employee("Bob", 25, "E101");
        System.out.println("Name: " + e.getName() + ", Age: " + e.getAge() + ", ID: " + e.getEmployeeId());

        System.out.println("\nCreating Manager object:");
        Manager m = new Manager("Charlie", 40, "M201", "IT");
        System.out.println("Name: " + m.getName() + ", Age: " + m.getAge() +
                           ", ID: " + m.getEmployeeId() + ", Department: " + m.getDepartment());
    }
}
```

**Output:** —



```
Creating Person object:
Person constructor called.
Name: Alice, Age: 30

Creating Employee object:
Person constructor called.
Employee constructor called.
Name: Bob, Age: 25, ID: E101

Creating Manager object:
Person constructor called.
Employee constructor called.
Manager constructor called.
Name: Charlie, Age: 40, ID: M201, Department: IT
```

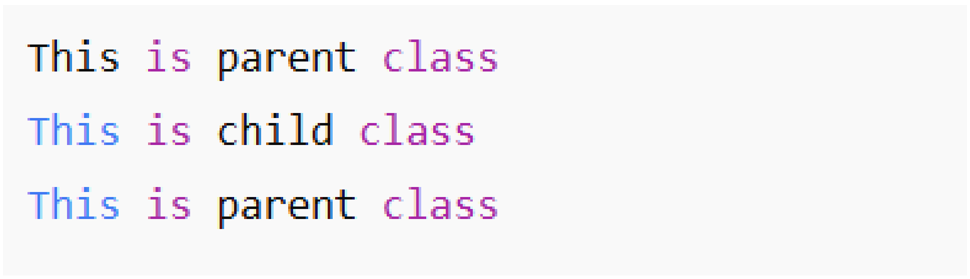
## Problem 5: Single Level Inheritance

**Task:** Create a class with a method that prints ”This is parent class” and its subclass with another method that prints ”This is child class”. Now, create an object for each of the class and call 1 - method of parent class by object of parent class 2 - method of child class by object of child class 3 - method of parent class by object of child class

**Code:** —

```java
// Parent class
class Parent {
    void displayParent() {
        System.out.println("This is parent class");
    }
}

// Child class extending Parent
class Child extends Parent {
    void displayChild() {
        System.out.println("This is child class");
    }
}

// Main class
public class ParentChildExample {
    public static void main(String[] args) {

        // 1. Method of parent class by object of parent class
        Parent p = new Parent();
        p.displayParent();

        // 2. Method of child class by object of child class
        Child c = new Child();
        c.displayChild();

        // 3. Method of parent class by object of child class
        c.displayParent();
    }
}
```

**Output:** —



```
This is parent class
This is child class
This is parent class
```

## Problem 6: One Base Class Two child class

**Task:** Create a class named 'Member' having the following members: Data members: 1 – Name, 2 – Age, 3 - Phone number, 4 – Address, 5 - Salary It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

**Code:** —

```java
// Base class
class Member {
    private String name;
    private int age;
    private String phoneNumber;
    private String address;
    private double salary;

    // Constructor
    public Member(String name, int age, String phoneNumber,
       String address, double salary) {
        this.name = name;
        this.age = age;
        this.phoneNumber = phoneNumber;
        this.address = address;
        this.salary = salary;
    }

    // Method to print salary
    public void printSalary() {
        System.out.println("Salary: " + salary);
    }

    // Method to display details
    public void displayDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Address: " + address);
        printSalary();
    }
}

// Employee class extending Member
class Employee extends Member {
    private String specialization;

    public Employee(String name, int age, String phoneNumber,
       String address, double salary, String specialization) {
        super(name, age, phoneNumber, address, salary); // call to parent constructor
        this.specialization = specialization;
    }

    public void displayEmployeeDetails() {
        displayDetails();
        System.out.println("Specialization: " + specialization);
    }
}

// Manager class extending Member
class Manager extends Member {
    private String department;

    public Manager(String name, int age, String phoneNumber, String address, double salary, String department) {
        super(name, age, phoneNumber, address, salary); // call to parent constructor
        this.department = department;
    }

    public void displayManagerDetails() {
        displayDetails();
        System.out.println("Department: " + department);
    }
}

// Main class
public class MemberExample {
    public static void main(String[] args) {
        // Creating Employee object
        Employee emp = new Employee("Alice", 28, "9876543210", "Hyderabad", 50000, "Software Development");
        System.out.println("=== Employee Details ===");
        emp.displayEmployeeDetails();

        System.out.println();

        // Creating Manager object
        Manager mgr = new Manager("Bob", 40, "9123456780", "Bengaluru", 90000, "IT Department");
        System.out.println("=== Manager Details ===");
        mgr.displayManagerDetails();
    }
}
```

**Output:** —

```
=== Employee Details ===
Name: Alice
Age: 28
Phone Number: 9876543210
Address: Hyderabad
Salary: 50000.0
Specialization: Software Development

=== Manager Details ===
Name: Bob
Age: 40
Phone Number: 9123456780
Address: Bengaluru
Salary: 90000.0
Department: IT Department
```