EXPERIMENT 10

// Inheritance in Java

Prerequisite Knowledge

Inheritance – It is a feature of object-oriented programming that allows the objects of a class to acquire the characteristics of some other class.

- ➔ The class that gets inherited is known as the **parent class** or **base class**, and the class that inherits another class is known as the **child class** or **subclass** or **derived class**.
  So when the objects of the derived class are created, they possess the characteristics defined in this class as well as its parent class.
- ➔ The feature of inheritance empowers the programmer with the facility of **reusability of code**, as by deriving a class from another and adding some new functionalities to it, he can reuse a class that is almost (not exactly) what he wants, and update it in such a way that the original class does not get transformed.

Types of Inheritance in Java -

I. **Simple/Single Inheritance -** Only one class is derived from a base class.
II. **Multilevel Inheritance -** Only one class is derived from another derived class.
III. **Hierarchical Inheritance -** Multiple classes are derived from one base class.
IV. **Hybrid Inheritance -** A combination of two or more inheritance mechanisms discussed above.

Access Specifiers in Java -

Access specifiers, also known as **access modifiers** or **visibility specifiers**, are keywords that control access to a program's classes, fields, and methods.

- ➔ They are crucial to decide which data members and member functions of a class can be accessed by some other class or not.

Here are some of the access specifiers in Java:

- **Public -** It allows unrestricted access to a class and its members from any other class or package.
- **Private -** It restricts access to a class and its data members only within the same class, preventing direct access from outside classes.
- **Protected -** It allows access to a class and its data members within the same package or by subclasses, even if they are in different packages.
- **Default -** Also known as "**package-private**" or "**no modifier**", this is the default access that has no keyword. It allows access to a class and its data members only from the other classes in the same package.

PROGRAM 10

// Simple Inheritance

Source Code -

```
class Parent
{
   private int ParentNo;
   public int num1;

   Parent()
   {
      ParentNo=1; num1=10;
   }

   int Add_Parent()       //default function
   {
      return (ParentNo+num1);
   }
}
class Child extends Parent
{
   private int ChildNo;

   Child()
   {
      ChildNo=2;
   }
   int Add_Child()       //default function
   {
      return (ChildNo + Add_Parent());
   }
}
class MainClass
{
   public static void main(String args[])
   {
      Parent p = new Parent();
      Child c = new Child();
```

```
    System.out.println("Sum of parent class and child class numbers = " + c.Add_Child());
    }
}
```

Output -

```
Sum of parent class and child class numbers = 13
```

PROGRAM 11

// Multilevel Inheritance

Source Code -

```
class Parent
{
    private int ParentNo;
    public int num1;

    Parent()
    {
        ParentNo=1; num1=10;
    }

    int Add_Parent()        //default function
    {
        return (ParentNo+num1);
    }
}
class Child extends Parent
{
    private int ChildNo;

    Child()
    {
        ChildNo=2;
    }
    int Add_Child()        //default function
    {
        return (ChildNo + Add_Parent());
    }
}
class GrandChild extends Child
{
   private int GrandChildNo;

   GrandChild()
   {
      GrandChildNo=3;
   }

   int Add_GrandChild()        //default function
   {
      return (GrandChildNo + Add_Child());
   }
}
class MainClass
{
   public static void main(String args[])
   {
      Parent p = new Parent();
      Child c = new Child();
      GrandChild g = new GrandChild();
          System.out.println("Sum of parent, child and grandchild class numbers = " +
g.Add_GrandChild());
   }
}
```

Output -

```
Sum of parent, child and grandchild class numbers = 16
```

PROGRAM 12

// Hybrid Inheritance (Hierarchical Inheritance included)

Source Code -

```
class Parent
{
   private int ParentNo;
   public int num1;

   Parent()
   {
      ParentNo=1; num1=10;
   }

   int Add_Parent()      //default function
   {
      return (ParentNo+num1);
   }
}
class Child extends Parent
{
   private int ChildNo;

   Child()
   {
      ChildNo=2;
   }
   int Add_Child()      //default function
   {
      return (ChildNo + Add_Parent());
   }
}
class GrandChild1 extends Child
{
   private int GrandChildNo1;

   GrandChild1()
   {
      GrandChildNo1=13;
```

```
    }

    int Add_GrandChild1()        //default function
    {
        return (GrandChildNo1 + Add_Child());
    }
}
class GrandChild2 extends Child
{
    private int GrandChildNo2;

    GrandChild2()
    {
        GrandChildNo2=31;
    }

    int Add_GrandChild2()        //default function
    {
        return (GrandChildNo2 + Add_Child());
    }
}
class MainClass
{
    public static void main(String args[])
    {
        Parent p = new Parent();
        Child c = new Child();
        GrandChild1 g1 = new GrandChild1();
        GrandChild2 g2 = new GrandChild2();

            System.out.println("Sum of parent, child and grandchild class numbers = " +
g1.Add_GrandChild1() + " and " + g2.Add_GrandChild2());
    }
}
```

Output -

```
Sum of parent, child and grandchild class numbers = 26 and 44
```