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.

Main.java Person.java

```java
package week10_1;

class Person {  2 usages
    String name;  2 usages
    String phone;  2 usages
    private Address address;  2 usages
    private DateOfBirth dob;  2 usages

    public Person(String name, String phone, String houseNo, String street,
                  String city, String state, int day, int month, int year) {
        this.name = name;
        this.phone = phone;
        this.address = new Address(houseNo, street, city, state);
        this.dob = new DateOfBirth(day, month, year);
    }
    public void Display() {  1 usage
        System.out.println("--- Person Details ---");
        System.out.println("Name: " + this.name);
        System.out.println("Phone: " + this.phone);

        this.address.displayAddr();
        this.dob.displayDOB();
        System.out.println("----------------------");
    }
    class Address {  2 usages
        String House_No;  2 usages
        String Street;  2 usages
        String City;  2 usages
        String State;  2 usages

        public Address(String houseNo, String street, String city, String state) {
            this.House_No = houseNo;
            this.Street = street;
            this.City = city;
            this.State = state;
        }

        public void displayAddr() {  1 usage
            System.out.println("Address: " + this.House_No + ", " +
                    this.Street + ", " + this.City + ", " + this.State);
        }
    }
    class DateOfBirth {  2 usages
        int Day;  2 usages
        int Month;  2 usages
        int Year;  2 usages

        public DateOfBirth(int day, int month, int year) {  1 usage
            this.Day = day;
            this.Month = month;
            this.Year = year;
        }

        public void displayDOB() {  1 usage
            System.out.println("Date of Birth: " +
                    this.Day + "/" + this.Month + "/" + this.Year);
        }
    }
}
```

WEEK 10

Main.java | Person.java

```java
package week10_1;

public class Main {
    public static void main(String[] args) {
        Person person1 = new Person(
                name: "Amit Kumar",
                phone: "+91-9876543210",
                houseNo: "123-B",
                street: "Kamla Nagar",
                city: "Agra",
                state: "Uttar Pradesh",
                day: 15,
                month: 8,
                year: 1990
        );
        person1.Display();
    }
}
```

```
--- Person Details ---
Name: Amit Kumar
Phone: +91-9876543210
Address: 123-B, Kamla Nagar, Agra, Uttar Pradesh
Date of Birth: 15/8/1990
----------------------
```

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.

WEEK 10

Main.java × Edible.java

```java
package week10_2;

public class Main {

    public static void main(String[] args) {

        System.out.println("--- Demonstrating Fruit Class ---");
        Edible.Fruit.fruitDetails();

        Edible.Fruit myFruit = new Edible.Fruit();
        myFruit.fruitPackaging();

        System.out.println("\n--- Demonstrating Vegetable Class ---");
        Edible.Vegetable.vegetableDetails();

        Edible.Vegetable myVegetable = new Edible.Vegetable();
        myVegetable.vegetablePackaging();
    }
}
```

Main.java Edible.java ×

```java
package week10_2;

public class Edible {  6 usages

    public static class Fruit {  3 usages
        public static void fruitDetails() {  1 usage
            System.out.println("Static Method (Fruit)");
        }
        public void fruitPackaging() {  1 usage
            System.out.println("Non-Static Method (Fruit)");
        }
    }
    public static class Vegetable {  3 usages
        public static void vegetableDetails() {  1 usage
            System.out.println("Static Method (Vegetable)");
        }
        public void vegetablePackaging() {  1 usage
            System.out.println("Non-Static Method (Vegetable)");
        }
    }
}
```

```
--- Demonstrating Fruit Class ---
Static Method (Fruit)
Non-Static Method (Fruit)

--- Demonstrating Vegetable Class ---
Static Method (Vegetable)
Non-Static Method (Vegetable)
```

WEEK 10

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

Main.java

```java
package week10_3;

public class Main {
    public static void minMaxAdd(int a, int b) {  1 usage
        System.out.println("Integer");
        System.out.println("Min: " + Math.min(a, b));
        System.out.println("Max: " + Math.max(a, b));
        System.out.println("Sum: " + (a + b));
    }
    public static void minMaxAdd(double a, double b) {  1 usage
        System.out.println("Double (Real)");
        System.out.println("Min: " + Math.min(a, b));
        System.out.println("Max: " + Math.max(a, b));
        System.out.println("Sum: " + (a + b));
    }
    public static void minMaxAdd(char a, char b) {  1 usage
        System.out.println("Character");
        System.out.println("Min: '" + (char)Math.min(a, b) + "'");
        System.out.println("Max: '" + (char)Math.max(a, b) + "'");
        System.out.println("Sum (as int): " + (a + b));
    }
    public static void main(String[] args) {
        minMaxAdd( a: 10,  b: 20);
        minMaxAdd( a: 7.5,  b: 3.2);
        minMaxAdd( a: 'B',  b: 'X');
    }
}
```

```
Integer
Min: 10
Max: 20
Sum: 30
```

```
Double (Real)
Min: 3.2
Max: 7.5
Sum: 10.7
```

```
Character
Min: 'B'
Max: 'X'
Sum (as int): 154
```

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.

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.

WEEK 10

Main.java | JavaLanguage.java | ObjectOriented.java | Cpp.java

```java
package week10_4;

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

        JavaLanguage myJava = new JavaLanguage();
        System.out.println("Accessing Parent's Methods (from ObjectOriented)");
        myJava.abstraction();
        myJava.polymorphism();
        myJava.inheritance();

        System.out.println("\nAccessing Own Methods (from JavaLanguage)");
        myJava.persistence();
        myJava.interfaces();

        Cpp myCpp = new Cpp();
        System.out.println("\nAccessing Parent's Methods (from ObjectOriented)");
        myCpp.abstraction();
        myCpp.polymorphism();
        myCpp.inheritance();

        System.out.println("\nAccessing Own Methods (from CppLanguage)");
        myCpp.template();
        myCpp.friendFunction();
    }
}
```

.java | ObjectOriented.java | JavaLanguage.

```java
package week10_4;

class ObjectOriented {  2 usages  2 inheritors

    public void abstraction() {  2 usages
        System.out.println("Abstraction");
    }

    public void polymorphism() {  2 usages
        System.out.println("Polymorphism");
    }

    public void inheritance() {  2 usages
        System.out.println("Inheritance");
    }
}
```

.java | ObjectOriented.java | JavaLanguag

```java
package week10_4;

class JavaLanguage extends ObjectOriented {

    public void persistence() {  1 usage
        System.out.println("Persistence");
    }

    public void interfaces() {  1 usage
        System.out.println("Interfaces");
    }
}
```

WEEK 10

Main.java | ObjectOriented.java | JavaLanguage.java | Cpp.java

```
1   package week10_4;
2
3   class Cpp extends ObjectOriented {  2 usages
4       public void template() { System.out.println("Templates"); }
7       public void friendFunction() { System.out.println("Friend Functions"); }
10  }
11
```

```
Accessing Parent's Methods (from ObjectOriented)
Abstraction
Polymorphism
Inheritance

Accessing Own Methods (from JavaLanguage)
Persistence
Interfaces

Accessing Parent's Methods (from ObjectOriented)
Abstraction
Polymorphism
Inheritance

Accessing Own Methods (from CppLanguage)
Templates
Friend Functions
```

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.

WEEK 10

Main.java × University.java Faculty.java Department.java

```java
package week10_6;

public class Main {
    public static void main(String[] args) {
        Department csDept = new Department(
                uniName: "Stanford University",
                ranking: 1,
                facultyName: "School of Engineering",
                deptName: "Computer Science",
                chairman: "Dr. John Hennessy"
        );
        System.out.println("Calling Display():");
        csDept.Display();
        System.out.println("\nAccessing University Ranking directly:");
        System.out.println("Ranking: " + csDept.ranking);
    }
}
```

Main.java University.java × Faculty.java Dep

```java
package week10_6;

class University {  1 usage  2 inheritors
    String name;  2 usages
    int ranking;  2 usages

    public University(String name, int ranking) {
        this.name = name;
        this.ranking = ranking;
    }
}
```

Main.java × University.java Faculty.java × Department.java

```java
package week10_6;

class Faculty extends University {  1 usage  1 inheritor
    String name;  2 usages

    public Faculty(String uniName, int ranking, String facultyName) {
        super(uniName, ranking);
        this.name = facultyName;
    }
    public void Details() {  2 usages  1 override
        System.out.println("University: " + super.name);
        System.out.println("Faculty: " + this.name);
    }
}
```

WEEK 10

Main.java | University.java | Faculty.java | Department.java

```java
package week10_6;

class Department extends Faculty {  2 usages
    String name;  2 usages
    String chairman;  2 usages

    public Department(String uniName, int ranking, String facultyName,  1 usage
                      String deptName, String chairman) {

        super(uniName, ranking, facultyName);

        this.name = deptName;
        this.chairman = chairman;
    }
    @Override  2 usages
    public void Details() {
        System.out.println("Department: " + this.name);
        System.out.println("Chairman: " + this.chairman);
    }
    public void Display() {  1 usage
        super.Details();
        this.Details();
    }
}
```

```
Calling Display():
University: Stanford University
Faculty: School of Engineering
Department: Computer Science
Chairman: Dr. John Hennessy

Accessing University Ranking directly:
Ranking: 1
```

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

WEEK 10

Main.java × Employee.java

```java
package week10_7;

public class Main {
    public static void main(String[] args) {
        Employee emp1 = new Employee( empName: "Rajesh Kumar",
                empId: 101, basic: 80000, hra: 20000,
                pf: 8000, day: 14, month: 10, year: 2021);
        emp1.displayEmployee();
    }
}

```

Main.java Employee.java ×

```java
package week10_7;

class Employee {
    String empName;
    int empId;
    private Salary salary;
    private JoiningDate joiningDate;
    public Employee(String empName, int empId, double basic,
                    double hra, double pf, int day, int month, int year) {
        this.empName = empName;
        this.empId = empId;
        this.salary = new Salary(basic, hra, pf);
        this.joiningDate = new JoiningDate(day, month, year);
    }
    public void displayEmployee() {
        System.out.println("Employee Name: " + this.empName);
        System.out.println("Employee ID: " + this.empId);
        this.salary.displaySalary();
        this.joiningDate.displayJoiningDate();
    }
```

WEEK 10

```
    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() {
            double total = (this.basic + this.hra) - this.pf;
            System.out.println("Salary Details for: " + empName);
            System.out.println("Basic Pay: " + this.basic);
            System.out.println("HRA: " + this.hra);
            System.out.println("PF: " + this.pf);
            System.out.println("Total Salary: " + total);
        }
    }
    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: " + this.day + "/" + this.month + "/" + this.year);
        }
    }
}
```

```
Employee Name: Rajesh Kumar
Employee ID: 101
Salary Details for: Rajesh Kumar
Basic Pay: 80000.0
HRA: 20000.0
PF: 8000.0
Total Salary: 92000.0
Joining Date: 14/10/2021
```

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.

WEEK 10

```java
package week10_8;

public class Main {
    public static void main(String[] args) {
        Shape myShape = new Shape();
        myShape.area( side: 10);
        myShape.area( length: 20,  breadth: 10);
        myShape.area( radius: 7.5);
    }
}
```

```java
package week10_8;

class Shape {
    public void area(int side) {
        int result = side * side;
        System.out.println("Area of the Square: " + result);
    }
    public void area(int length, int breadth) {
        int result = length * breadth;
        System.out.println("Area of the Rectangle: " + result);
    }
    public void area(double radius) {
        double result = Math.PI * radius * radius;
        System.out.printf("Area of the Circle: %.2f\n", result);
    }
}
```

```
Area of the Square: 100
Area of the Rectangle: 200
Area of the Circle: 176.71
```

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.

```java
package week10_9;

public class Main {
    public static void main(String[] args) {
        Vehicle v1 = new Bike();
        Vehicle v2 = new Car();
        v1.run();
        v2.run();
    }
}
```

```java
package week10_9;

class Vehicle {  4 usages  2 inheritors
    public void run() {  2 usages  2 overrides
        System.out.println("Vehicle is running.");
    }
}
```

WEEK 10

Main.java | Vehicle.java | Bike.java | Car.jav

```java
package week10_9;

class Bike extends Vehicle {  1 usage
    @Override  2 usages
    public void run() {
        System.out.println("Bike is running.");
    }
}
```

Main.java | Vehicle.java | Bike.java | Car.ja

```java
package week10_9;

class Car extends Vehicle {  1 usage
    @Override  2 usages
    public void run() {
        System.out.println("Car is running.");
    }
}

```

```
Bike is running.
Car is running.
```