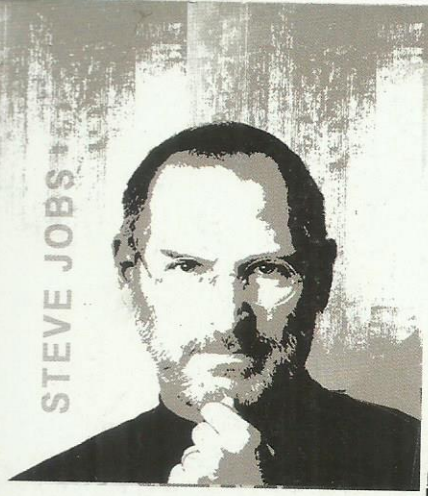
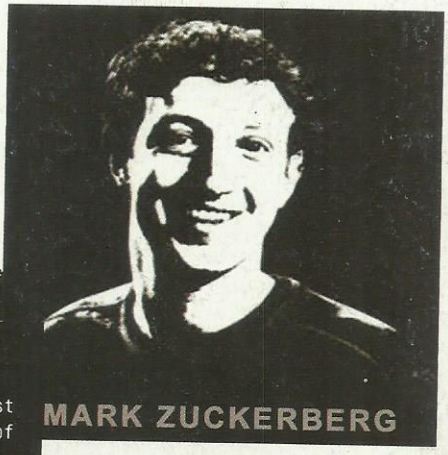
globsyn skills
good skills • good jobs

N·S·D·C National Skill Development Corporation



Steve Jobs was born in San Francisco, California, on February 24, 1955, to two University of Wisconsin graduate students who gave him up for adoption. Smart but directionless, Jobs experimented with different pursuits before starting Apple Computers with Steve Wozniak in 1976.
Apple's revolutionary products, which include the iPod, iPhone and iPad, are now seen as dictating the evolution of modern technology. He died in 2011.

Entrepreneur **BILL GATES**, born on October 28, 1955, in Seattle, Washington, began to show an interest in computer programming at the age of 13. Through technological innovation, keen business strategy, and aggressive competitive tactics, he and his partner Paul Allen built the world's largest software business, Microsoft. In the process, Bill Gates became one of the richest men in the world.



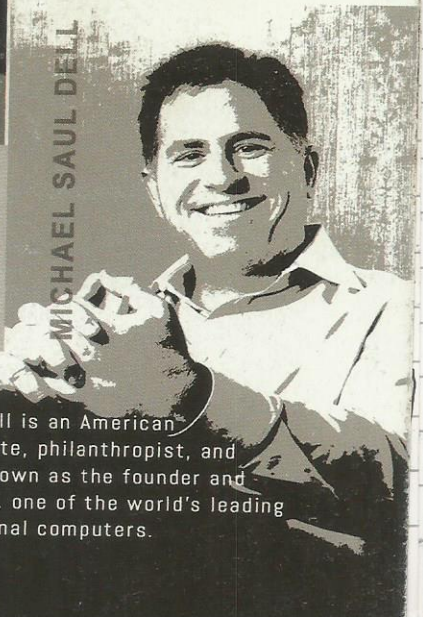
MARK ZUCKERBERG

Born on May 14, 1984, in DobFerry, New York, **Mark Zuckerberg** co-founded the social-networking website Facebook out of his college dorm room. He left Harvard after his sophomore year to concentrate on the site, the user base of which has grown to more than 250 million people, making Zuckerberg a billionaire. The birth of Facebook was recently portrayed in the film The Social Network.

LARRY PAGE

Page is an American computer scientist and Internet entrepreneur who, with Sergey Brin, is best known as the co-founder of Google. On April 4, 2011, he took on the role of chief executive officer of Google, replacing Eric Schmidt.

Business person, Lawrence Joseph "Larry" Ellison is an American business magnate, co-founder and chief executive of Oracle Corporation, an enterprise software company. He also races sailboats, flies planes, and plays tennis and guitar.



Michael Saul Dell is an American business magnate, philanthropist, and author. He is known as the founder and CEO of Dell Inc., one of the world's leading sellers of personal computers.

13.06.13

How to take i/p ~~through~~ from the user through the keyboard

```
import java.util.*;
```

It reads integer value - nextInt.

Creating object in a scanner class :

① `Scanner sc = new Scanner(System.in);`

[System defined class]

② For buffer Reader - `import java.io.*;`

```
BufferedReader br = new BufferedReader;
    (new InputStreamReader(System.in));
```

getting i/p :

```
void input() throws Exception   (have to write in main)
{
    System.out.println("Enter any intno");
    value = Integer.parseInt(br.readLine());
}
```

③ command line argument.

Develope a java program to create a class to keep track of how many objects an unique id no (starting from 0) sequentially for each of its objects. Display total no. of objects & the id no. of each objects.

```
package test;
import java.util.*;
public class Avg1
{
    private int math;
    private int phy;

package test;
class Obj
{
    private static int count;
    private int id;
    Obj()
    {
        id = count;
        count++;
    }
```

```
void create Id ()
{
   id = count;
   count++;
}
  void display Id ()
  {
    System.out.println("Id is" + id);
  }
    void display Count()
    {
      System.out.println("Total No of object" +
                                        count);
    }
      public class Objectcount
      {
        public static void main(String [] args)
        {
          obj ob1 = new obj();
          obj ob2 = new obj();
          obj ob3 = new obj();
```

```
*/  ob1.createdId();
    ob2.createdId();
    ob3.createdId();

*/  ob3.displayId();
    ob1.displayId();
    ob2.displayId();

    ob1.displayCount();
}
```

C++

It has only compiler

Extension code –

FILE NAME . C

↓ (compile)

FILE NAME . obj (object file)

↓

- .exe
- .bak (we get last perfect running source code)

in Java

FILE NAME.java
↓ (compile)
.class
↓ JRE
Execute

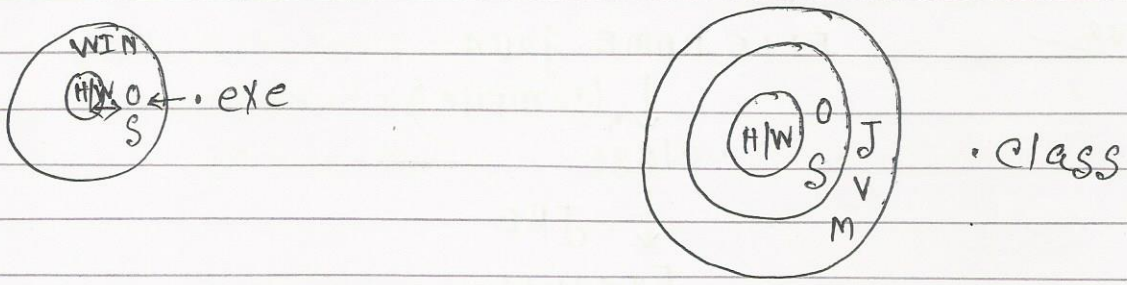
Java
compiler — Interpreter (less memory is required)
(large memory is required)

For compiling - use JDK
For executing - use JRE

JDK - Java Development Kit.
JRE - Java Run Time Environment.

- The containts, which are present in .class, are called 'Byte code'.
  Each instruction's length is 1 byte, so we called this byte code.

- Jvm – Java virtual machine. It reads the byte code. It looks what type of OS is present.



- Jvm takes [.class] as a i/p. Every Jvm takes same byte code as a i/p.
- Different Jvm is used in different OS.
  Jvm is a platform dependent for different different OS.

(**) Why Java is a platform independent language?
Because of Jvm. Jvm is a seperate Software & is dependent.

N.B: [When we run java, jvm is also installed]

JIT – Just in time
Advantage – Big program loads in small memory
Disadvantage – very slow.

(*) SandBox – Inside the Jvm the component is called sandbox. It is used for sequre.

In JAVA, 4 data types are present –

	bit
i) byte	8
ii) short	16
iii) int	32
iv) long	64

N.B - In Java, all data types are signed. There were no unsigned type.

Automatic type conversion / promotion / implicit type-casting : In this, compiler automatically converts int to float data type.

Without this, we will loss information or data at the time of conversion.

float → int

```
int x = 6;
float y = 6.25;
float z = x + y;
          ↙   ↘
        int    float.
```

int k = int(x+y);
called 'explicit' type-casting.

float z = x + y = 6 + 6.25
= 12.25
= float.

✓ In java, use only two data types.
i) float
ii) double.

'C' follows ASCII character set. (Char - 1 byte means 1 bit)

* 'Java' generally follow 'unicode' character code.
↘ char - 2 byte.
(using 16 bit).

For every alphabet gives unique no.
Another data type is - `boolean b1;`
↓
It can only store

** boolean variable takes only 1 bit.

b1 = { true / false

```
int x = -5;
if (x)
  Pf("A");
else
  Pf("B");
```

In C: If there is no condition is given, then compiler treats as `x != 0`

In Java: (Explicitely) put the condition in java, otherwise it gives compile time error.

```
boolean b = true;
if (b)
{

}
```

Here b has already given. so it will print always true

In C :

```
while (b)
{
    (infinite loop)
}
```

- Java naming conversion -

i) When you give any class met first char is capital.
ii) Java is case sensitive.
iii) When you write any variable name, try to give proper name of that variable.
iv) In function, all char is small.

```
Student ram = new Student();
```

(class name) → Student
(constructor) → Student()
ram → (object name)
new Student() → (Run time memory allocation)

```
Class Student
{ int roll;
  int math;
  int phy;
  void f1() { }
  int f2() { }
}
```

user defined constructor. As we write in pgrm.

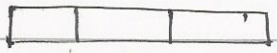
```
Student()
{ roll = 10;
  math = 80
  phy = 95
}
```

10	80	95
roll	m.	p.

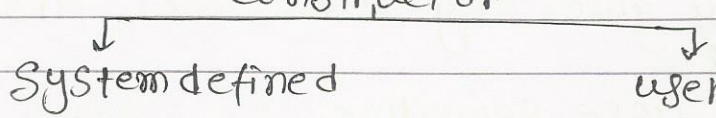
* In Java - all memory allocation are Dynamic.

* 'Object' is nothing but a 'variable name'.

* 'new' is responsible for allocating memory i.e.,



Constructor : If Class name & function name are same, then it is called as 'constructor.'



cons. initialized the value as like this- [0|0|0]

The default return type in 'c' is int. But in 'Java' every function should have return type.

[N.B:] Constructor has no return type.

Why in OOPs their is no concept of garbage value)

Because of the constructor.

If constructor is not executed, then what will happened)

If userdefined is there then System defined cons. is not required.

* Java pgrm is collection of 'classes.'
* String means char. type array.
* String is a system defined class.

1. Class First
{
public static void main (String a[])
{
System.out.println ("WELCOME");
}
};

~~In Terminal = cd~~

2/*/ pgrm of a factorial no.

Class Factorial
{
private int result;
private int value;
Factorial ()
{
result = 1;
value = 5;
}
void cal ()
{
for (int i = 1; i <= value; i++)

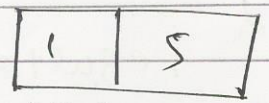
```
    result= result * i;
  }
 }
  void display()
  {
   System.out.println(" Factorial is " + result);
  }
  }

  class Main
  {
   public static void main(String a[])
   {
    Factorial f1 = new Factorial();
    f1.cal();
    f1.display();
   }
  }
```

object

(class name)

(constructor)



3. */ pgrm of Fibonacciseries.

```
public class Fibo
{
   private int a;
   private int b;
   private int c;
   private int n;
  Fibo ()
  {
     a=0;
     b=1;
     n=5;
  }
   void cal ()
   {
      System.out.println(a);
      System.out.println(b);
        for( i=1 ; i<=n ; i++)
      {
          c=a+b;
          a=b;
          b=c;
        System.out.print(c);
      }
```

```
public class main
{
    public static void main (String [] args)
    {
        Fibo p1 = new Fibo();
        p1.cal();
    }
}
```

In Java - i) && → shortCKt AND
(this is same as 'logical AND' in C)
ii) & → Logical AND.
iii) | → Bitwise AND.

In Logical AND - if the 1st condition is false it checks the 2nd condition.

If x & y → read as 'bitwise operator'

Features of OOPS:

i) Dynamic initialization
ii) Modularity
iii) Data Encaptulation
iv) Data Hiding
v) polymorphism
vi) Abstruction

ii) **Modularity**: It divided the program in different different module.

iii) **Encaptulation**: It means to put the data with something. i.e., Data & behaviour(function) groups together & encaptulated & put in a class.

(*) in main - only object & function.

iv) **Data Hiding**: In Data Hiding, every function should not get all values.
For Hiding we always use 'private.'
Example: 'Home' is a class but 'my Home' is a object.

```
class A
{
    private int x;
}
main ()
{
}
f1()
{
}
```

Keyword.

before variable, it will hide the data from the outside of the class.

[To access the value we can use function.]

v) Polymorphism

- Function overloading
- Function overwritting

(*)(*) Function overloading : It has same function name with different arguments.

Method: First it checks, no. of arguments. If the arguments are same then it check data type.

i) `void fact (int x)` ✓
`{ }`
`void fact (int x, int y)`
`{ }`

(fact — function name)

ii) `void fact (int x, float y)`
`{ }`
`void fact (float x, int y)`
`{ }`

`int fact (int K)`

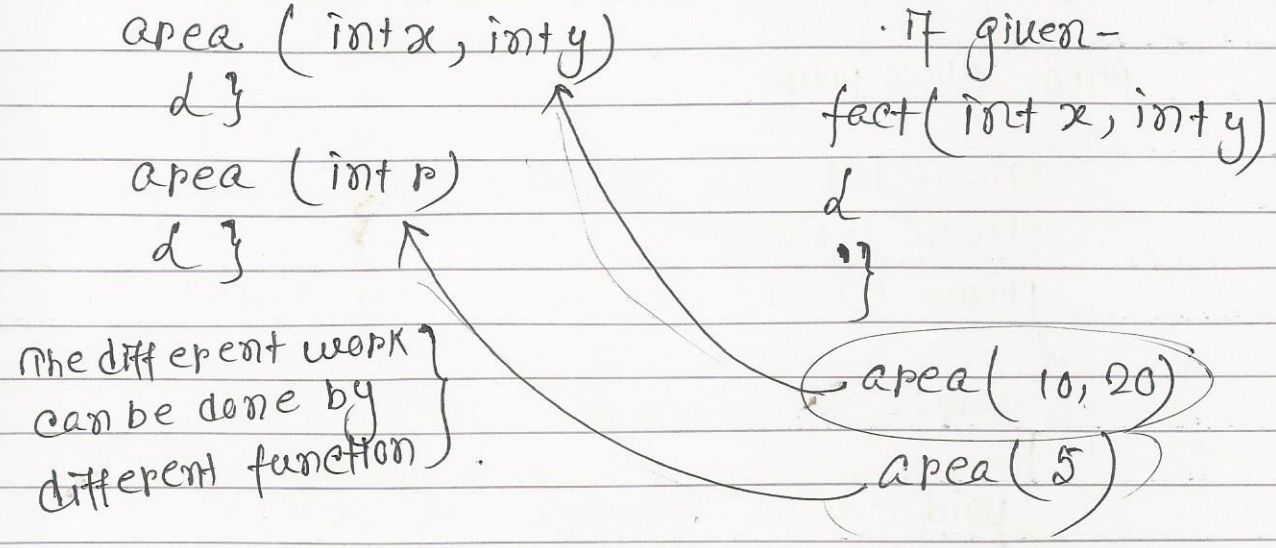
↓

It will give compile time

`void fact (float K)`
`{ }`

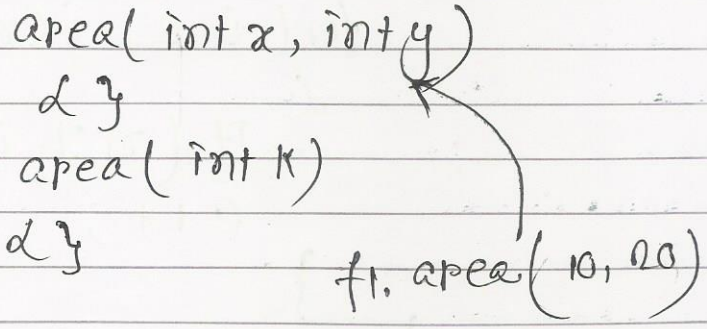
Why this is called overloading?
Because it has multiple body.

- Advantage : We use different different function for diff. diff. time.



i) Static Mapping.
ii) Dynamic Mapping.

Static Mapping :



In static mapping, first check arguments. Then the function call its same function body. at compile time.

Abstruction: is another features.

From main, the function becomes abstruct.

4. */ Wap of a prime no.

```
public class prim
{
    private int n;
    private int c;
    prime()
    {
        n;
    }
    void cal()
    {
        System.out.println(n);
        c=0;
        for(i=2; i<n; i++)
        {
            if(n%i==0)
            c++;
        }
    }
}
```

```
void display ()
    {
        if (c==0)
            System.out.print("Number is prime");
        else
            System.out.print("No. is not prime");
    }
}

Public class Main
{
  public static void main(String[] args)
  {
      Prime p1 = new Prime();
          p1.cal();
          p1.display();
  }
}.
```

5. Develope a Java pgrm to create a class student get marks in three subjects & calculate appropriate class.

```
import java.util.*;
public class Avg1
{
    private int math;
    private int phy;
    private int chem;
    private int sum;
    private int total; private int double avg;
    Scanner sc = new Scanner(System.in);
    void input()
    {
        math = sc.nextInt();
        phy = sc.nextInt();
        chem = sc.nextInt();
    }
    void cal()
    {
        sum = math + phy + chem;
        Avg = sum/3;
    }
    void display()
    {
        System.out.print(" Sum = " + sum);
        System.out.print(" Avg = " + avg);
```

```
public class Main
{
    public static void main (String [ ] args)
    {
        Avg1 p1 = new Avg1 ();
            p1. input ();
            p1. cal ();
            p1. display ();
    }
}
```

- Command lined Argument : (user i/p)

6. */ Wap of a factorial no.

```
Class Fact
    {
        private int result;
        private int value;
        Fact (int x)
        {
            value = x;
            result = 1;
        }
```

```
void cal ()
{
  for (int i=1; i<=value; i++)
  {
    result = result * i;
  }
}
  void display ()
  {
    System.out.println (" The factorial is " +
                                        result);
  }
}
Class Factorial
{
  public static void main (String a[])
  {
    int number = Integer.parseInt (a[0]);
    Fact f1 = new Fact (number);
        f1.calculation ();
        f1.display ();
     Fact f3 = new Fact ();
    int number1 = new Integer.parseInt (a[1]);
     Fact f2 = new Fact (number1);
         f2.cal ();
    }    f2.display ();
```

: Array Creation in a Class :

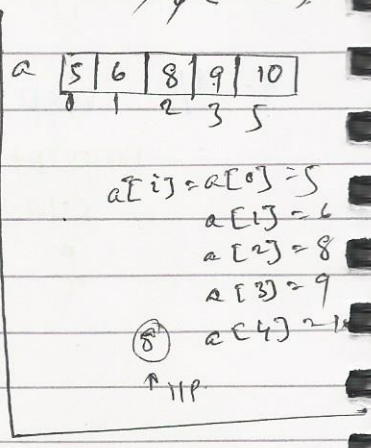
In Java, every array is system defined.

Syntax - `arrayname.length`

7. */ Wap of a Array creation & searching a value.

```
import java.util.Scanner;
Class Array Creation
{
    private int a[];
    Private int value;
    Private boolean flag;
    private int p;
    private int asize;
  Scanner Sc = new Scanner(System.in);
    Array Creation ()
    {
      flag = false;
    }
      void arraysizeInput ()
    {
      System.out.println("Enter the array
                          size");
        asize = Sc.nextInt();
    }
      void array Creation ()
      {
        a= new int[asize];
      }
```

```
void arrayValueInput()
{
    System.out.println("Enter values");
    for(int i=0; i<asize; i++)
    {
        a[i] = sc.nextInt();
    }
void arrayDisplay()
{
    for(int i=0; i<a.length; i++)
    {
        System.out.println("Array is " + a[i]);
    }
}
void searchValueInput()
{
    System.out.println("Enter the no. for search");
        value = sc.nextInt();
}
void search()
{
    for(int i=0; i<asize; i++)
    {
        if(a[i] == value)
        {
            flag=true;
```



⑨

```
        p = i;
        break;
      }
    }
  }
}
```

a[0] = = 9
a[1] = = 9 x
a[2] = = 9 x
a[3] = = 9 ✓
p = 3;

x a[3]
p < a[5]
True
flag = false

```
* public class Sequential search
  {
    public static void main( String [] args)
    {
      Array Creation a1 = new Array Creation ();
        a1. array Size Input ();
        a1. array Creation ();
        a1. array value Input ();
        a1. array Display ();
        a1. Search value Input ();
        a1. Search ();
        a1. Search Result Display ();
    }
  }

* void Search Result Display ()
  {
    if (flag)
      System.out.println(" The item is found ");
    else
      System.out.println (" Not found ");
  }
```

8. */ adding of two number.

```
import java.util.*;
class Add
{
    private int a;
    private int b;
    private int c;
   Scanner sc = new Scanner(System.in);
   void Input ()
   {
     System.out.println (" Enter values of
                          a and b ");
       a= sc.nextInt ();
       b= sc.nextInt ();
   }
    void cal ()
    {
       c = a+b;
    }
     void display ()
    {
      System.out.println (" c ");
    }
     public class Main
    { public static void main(String[] args)
```

```
{
    Add d1 = new Add ();
    d1.input ();
    d1.cal ();
    d1.display ();
  }
}
```

9. : Industry Pattern Program :

Src → new package (Name add) → Finish
Class → (Name AdditionData)

```
public class AdditionData
{
    Private double firstvalue;
    private double secondvalue;
    private double result;
}
```

Right click on body
↳ Source → Generate Getters & Setters
Select i) 1st value
ii) result
iii) 2nd value
↳ OK.

```
public double getFirstvalue ()
{
    return firstvalue;
}
public void setFirstvalue (double f1)
{
    firstvalue = f1;
}
```

```
public void getSecondValue()
  {
    return secondValue;
  }
   public void setSecondValue(double s1)
   {
     secondValue = s1;
   }
    public double getResult()
    {
      return result;
    }
     public void setResult(double r1)
     {
        result = r1;
     }
     }
```

↳ Class → (Name- AdditionModule).

```
public class AdditionModule
  {
    Scanner sc = new Scanner(System.in);
    public AdditionData obj = new AdditionData();
    ✓ void dataInput()
     {
       System.out.print("Enter 1st value");
        obj.setFirstValue(sc.nextDouble());
```

```
System.out.println(" Enter 2nd value ");
 obj.setSecondvalue(Se.nextDouble());
}
void cal()
{
   obj.setResult(obj.getFirstvalue() +
                 obj.getSecondvalue());
}
void display()
{
   System.out.println(" Result is " + Obj.getResult
                                          ());
}
```

↳ class → (Name → Add Main)

```
public class Add Main
{
   public Static void main (String [] args)
   {
      Addition Module ob1 = new Addition Module();
      {
         ob1.inputdata();
         ob1.cal();
         ob1.display();
      }
   }
```

** To get the value, use (Set Function)
To assign the value, use (get Function).

* 'All variables' should be kept in one class &
* 'behaviour' should be kept in another class.

parameterwise constructor :

Constructor with one argument called 'parameterwise constructor'.

* Here use the keyword is ↳ "this".

(To call a constructor from another ↲ constructor)

N.B: [If system defined constructor is there, system defined cons. don't work.

```
Class Fact
  {
    Private int result;
    Private int value;
    Fact ()
   {
     System.out.println("Hello");
   }
    Fact (int x)
    {
      this ();
      System.out.println("Hi");
        value = x;
        result = 1;
    }
```

```
void cal ()
  {
    for(int i=1 ; i <= value ; i++)
    {
      result = result * i;
    }
  }
    void display ()
    {
      System.out.println(" The factorial is " +
                                    result);
    }
  }
    class Factorial
    {
      public static void main(String[] args)
      {
        int number = Integer.parseInt(a[0]);
         Fact f1 = new Fact (number);
          f1.cal();
          f1.display ();
      }
    }
```

Generices : variable name same but datatype is different.

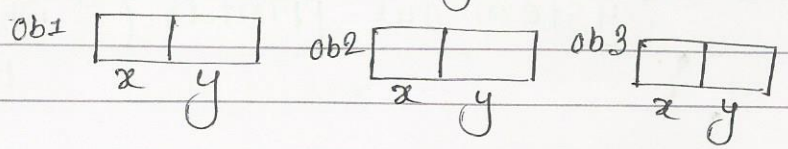
```
class <T>
{
  T x;
  T y;
}
```

we create same obj. of this class.

ob1 ob2 ob3

<integer> <String> <double>

Then it is a int type class & memory will allocated.




```
package test
  class D<T>
{
  private T firstvalue;
  private T secondvalue;
    D(T f1, T f2)
  {
     first value = f1 ;
     Second value = f2;
  }
    void display ()
  {
    System.out.println(" First value is " + first value);
    System.out.println(" Second value is " + second value);
  }
}
```

```
public class Gen1
{
  public static void main(String [ ] args)
  {
    D<Integer> ob1 = new D<Integer> (5,10);
    D<String> ob2 = new D<String> ("HELLO","hi");
      ob1.display ( );
      ob2.display ( );
  }
}
```

- **ArrayList class :** It is a system defined class & it is in inside the [java.util.*].

Advantage : i) efficient memory handling mechanism
ii) don't need to specific array size.

```
import java.util.ArrayList;
  class java.util.ArrayList;
    class A1
    {
      ArrayList <Integer> alist = new ArrayList
                                  <Integer> ();
      void add data ()
      {
        int v = 50;
        for( int i=0; i<5 ; i++)
        {
          alist.add (v);
          v++ ;
```

```
void f1 ()
{
    alist.add(1,500);
    // alist.remove(2); (1st it is off).
}
void displayData()
{
    for(int i=0; i<alist.size(); i++)
    {
        System.out.println(alist.get(i));
    }
}
public class ArrayListDemo
{
    public static void main(String[] args)
    {
        A1.p1 = new A1();
        p1.addData();
        // p1.f1(); (1st it is off)
        p1.display();
    }
}
```

: Date Format :

```
import java.text.DateFormat;
import java.util.Date;
 class DateShow
  {
    void f1 ()
  {
    DateFormat dft;          (DateFormat.Long)
     String S;
     Date date_ob = new Date ();
   dft = DateFormat.getDateInstance("DateFormat.
                                    SHORT");
     S= dft.format( date_ob );
     System.out.println(S);
   }
}
public class DateFormat 1
  {
    public static void main (String[] args);
   {
     DateShow p1 = new DateShow ();
          ob1.f1();
    }
  }
```

: Interface :

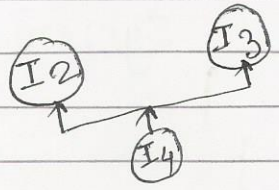
* Interface is nothing but a block of code, where we can write only function declearation.

* We should use 'implement' Keyword when interface is attached with a class.

[Here all data are given but we don't Know it how to work]

```
Interface I2
{
   void f1();
}
Interface I3 {
{
   void f2();
}
Interface I4 extends I2, I3 →
{
   void f3();
}
   A11 implements I4
   {
      public void f1()
      {
         System.out.println("F1");
      }
      public void f2()
      {
         System.out.println("F2");
```

- Multiple inheritance is possible in interface



```
public void f3()
{
    System.out.println("F3");
}
}
public class Interface main
{
    public static void main(String[] args)
    {
        A11 ob1 = new A11();
        ob1.f1();
        ob1.f2();
        ob1.f3();
    }
}
```

* Access precifier in JAVA-
  i) public ii) private iii) protected
  iv) default.

Access modifier- i) Final ii) Static.

1. Difference between public & default?
   If we write ~~package~~ public before the package or folder name, then it will be accessed by different folder. But if we not write public before package name then it will becomes default & it will be not accessed by other folder.

- **Protected:-** protected is only used in child class { it is used for accessing the variable from different package.
- **private:-** private is only used in same class. ~~other~~ variables In this class, the ~~functions~~ a will be accessed by the same class's functions not other class's functions.
- **Default:-** Default is only used in same package.
- In a single [.java file] only one public class is possible.

package 1.

public
[if it is public, then it will be accessed by diff folder. But if it is default then it will be accessed only its same package

```
Class A
{
  private int x;
  public int y;
  Protected int z;
        int K;
  f1()
  {
    x
    y
    K
  }
}
```

```
Class B extends A
{
  f2()
  {
    y
    z
    K
  }
}
```

ob1.y
ob1.z
ob1.K

```
Class C
{
  A ob1 = new A();
  f3()
  {
  }
```

package 2

```
class D extends p1 A
{
    f4()
    {
        y
        z
    }
}

class E
{
    p1.A ob2 = new p1.A()
    ob2.y ;
}
```

- In main always use public.

* For sharing one .java file by different package, we must write public before
  i) class name
  ii) function name () { }

* If we write public before function name but not write before class name, then obj. is not create & the function will not be called.

Abstract class :

i) Function with no body called 'abstract' function.

ii) If abstract Function is present in a class, then this class becomes an 'abstract class.'

iii) 'Abstract' keyword should be written before function & class name.

iv) Abstract class contain normal method.

v) Object of an abstract class not be created but reference should be created.

vi) To access the normal method of an abstract class we should create the child & should override all abstract method of the parent.

vii) If child class is not becomes an abstract class, creating child class object to access normal methods of an abstract class.

: Enum :

```
package test
 class A15City
{
  void f1()
  {
    enum City { KOL, Delhi } { KOLKATA, DELHI, MUMBAI,
                               CHENNAI }
    void f1()
    {
      city C = city.KOL; C C1 = C.KOLKATA;
      System.out.println(C1);
    }
  }
}
  public class Enum Example
  {
     public static void main(String[] args)
  { City
    A15 Ob1 = new City A15();
        Ob1.f1();
  }
  }
```

- Enum is a user defined type, class.

Spliting of String :

"Split()" String into parts & returns an array containing result.

(read)

```
String S = "E1 : John : paris";
String t[] = new string [3];
t = S.split (" : ");
for (String x : t)
System.out.println (x);
```

t[3]

E1	0
Jo	1
Pa	2

OUTPUT :
E1
John
paris

operate string class :

```
Public class ~~String~~ StringOperation1
{
  public static void main (String args[])
  {
    String S1 = new String ("~~Hello~~ HELLO");
    String S2 = new String ("Hello");
    String S3 = "Hi";
  System.out.println (S3.length ());
  System.out.println (S1.toUpperCase ());
  ~~S1.equals (S2)~~
```

```
if ( s1 == s2)
{
  System.out.println( "AA");
}
 else
 {
   System.out.println( "BB");
 }
System.out.println (s1.substring (2));
System.out.println (ss);
System.out.println (s1.substring(1,4));
  char a = s1.charAt(2);
   System.out.println (a);
   if( s1.equalsIgnoreCase(s2))
    {
     S.O.P ("AA");
    }
    else
     {
       S.O.P ("BB");
     }
   if( s1.equals(s2))
   {
     S.O.P("AA");
   }
   else
     {
```

```
S.O.P( s1.compareTo(s2));
}
}
```

/* pgrm of a 'array' and 'searching' */.

```
Package test;
public class Array
{
   private int a[] = new int[5];
   private int num;
   private int i;
   Scanner sc = new Scanner(System.in);
   void input()
   {
      S.O.P("Enter five values of array");
       for(int i=0; i<5; i++)
        {
          a[i] = sc.nextInt();
        }
        S.O.P("Enter a number");
          num = sc.nextInt();
   }
   void display()
    {
       for(int i=0; i<5; i++)
        S.O.P("a[i]");
    }
    void search()
     {
        int c=0;
       for(i=0; i<5; i++)
       {
          if(num==a[i])
```

```
{
    c++;
    break;
  }
}

    if (c==1)
      S.O.P(" MATCH FOUND at position" + i);
    else
      S.O.P(" Number not found");
  }
}

  Class main
    {
      public static void main(String args[])
      {
        Array obj = new Array();
        obj. arrayCreate();
        Obj. input();
        Obj. search();
        obj. display();
      }
    }
```

Through and Throughs :

```
package test;
class ExceptionHandleing2
{
    void f1() throws @Exception
    {
    // NullPointerException np = new NullPointer
                                 Exception();
    // through np;
        int z = 5/0;
    }
}
class C
{
  public static void main(String args[])
  {
    try
    {
      ExceptionHandeling2 obj = new ExceptionHandeling2();
       obj.f1();
    }catch(Exception e)
    {
      S.O.P("e.toString());
    }
      S.O.P ("HELLO");
  }
}
```

user defined Exception Class :

```
Class Excep extends Exception
  {
    private String s;
     Excep(String s1)
      {
        s = s1;
      }
      String info()
      {
        return s;
      }
    }
Class AB
   {
     void f1() throws Excep
      {
        throw new Excep("user defined");
      }
   }
 Class main
  {
    public static void main(String args[])
    {
      try
      {
        S.O.P("aaa");
       // throw new Excep("user defined"); //(obj of
                                          user def class)
```

```
        AB Ob1 = new AB ();
         Ob1.f1 ();
    } catch (Except e)
    {
        S.O.P (" e.info ());
    }
}
}
```