

## **Week 10**

1. Create class Person (Data Member- name, phone). Create two member inner classes Address (Data Member- House\_No, Street, City, State; Method- displayAddr()) and DateOfBirth (Data Member- Day, Month, Year; Method- displayDOB()). Display() is the method of Person class which will display name, address and date of birth of a Person object

```
class Person {  
    String name;  
    String phone;  
  
    public Person(String name, String phone) {  
        this.name = name;  
        this.phone = phone;  
    }  
  
    // Inner class Address  
    class Address {  
        String houseNo;  
        String street;  
        String city;  
        String state;  
  
        public Address(String houseNo, String street, String city, String state) {  
            this.houseNo = houseNo;  
            this.street = street;  
            this.city = city;  
            this.state = state;  
        }  
  
        public void displayAddr() {  
            System.out.println("Address: " + houseNo + ", " + street + ", " + city + ", " +  
state);  
        }  
    }  
}
```

```

// Inner class DateOfBirth
class DateOfBirth {
    int day;
    int month;
    int year;

    public DateOfBirth(int day, int month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

    public void displayDOB() {
        System.out.println("Date of Birth: " + day + "/" + month + "/" + year);
    }
}

public void Display() {
    System.out.println("Name: " + name);
    System.out.println("Phone: " + phone);

    // Create inner class objects
    Address addr = new Address("123", "MG Road", "Pune", "Maharashtra");
    DateOfBirth dob = new DateOfBirth(15, 8, 1990);

    addr.displayAddr();
    dob.displayDOB();
}
}

// Main class for testing
class PersonTest {
    public static void main(String[] args) {
        Person person = new Person("John Doe", "9876543210");
        person.Display();
    }
}

```

2. Create class Edible. Within that define two static classes Fruit and Vegetable. Fruit class will have two methods- fruitDetails() is a static

method and fruitPackaging() is a non-static method. Vegetable class also has similar methods - vegetableDetails() and vegetablePackaging(). Call all the four methods from main method.

```

class Edible {
    // Static inner class Fruit
    static class Fruit {
        public static void fruitDetails() {
            System.out.println("Fruit Details: Contains vitamins and minerals");
        }

        public void fruitPackaging() {
            System.out.println("Fruit Packaging: Packed in ventilated boxes");
        }
    }

    // Static inner class Vegetable
    static class Vegetable {
        public static void vegetableDetails() {
            System.out.println("Vegetable Details: Rich in fiber and nutrients");
        }

        public void vegetablePackaging() {
            System.out.println("Vegetable Packaging: Packed in plastic bags");
        }
    }
}

// Main class for testing
class EdibleTest {
    public static void main(String[] args) {
        // Call static methods without creating objects
        Edible.Fruit.fruitDetails();
        Edible.Vegetable.vegetableDetails();

        // Call non-static methods by creating objects
        Edible.Fruit fruit = new Edible.Fruit();
        Edible.Vegetable vegetable = new Edible.Vegetable();

        fruit.fruitPackaging();
        vegetable.vegetablePackaging();
    }
}

```

```
}
```

3. Create three different minMaxAdd() methods to calculate minimum, maximum and addition of integers, real numbers and characters.

```
class Calculator {  
    // For integers  
    public void minMaxAdd(int a, int b) {  
        int min = Math.min(a, b);  
        int max = Math.max(a, b);  
        int add = a + b;  
        System.out.println("Integers - Min: " + min + ", Max: " + max + ", Addition: " +  
add);  
    }  
  
    // For real numbers (double)  
    public void minMaxAdd(double a, double b) {  
        double min = Math.min(a, b);  
        double max = Math.max(a, b);  
        double add = a + b;  
        System.out.println("Real Numbers - Min: " + min + ", Max: " + max + ",  
Addition: " + add);  
    }  
  
    // For characters  
    public void minMaxAdd(char a, char b) {  
        char min = (char) Math.min(a, b);  
        char max = (char) Math.max(a, b);  
        int add = a + b; // ASCII values addition  
        System.out.println("Characters - Min: " + min + "", Max: " + max + "", ASCII  
Addition: " + add);  
    }  
}  
  
// Main class for testing  
class CalculatorTest {  
    public static void main(String[] args) {  
        Calculator calc = new Calculator();  
  
        calc.minMaxAdd(50,20);  
        calc.minMaxAdd(23.5,16.7);  
    }  
}
```

```
    calc.minMaxAdd('A', 'D');
}
}
```

4. Create a class ObjectOriented which has methods- abstraction(), polymorphism() and inheritance(). Create a class JavaLanguage which inherits from ObjectOriented class and has its own methods- persistence() and interfaces(). Create an object of JavaLanguage class to access all of its own and parent's methods.

```
class ObjectOriented {
    public void abstraction() {
        System.out.println("Abstraction: Hiding implementation details");
    }

    public void polymorphism() {
        System.out.println("Polymorphism: Same method, different behaviors");
    }

    public void inheritance() {
        System.out.println("Inheritance: Acquiring properties of parent class");
    }
}

class JavaLanguage extends ObjectOriented {
    public void persistence() {
        System.out.println("Persistence: Storing object state in databases");
    }

    public void interfaces() {
        System.out.println("Interfaces: Defining contracts for classes");
    }
}

// Main class for testing
class OOPTest {
    public static void main(String[] args) {
        JavaLanguage java = new JavaLanguage();

        // Parent class methods
        java.abstraction();
        java.polymorphism();
```

```

        java.inheritance();

        // Own methods
        java.persistence();
        java.interfaces();
    }
}

```

5. In previous question, create a new class C++ which also inherits from ObjectOriented class and has its own methods- template() and friendFunction(). Create an object of C++ class to access all of its own and parent's methods.

```

class CPlusPlus extends ObjectOriented {
    public void template() {
        System.out.println("Templates: Generic programming in C++");
    }

    public void friendFunction() {
        System.out.println("Friend Function: Accessing private members");
    }
}

// Main class for testing
class CPlusPlusTest{
    public static void main(String[] args) {
        CPlusPlus cpp = new CPlusPlus();

        // Parent class methods
        cpp.abstraction();
        cpp.polymorphism();
        cpp.inheritance();

        // Own methods
        cpp.template();
        cpp.friendFunction();
    }
}

```

6. Create class University which has data members- name and ranking. Create class Faculty that extends University class has data member- name and method- Details(). Create a new class Department which

is derived from Faculty and has data member- name, chairman and method- Details() and Display() where Display() method calls Details() methods of both Faculty and Department class in its body. Create an object of Department class to Display() method and University ranking.

```
class University {  
    String name;  
    int ranking;  
  
    public University(String name, int ranking) {  
        this.name = name;  
        this.ranking = ranking;  
    }  
}  
  
class Faculty extends University {  
    String facultyName;  
  
    public Faculty(String uniName, int ranking, String facultyName) {  
        super(uniName, ranking);  
        this.facultyName = facultyName;  
    }  
  
    public void Details() {  
        System.out.println("Faculty Name: " + facultyName);  
    }  
}  
  
class Department extends Faculty {  
    String departmentName;  
    String chairman;  
  
    public Department(String uniName, int ranking, String facultyName,  
                     String departmentName, String chairman) {  
        super(uniName, ranking, facultyName);  
        this.departmentName = departmentName;  
        this.chairman = chairman;  
    }  
  
    public void Details() {
```

```

        System.out.println("Department Name: " + departmentName);
        System.out.println("Chairman: " + chairman);
    }

    public void Display() {
        System.out.println("University: " + name + " (Ranking: " + ranking + ")");
        super.Details(); // Faculty Details
        Details(); // Department Details
    }
}

// Main class for testing
class UniversityTest {
    public static void main(String[] args) {
        Department dept = new Department("ABC University", 5,
            "Engineering", "Computer Science",
            "Dr. Smith");
        dept.Display();
    }
}

```

**7. Create a class Employee (Data Members – empName, empld).**

Create two member inner classes:

- Salary (Data Members – basic, hra, pf; Method – displaySalary() to print salary details).
  - JoiningDate (Data Members – day, month, year; Method – displayJoiningDate() to print joining date).
- In the Employee class, create a method displayEmployee() that prints the employee's name, ID, salary details, and joining date.

```

class Employee {
    String empName;
    int empld;

    public Employee(String empName, int empld) {
        this.empName = empName;
        this.empld = empld;
    }

    // Inner class Salary
    class Salary{

```

```

        double basic;
        double hra;
        double pf;

        public Salary(double basic, double hra, double pf) {
            this.basic = basic;
            this.hra = hra;
            this.pf = pf;
        }

        public void displaySalary() {
            System.out.println("Salary Details:");
            System.out.println(" Basic: " + basic);
            System.out.println(" HRA: " + hra);
            System.out.println(" PF: " + pf);
            System.out.println(" Total: " + (basic + hra - pf));
        }
    }

    // Inner class JoiningDate
    class JoiningDate {
        int day;
        int month;
        int year;

        public JoiningDate(int day, int month, int year) {
            this.day = day;
            this.month = month;
            this.year = year;
        }

        public void displayJoiningDate() {
            System.out.println("Joining Date: " + day + "/" + month + "/" + year);
        }
    }

    public void displayEmployee() {
        System.out.println("Employee Name: " + empName);
        System.out.println("Employee ID: " + emplId);

        // Create inner class objects
    }
}

```

```

Salary salary = new Salary(50000, 10000, 5000);
JoiningDate joinDate = new JoiningDate(1, 6, 2023);

    salary.displaySalary();
    joinDate.displayJoiningDate();
}

}

// Main class for testing
class EmployeeTest {
    public static void main(String[] args) {
        Employee emp = new Employee("Alice Johnson", 101);
        emp.displayEmployee();
    }
}

```

#### 8. Create a class Shape with overloaded methods area():

- area(int side) – calculates area of a square.
- area(int length, int breadth) – calculates area of a rectangle.
- area(double radius) – calculates area of a circle.

```

class Shape {
    // Area of square
    public double area(int side) {
        return side * side;
    }

    // Area of rectangle
    public double area(int length, int breadth) {
        return length * breadth;
    }

    // Area of circle
    public double area(double radius) {
        return Math.PI * radius * radius;
    }
}

// Main class for testing
class ShapeTest {
    public static void main(String[] args) {
        Shape shape = new Shape();

        double squareArea = shape.area(5);
    }
}

```

```

        double rectangleArea = shape.area(4, 6);
        double circleArea = shape.area(3);

        System.out.println("Area of Square (side=5): " + squareArea);
        System.out.println("Area of Rectangle (4x6): " + rectangleArea);
        System.out.println("Area of Circle (radius=3.5): " + circleArea);
    }
}

```

9. Create a class Vehicle with a method run(). Create subclasses Bike and Car that override the run() method. In the main() method, use a reference of Vehicle to call run() for objects of Bike and Car.

```

class Vehicle {

    public void run() {

        System.out.println("Vehicle is running");

    }

}

```

```

class Bike extends Vehicle {

    @Override

    public void run() {

        System.out.println("Bike is running safely at 60kmph");

    }

}

```

```

class Car extends Vehicle {

    @Override

    public void run() {

        System.out.println("Car is running smoothly at 80kmph");

    }

}

```

```
// Main class for testing

class VehicleTest {

    public static void main(String[] args) {
        // Using Vehicle reference for polymorphism
        Vehicle vehicle1 = new Bike();
        Vehicle vehicle2 = new Car();

        vehicle1.run(); // Calls Bike's run method
        vehicle2.run(); // Calls Car's run method

        // Direct object creation
        System.out.println("\nDirect object calls:");
        Bike bike = new Bike();
        Car car = new Car();

        bike.run();
        car.run();
    }
}
```