WEEK-10

Question: 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.

Code:

```
class Person {
   String name, phone;
Person(String n, String p) {
name = n; phone = p;
   }
   class Address {
      String house, street, city, state;
      Address(String h, String s, String c, String st) {
house = h; street = s; city = c; state = st;
      }
      void displayAddr() {
         System.out.println("Address: " + house + ", " + street + ", " + city + ", " + state);
      }
   }
   class DateOfBirth {
int d, m, y;
      DateOfBirth(int a, int b, int c) {
d = a; m = b; y = c;
      }
      void displayDOB() {
         System.out.println("DOB: " + d + "/" + m + "/" + y);
      }
   }
   void Display(Address a, DateOfBirth d) {
```

```
        System.out.println("Name: " + name + ", Phone: " + phone);
        a.displayAddr();
        d.displayDOB();
    }
    public static void main(String[] args) {
        Person p = new Person("Ali", "12345");
        Address a = p.new Address("12", "Park Road", "Delhi", "India");
DateOfBirth dob = p.new DateOfBirth(1, 1, 2000);
        p.Display(a, dob);
    }
}
```

Output:

Output:

Name: Ali, Phone: 12345

Address: 12, Park Road, Delhi, India

DOB: 1/1/2000

Question: 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.

Code:

```
class Edible {     static class Fruit {         static void fruitDetails() {
System.out.println("Fruit: Sweet and Juicy"); }         void fruitPackaging() {
System.out.println("Fruit packed in box"); }
    }
    static class Vegetable {         static void vegetableDetails() {
System.out.println("Vegetable: Healthy and Green"); }         void vegetablePackaging() {
System.out.println("Vegetable packed in bag"); }
    }
```

```
   public static void main(String[] args) {
      Fruit.fruitDetails();
      Vegetable.vegetableDetails();       new
Fruit().fruitPackaging();       new
Vegetable().vegetablePackaging();
   }
}
```

Output:

Output:

Fruit: Sweet and Juicy

Vegetable: Healthy and Green

Fruit packed in box

Vegetable packed in bag

Question: 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.

Code:

```
class ObjectOriented {     void abstraction() { System.out.println("Concept
of Abstraction"); }     void polymorphism() { System.out.println("Concept of
Polymorphism"); }     void inheritance() { System.out.println("Concept of
Inheritance"); }
}
class JavaLanguage extends ObjectOriented {     void persistence() { System.out.println("Java
persistence using JVM"); }     void interfaces() { System.out.println("Java supports multiple
inheritance through interfaces"); }     public static void main(String[] args) {
      JavaLanguage j = new JavaLanguage();
      j.abstraction();
      j.polymorphism();
```

```
        j.inheritance();
        j.persistence();
        j.interfaces();
    }
}
```

Output:

Output:

Concept of Abstraction

Concept of Polymorphism

Concept of Inheritance

Java persistence using JVM

Java supports multiple inheritance through interfaces

Question: 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.

Code:

```
class ObjectOriented2 {     void abstraction() { System.out.println("Concept
of Abstraction"); }     void polymorphism() { System.out.println("Concept of
Polymorphism"); }     void inheritance() { System.out.println("Concept of
Inheritance"); }
}
class CppLanguage extends ObjectOriented2 {     void template() {
System.out.println("Templates in C++"); }     void friendFunction() {
System.out.println("Friend function in C++"); }     public static void
main(String[] args) {        CppLanguage c = new CppLanguage();
        c.abstraction();
        c.polymorphism();
        c.inheritance();
        c.template();
```

```
        c.friendFunction();
    }
}
```

Output:

Output:

Concept of Abstraction

Concept of Polymorphism

Concept of Inheritance

Templates in C++

Friend function in C++

Question: 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.

Code:

```
class University {
    String name = "ABC University";
    int ranking = 1;
}
class Faculty extends University {
String name = "Computer Science";
    void Details() {
        System.out.println("Faculty: " + name);
    }
}
class Department extends Faculty {
```

```
   String name = "IT";
String chairman = "Dr. Ali";
void Details() {
      System.out.println("Department: " + name + ", Chairman: " + chairman);
   }
   void Display() {
super.Details();
this.Details();
      System.out.println("University Ranking: " + ranking);
   }
   public static void main(String[] args) {
Department d = new Department();
      d.Display();
   }
}
```

Output:

Output:

Faculty: Computer Science

Department: IT, Chairman: Dr. Ali

University Ranking: 1

Question: 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.

Code:

```
class Employee {
String empName;
int empId;
Employee(String n,
```

```
int id){ empName =
n; empId = id; }

  class Salary {
double basic, hra, pf;
    Salary(double b, double h, double p){ basic=b; hra=h; pf=p; }    void
displaySalary(){ System.out.println("Basic: "+basic+", HRA: "+hra+", PF: "+pf); }
  }
  class JoiningDate {
int day, month, year;
    JoiningDate(int d, int m, int y){ day=d; month=m; year=y; }    void displayJoiningDate(){
System.out.println("Joining Date: "+day+"/"+month+"/"+year); }
  }
  void displayEmployee(Salary s, JoiningDate j){
    System.out.println("Name: "+empName+", ID: "+empId);
    s.displaySalary();
    j.displayJoiningDate();
  }
  public static void main(String[] args){
    Employee e = new Employee("Rahul", 101);
    Salary s = e.new Salary(40000, 5000, 2000);
JoiningDate j = e.new JoiningDate(1, 6, 2020);
    e.displayEmployee(s, j);
  }
}
```

Output:

Output:

Name: Rahul, ID: 101

Basic: 40000.0, HRA: 5000.0, PF: 2000.0

Joining Date: 1/6/2020

Question: 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.

Code: class

```
ShapeOverload {
   int area(int side) { return side*side; }     int area(int length,
int breadth) { return length * breadth; }     double
area(double radius) { return 3.14 * radius * radius; }     public
static void main(String[] args){
      ShapeOverload s = new ShapeOverload();
      System.out.println("Square area: "+s.area(4));
      System.out.println("Rectangle area: "+s.area(5,3));
      System.out.println("Circle area: "+s.area(2.5));
   }
}
```

Output:

Output:

Square area: 16

Rectangle area: 15

Circle area: 19.625

Question: 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.

Code:

```
class VehicleRun {     void run(){
System.out.println("Vehicle running"); }
}
class Bike extends VehicleRun {
```

```
   @Override
   void run(){ System.out.println("Bike is running"); }
}
class Car extends VehicleRun {     @Override
void run(){ System.out.println("Car is running"); }
}
public class MainRun {     public static
void main(String[] args){
VehicleRun v;        v = new Bike();
      v.run();       v
= new Car();
      v.run();
   }
}
```

Output:

Output:

Bike is running

Car is running