Input Through Keyboard

```
import java.util.*;
class InputKeybord2
{   public static void main(String args[])
{    Scanner sc = new Scanner (System.in);
      System.out.print("Enter roll no: ");
       int x = Integer.parseInt(sc.nextLine());
       System.out.print("Enter Name ");
       String name=sc.nextLine();
        System.out.print("Enter marks ");
         double d=Double.parseDouble(sc.nextLine());
         System.out.println("Roll no ==" + x);
         System.out.println("Name== " + name);
         System.out.println("Marks== " + d);
   }
  }
```

Unit-2

Argument Passing Mechanisms

1. Command Line Argument

Example 1

```
class CommandLineArgument{
public static void main(String args[]){
      System.out.println("Command line argument:"+args[0]);
          }
    }
```

Example 2

```
class CommandLineArgument1{
public static void main(String args[])
{
System.out.println("Command line  arguments are :");
for(int i=0;i<args.length;i++)
 System.out.println(args[i]);
}
}
```

Example 3

```
class CommandLineArgument2{
public static void main(String args[])
{  int a,b,c,d;
System.out.println("Command line  arguments are :");
a=Integer.parseInt(args[0]);
b=Integer.parseInt(args[1]);
c=Integer.parseInt(args[2]);
d=Integer.parseInt(args[3]);
 System.out.println("Average =="+(a+b+c+d)/4.0);
}
}
```

Example 4

```
class CommandLineArgument3{
public static void main(String args[])
{  int a,b,c,d,m;
```

```
a=Integer.parseInt(args[0]);
b=Integer.parseInt(args[1]);
c=Integer.parseInt(args[2]);
d=Integer.parseInt(args[3]);
m=a;
if(b>m)
m=b;
if(c>m)
m=c;
if(d>m)
m=d;
 System.out.println("Largest no is =="+m);
}
}
```

Example5

```
class CommandLineArgument4{
public static void main(String args[])
{  int a[]= new int[10];
    int m,i;
for(i=0;i<args.length; i++)
   a[i]=Integer.parseInt(args[i]);
    m=a[0];
for(i=0;i<args.length; i++)
        if( a[i]>m)
        m=a[i];
```

```
System.out.println("Largest no is =="+m);
}
}
```

2. Call By Value Method Or Pass By Value Method

```
// Primitive types are passed by value.
import java.util.Scanner;
class CallByValue
 {
void Swapping(int x, int y)
        {
                x = x+y;
                y= x-y;
                x=x-y;
        }
}
class CallByValueDemo
{  public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
         System.out.print("Enter first number: ");
         int x = in.nextInt();
         System.out.print("Enter second number: ");
         int y = in.nextInt();
        CallByValue obj = new CallByValue();
        System.out.println("x and y before call: " + x + " " + y);
                obj.Swapping(x, y);
```

```
            System.out.println("x and y after call: " +x+ " " + y);
        }
    }

/*
```

When a primitive type is passed to a method, it is done by use of call-by-value. Objects are implicitly passed by the use of call-by-reference. */

3. Call-by-reference/ Pass by Ref.

```
import java.util.Scanner;
class PassByReference
 {
  int a, b;
   PassByReference(int x, int y)     // Parameterized Constructor
   {
     a = x;
     b= y;
   }
      void Swapping(PassByReference obj)       // Passed an object
     {
        obj.a=obj.a+obj.b;
        obj.b=obj.a-obj.b;
        obj.a=obj.a-obj.b;
      }
  void display()
                          {   System.out.println(a+ " And " + b);     }
}
class PassByReferenceDemo
 {
public static void main(String args[])
 {

        Scanner in = new Scanner(System.in);
```

```
System.out.print("Enter first number: ");
      int a = in.nextInt();
System.out.print("Enter second number: ");
       int b = in.nextInt();
  PassByReference  obj1= new PassByReference(a,b);
System.out.println("a and b before call: ");
       obj1.display();

       obj1.Swapping(obj1);
   System.out.println("a and b after call: ");
      obj1.display();

  }
}
```

4. Returning an object.

```
class ReturningObject
 {
   int a;
  ReturningObject(int i)
       {    a = i;     }
 ReturningObject incrementNo()
       {
       ReturningObject temp = new ReturningObject(a+20);
       return temp;
      }
}
class ReturningObjectDemo
 {    public static void main(String args[])
    {    ReturningObject obj1 = new ReturningObject(2);
         ReturningObject obj2;
     obj2 = obj1.incrementNo();
         System.out.println("obj1.a: " + obj1.a);
         System.out.println("obj2.a: " + obj2.a);
     obj2 = obj2.incrementNo();
        System.out.println("obj2.a after second increase: "+ obj2.a);
   }
}
```

Recursion

Recursion is the process of defining something in terms of itself. As it relates to Java programming, recursion is the attribute that allows a method to call itself. A method that calls itself is said to be recursive.

// Example of recursion. Factorial Computation

```
import java.util.Scanner;
    class FactorialRecursion
  {
        int factorial(int n)      // this is a recursive method
       {
           if(n==1)
              return 1;
           else
            return(factorial(n-1) *n);
       }
  }
class FactorialRecursionDemo
 {
    public static void main(String args[])
  {   FactorialRecursion f = new FactorialRecursion();
       Scanner in = new Scanner(System.in);
       System.out.print("Enter first number: ");
       int n = in.nextInt();
        System.out.println("Factorial value=" + f.factorial(n));
   }
}
```

```
//Recursion   Fibonacci Series
import java.util.Scanner;
class RecursionFibonacci
  {
    static int n1=0,n2=1,n3=0;
    static void displayFibonacci(int count)
     {
        if(count>0)
         {         n3 = n1 + n2;
                   n1 = n2;
                  n2 = n3;
          System.out.print(" "+n3);
          displayFibonacci(count-1);
     }
 }
}
class RecursionFibonacciDemo
   {      public static void main(String args[])
          {    Scanner in = new Scanner(System.in);
              System.out.print("Enter number of Terms: ");
               int count = in.nextInt();
System.out.print(RecursionFibonacci.n1+" "+RecursionFibonacci.n2);
RecursionFibonacci.displayFibonacci(count-2);
 }
}
```

// Constructor Overloading Unit -2

```
class BoxCO
{     double width, height, depth;
      BoxCO(double w, double h, double d)
   {  width = w;
      height = h;
      depth = d;
   }
    BoxCO(double len)
   {    width = height = depth = len;       }
   double volume()
   {           return width * height * depth;       }
}
 class BoxDemo
{       public static void main(String args[])
   {            BoxCO mybox = new BoxCO(10, 20, 15);
                BoxCO mycube = new BoxCO(7);
      double vol;
                 vol = mybox.volume();
      System.out.println(" Volume of mybox1 is " + vol);
        vol = mycube.volume();
      System.out.println(" Volume of mycube is " + vol);
   }
}
```