Website

https://www.curiousinfotechsolutions.com

Java Programming

Day-12

Email id:
info@curiousinfotechsolutions.com

Syllabus Day-12

- Applications of classes and objects
- Static Members
- Nesting of Methods
- Inheritance in Java
- Single Inheritance Concept
- Multilevel Inheritance Concept
- Hierarchical Inheritance Concept

Exp-1

- Create a class
  1. Rectangle
     - Declaring of Variables
       - Length and Width
     - Defining the Methods
       - getData()/Rectangle()
       - rectArea()
  2. Main
     - Defining the method
       - main()
         - Display area of object

ObjRec1	
Length	15
Width	10

ObjRec2	
Length	20
Width	12

ObjRec1 = 150
ObjRec2= 240

```
class Rectangle
   {
      int Length, Width;
      void getData(int len, int wid)
         {
            Length = len;
            Width = wid;
         }
      int rectArea()
         {
            return Length * Width;
         }
   }
```



```
class Main
   {
      public static void main (String[] args) {
         int area1, area2;
         Rectangle ObjRec1 = new Rectangle();
         Rectangle ObjRec2 = new Rectangle();
         ObjRec1.Length = 15;
         ObjRec1.Width = 10;
         area1 = ObjRec1.Length * ObjRec1.Length;

         ObjRec2.getData(20,12);
         area2 = ObjRec2.rectArea();

         System.out.println("Area1 = "+ area1);
         System.out.println("Area2 = "+ area2);
      }
   }
```

Static Members

- Generally, class contains two sections:
  - Declare Variables
  - Declare Methods
- These variables and methods are called instance variables and instance methods. And this is because every time the class is instantiated and a **new copy of each** of them is created. And also they are **accessed through using the objects** i.e. with dot operator.

Contd..

- Let
  - static int count;
  - static int max (int x, int y){}
- These members are associated with the class itself ***rather than individual objects***.
- Static members are used when we want to have a variable common to all instances of a class.

```
class Rectangle
   {
      int Length, Width;
      void getData(int len, int wid)
         {
            Length = len;
            Width = wid;
         }
      int rectArea()
         {
            return Length * Width;
         }
   }
```

```
class Main
   {
      public static void main (String[] args) {
         int area1, area2;
         Rectangle.getData(15,10);

         area1 = Rectangle.rectArea();

         System.out.println("Area1 = "+ area1);

         Rectangle Obj2area = new Rectangle();
         area2 = Obj2area.rectArea();
         System.out.println("Area2 = "+ area2);
      }
   }
```

```
Area1 = 150
Area2 = 150
```

Condition in using static members

- They can only call other static members.
- They can only access static data.
- They cannot refer to this or super in any way.

Nesting of Methods

- A method can be called by using only its name by another of the same class. This is known as nesting of methods.

- Find Largest Number from give three numbers
- Class Largest
  - int A,B,C
  - Getdata(x,y,z) {A=x;B=y,C=Z;}
  - Big(M,N)
  - Display() {B=Big(Big(x,y),z); println(B);}

Main

```
        main()
                òbject – N;
                N.getdata(10,20,30);
                N.Display()
```

```
class NestingMethods
   {
      static int x,y,z;
      static void getData(int A, int B, int C)
      {
         x = A; y = B; z = C;
      }
      static int Big(int M, int N)
      {
         if (M>N) return M;
         else return N;
      }
      static int largetNumber()
      {
         int B = Big(Big(x,y),z);
         return B;
      }
   }
```

```
class Main
   {
      public static void main (String[] args) {
         int largest;
         NestingMethods.getData(10,20,30);
         largest = NestingMethods.largetNumber();
         System.out.println(largest);
      }
   }
```

Inheritance in Java

- Inheritance in Java is **a mechanism in which one object acquires all the properties and behaviors of a parent object**.
- It is an important part of OOPs (Object Oriented programming system).
- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

Contd..

- When we inherit from an existing class, we can reuse methods and fields of the parent class. Moreover, we can add new methods and fields in our current class also.
- Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

Why use inheritance in java

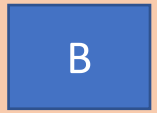
- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

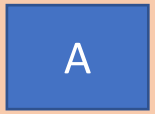
Terms used in Inheritance

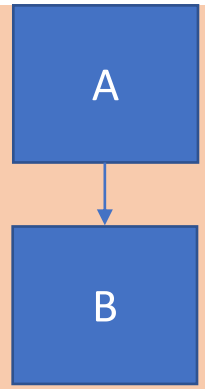
- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

The syntax of Java Inheritance


```
1.class Subclass-name extends Superclass-name
2.{
3.   //methods and fields
4.}
```





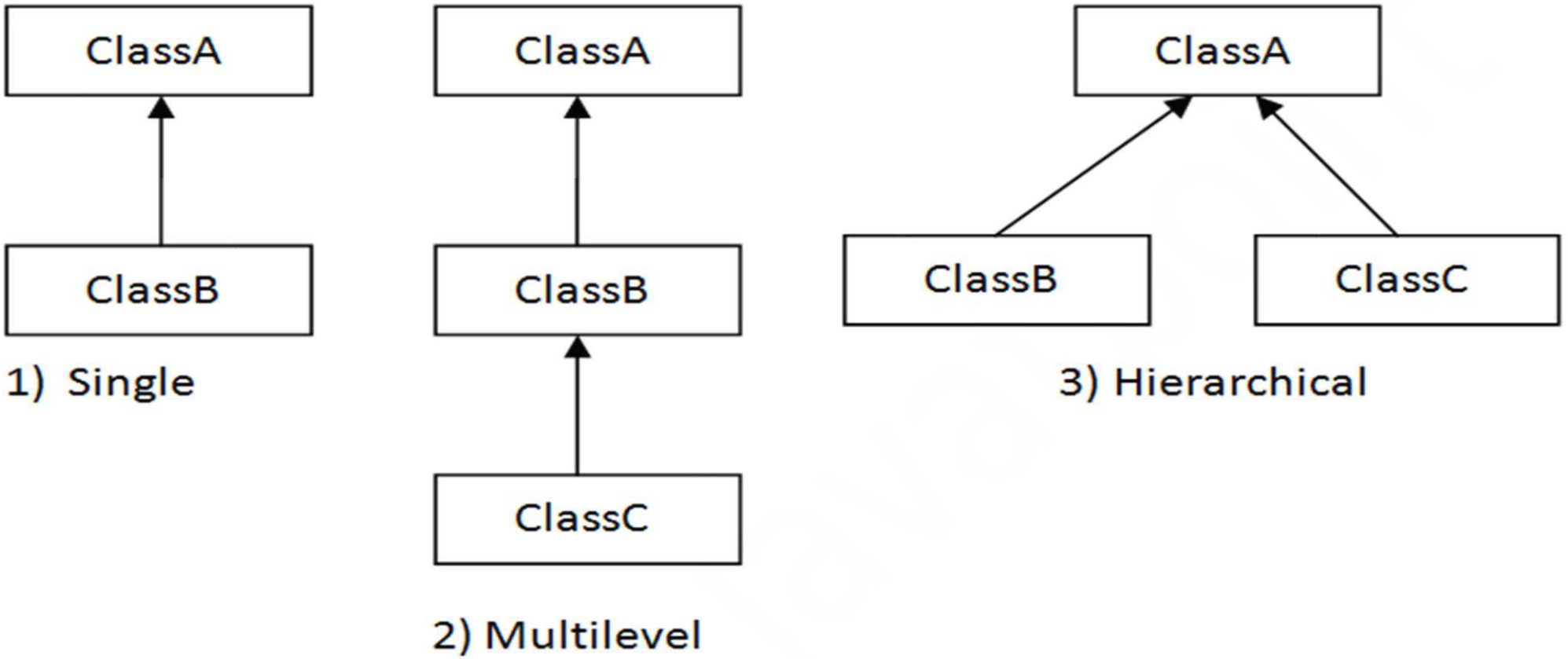


- The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.
- In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

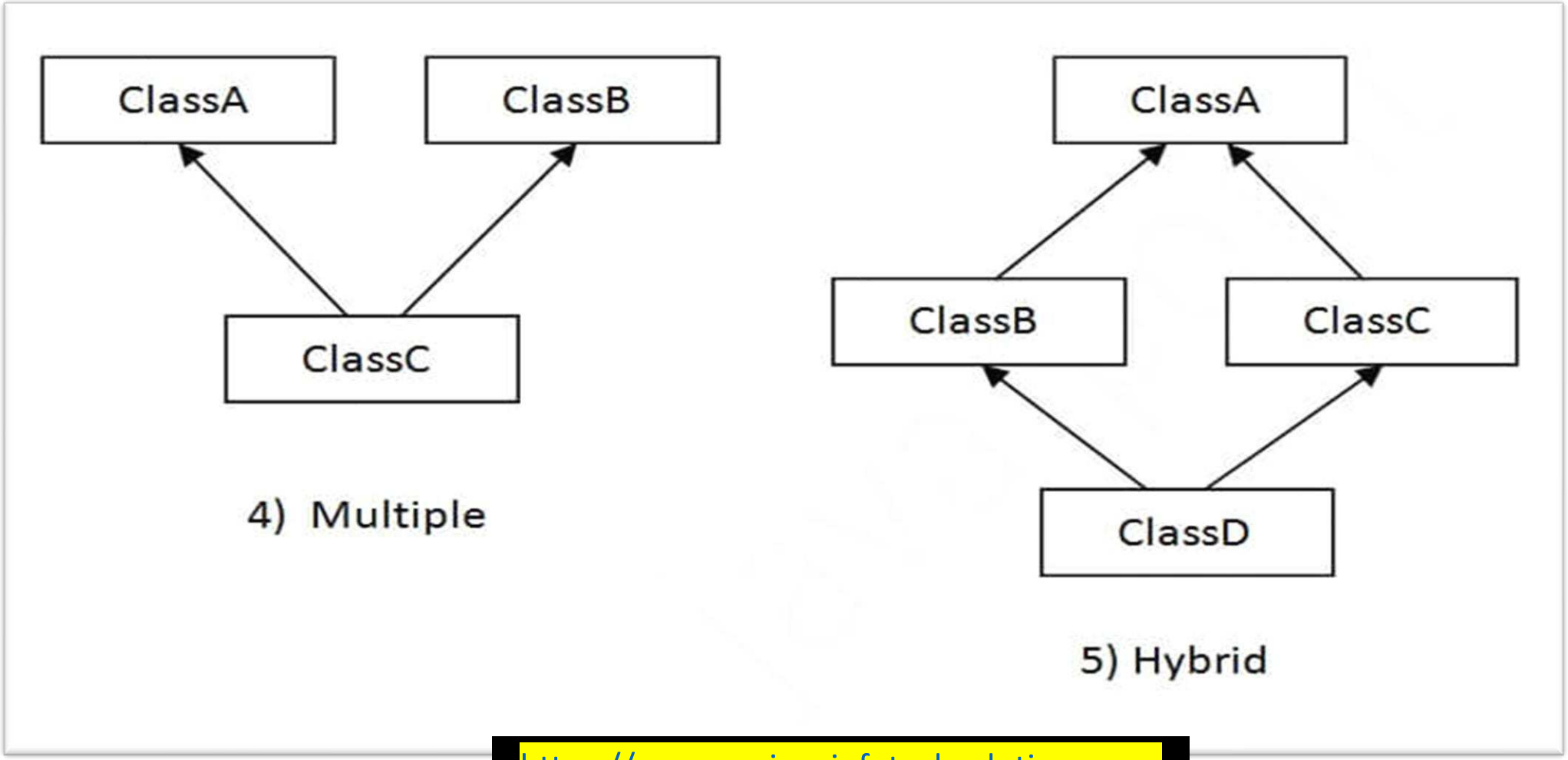
Types of inheritance in java

- On the basis of class, there can be three types of inheritance in java
  - Single,
  - Multilevel and
  - Hierarchical.
- In java programming, multiple and hybrid inheritance are supported through interface only.

Type of Inheritance Concept



Inheritance through Interface



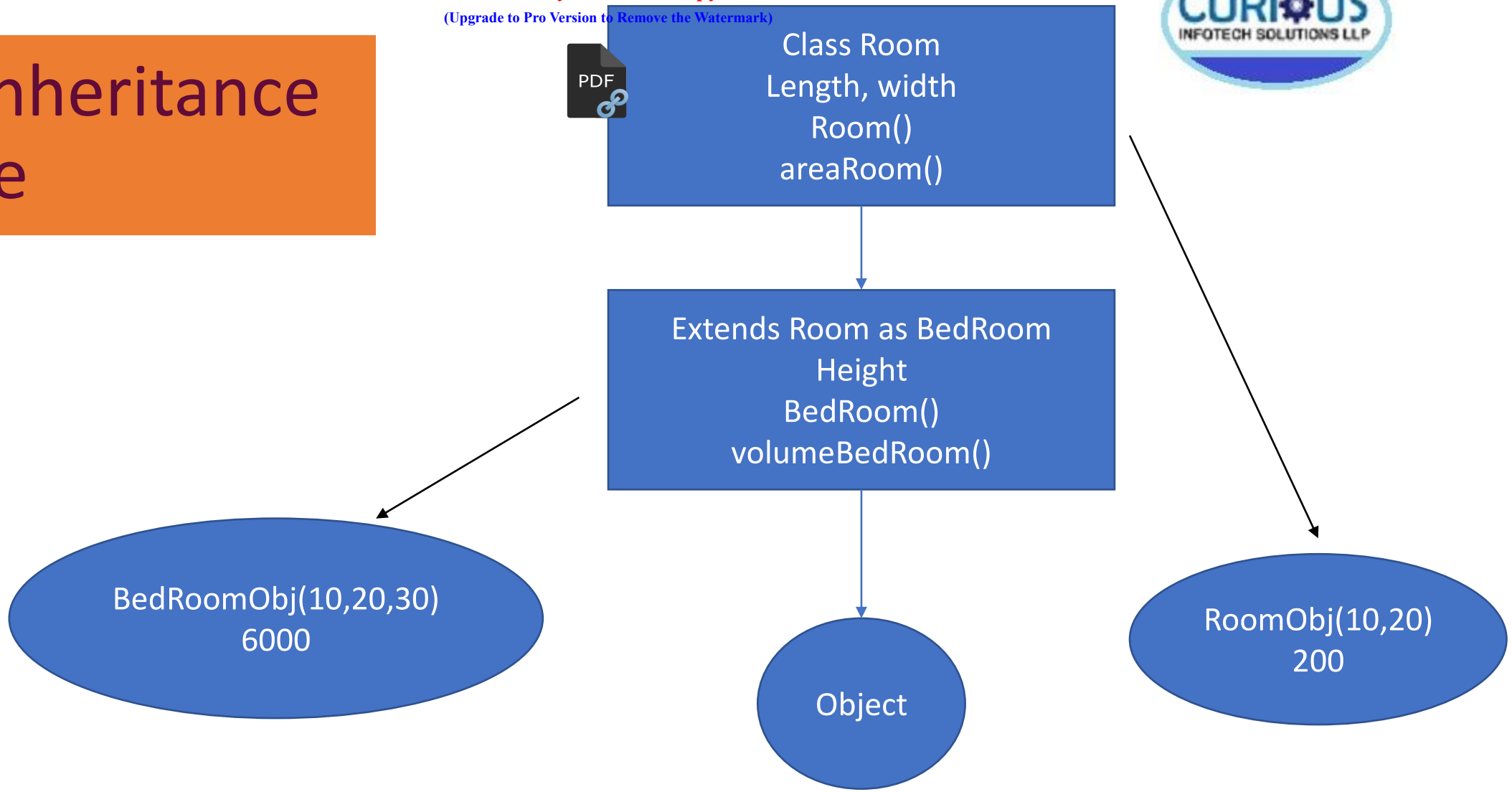
Single Inheritance Concept

When a class inherits another class it is known as a *single inheritance*.

```
class DataType
  {
     void Msg()
        {System.out.println("Datatype are int, float, char...");}
  }
class INTEGER extends DataType
  {
     void integering()
        {System.out.println("Integer Number are 1, 2,0,-5...");}
  }
```

```
class Main
  {
     public static void main(String args[])
     {
        INTEGER N=new INTEGER();
        N.Msg();
        N.integering();
     }
  }
```

Single Inheritance Example



Single Inheritance Example


```
class Room
   {
      int Length, Width;
      Room(int x, int y) {Length=x; Width=y;}
      int araeRoom() {return Length*Width;}
   }
class BedRoom extends Room
   {
      int Height;
      BedRoom( int x, int y, int z)
          {super(x,y); Height=z;}
      int volumeBedRoom()
          {return Length*Width*Height;}
   }
```

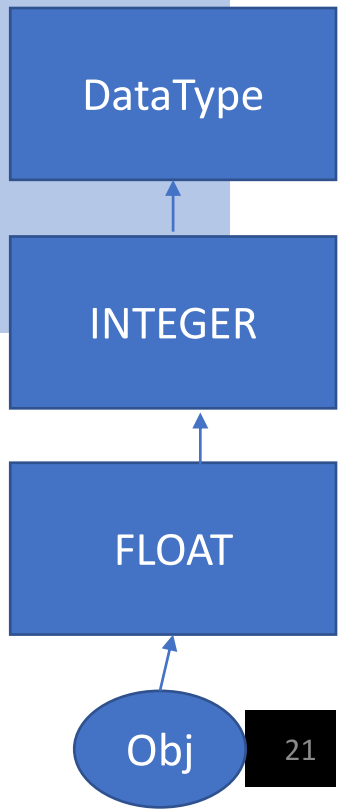
```
class Main
   {
      public static void main (String[] args) {
         BedRoom ObjRoom = new BedRoom(10,20,30);
         //ObjRoom.getBedRoomData(10,20,30);
         int roomArae1 = ObjRoom.araeRoom();
         int roomvolume = ObjRoom.volumeBedRoom();
         System.out.println("Area1 "+ roomArae1);
         System.out.println("Volume1 "+ roomvolume);
      }
   }
```

Multilevel Inheritance Concept

When there is a chain of inheritance, it is known as *multilevel inheritance*.

```
class DataType
   {
      void Msg()
         {System.out.println("Datatype are int, float, char...");}
   }
class INTEGER extends DataType
   {
      void integering()
         {System.out.println("Integer Number are 1, 2,0,-5...");}
   }
class FLOAT extends INTEGER
   {
      void floating()
         {System.out.println("Float Number are 1.5, 2.7, 0.0,-5.8...");}
   }
```

```
class Main
   {
      public static void main(String args[])
      {
         FLOAT N=new FLOAT();
         N.Msg();
         N.integering();
         N.floating();
      }
   }
```



Hierarchical Inheritance Concept

When two or more classes inherits a single class, it is known as *hierarchical inheritance*.

```
class DataType
   {
      void Msg()
         {System.out.println("Datatype are int, float, char...");}
   }
class INTEGER extends DataType
   {
      void integering()
         {System.out.println("Integer Number are 1, 2,0,-5...");}
   }
class FLOAT extends DataType
   {
      void floating()
         {System.out.println("Float Number are 1.5, 2.7, 0.0,-5.8...");}
   }
```

```
class Main
  {
     public static void main(String args[])
     {
        INTEGER INT = new INTEGER();
        FLOAT FLO =new FLOAT();
        INT.Msg();
        FLO.Msg();
        INT.integering();
        FLO.floating();
     }
  }
```