

Example :

Lecture - 1

23/11/21

①

```

class Demo10
{
    public static void main (String args [])
    {
        byte x = 10;
        byte y = 20;
        byte z = x+y;
        System.out.println(z);
    }
}

```

अवलोकन : incompatible types : possible loss of conversion
from int to byte.

यदि कृति में किसी फ्रैम में arithmetic operation perform करते हैं, तो output इस formula के according आता है।

Formula : max (type of x, type of y, int.)

Example :

```

class Demo10
{
    public static void main (String args [])
    {
        byte x = 10;
        long y = 20;
    }
}

```

```
byte z = x+y;
```

```
System.out.println(z);
```

error: incompatible type: possible lossy conversion
from long to byte

byte → short → int → long → float → double

Four points of type casting:

- i) A way of converting one data type to another data type is known as type casting.
- ii) A way of converting one object to another object is also known as type casting.
- iii) we can not convert one data type to object or one object to data type with the help of type casting.
- iv) if we want to convert data type to object or object to data type. then we have to use wrapper classes.

String is not a data type. it is a class.

variable are corresponding to data type.

Example:

```
class Demo10
{
    public static void main (String args[])
    {
        String s1 = "10";
        int y = s1;
        System.out.println (s1);
        System.out.println (y);
    }
}
```

error: incompatible type: string can not be converted to
int

```
int y = s1;
```

Example:

```
class Demo10
{
    public static void main (String args[])
    {
        int x = 0;
        int x = 10;
        System.out.println (x);
    }
}
```

error: variable x is already defined in method main (

```
String [] )
```

```
int x = 10;
```

- # java does not have garbage value.
- # It is compulsory to assign a value in variable before using that variable in java programming.

Example :

```
class Demo11
{
    public static void main (String [] args)
    {
        int x;
        System.out.println(x);
    }
}
```

error : variable x might not have been initialized.
System.out.println(x);
 ^

Example :

```
class Demo11
{
    public static void main (String args [])
    {
        int x= 11;
        System.out.println(y);
    }
}
```

error : can not find symbol
Symbol : variable y
Location : class Demo11

Example :-

```
class Demo12
{
    public static void main (String args[])
    {
        boolean x = 1;
        System.out.println(x);
    }
}
```

Error: incompatible type: int can not be converted to
boolean

Binary of -ve value :-

Binary of (-5) :-

- Step 1: $(5)_0 = (101)_2$

R	5		1
2	2		
2	1		
0	0		

- Step 2: byte $x = -5$ means 8 bit

0 0 0 0 0 1 0 1

- Step 3: 1's complement and 2's complement

$$\begin{array}{r}
 \text{1's} & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
 & + & & & & & & \\
 \hline
 \text{2's} & 1 & 1 & 1 & 1 & 0 & 1 & 1
 \end{array}$$

$$(-5)_{10} \Rightarrow (11111011)_2$$

Binary of (-6) :-

$$(-6)_{10} = (110)_2$$

2	2	2	6	R
3			0	
1			1	
0			1	

$$\begin{array}{r} - & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1's & \oplus & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 2's & \cancel{\oplus} & & & & & & + & 1 \\ \hline & & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}$$

$$(-6)_{10} = (11111010)_2$$

Example :-

```
class Demo12
{
    System public static void main (String args[])
    {
        byte x = -5;
        byte y = -6;
        System.out.println (x & y);
    }
}
```

O/P \Rightarrow -6.

Solution :-

$$\begin{aligned} (-5)_{10} &\Rightarrow 11111011 \\ (-6)_{10} &\Rightarrow \underline{1111100100} \\ x \& y &\Rightarrow \boxed{1} \uparrow 111100100 \end{aligned}$$

$$\begin{array}{r}
 1's \quad 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 00 \ 01 \\
 0's \qquad \qquad \qquad + 1 \\
 \hline
 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 00 \\
 - \qquad \qquad \qquad 4 \ 2 \ 1
 \end{array}$$

$$x + y = -6.$$

what 1st bit is called sign bit.

Example:

$$(-6)_{10} = (1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0)_2$$

Sign bit

Sign bit if 1 means -ve and 0 means +ve.

$$\begin{matrix} 1 & \Rightarrow & -ve \\ 0 & \Rightarrow & +ve \end{matrix}$$

first we will check sign bit.

जब sign bit \pm होती, तो only तरफ v's complement
निकालना है। अगर sign bit 0 होती तो as it
का output आएगा।

Example :

class Demo12

```
public static void main (String [] args)
```

۱۵

byte x = -5;

byte y = -6;

```
System.out.println(xby);
```

```
System.out.println(xly);
```

```
System.out.println(x^y);
```

$$\begin{array}{r} 0/p \Rightarrow \\ -6 \\ -5 \\ 1 \end{array}$$

Solution:

$$xly \Rightarrow 0 \ 0 \ 0 \ 0 \ 0$$

$$\begin{array}{r} +5 \Rightarrow 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\ +6 \Rightarrow 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \\ \hline xly \Rightarrow 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \end{array}$$

$$\begin{array}{r} -5 \Rightarrow 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ -6 \Rightarrow 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \\ \hline xly \cdot 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \end{array}$$

$$\begin{array}{r} 5 = 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\ 1's = 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ 2's = \qquad \qquad \qquad +1 \\ \hline (-5)_{10} \overline{(1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1)}_2 \end{array} \quad (1)$$

$$\begin{array}{r} 6 = 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \\ 1's = 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \\ 2's = \qquad \qquad \qquad +1 \\ \hline (-6)_{10} \overline{(1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0)}_2 \end{array} \quad (2)$$

$$\begin{array}{r} x = 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \\ y = 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \\ \hline xly = \boxed{1} \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \end{array} \quad (3)$$

$$\begin{array}{r}
 1's \quad 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\
 2's \quad + \ 1 \\
 \hline
 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \\
 84. 2 \ 1
 \end{array}$$

$$x \& y \Rightarrow -6$$

$$\begin{array}{r}
 x \mid y \Rightarrow 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \\
 1's \quad 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \\
 2's \quad + \ 1 \\
 \hline
 -0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \\
 - \quad 4 \ 2 \ 1
 \end{array}$$

$$x \mid y \Rightarrow -5$$

$$x^n y = 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \quad \text{---} \ 5$$

$$x^n y = +1$$

Example:

```

class Demo12
{
    public static void main (String args[])
    {
        byte int x = -12 -12;
        byte int y = -13;
    }
}
    
```

System.out.println(x & y);

System.out.println(x | y);

System.out.println(x ^ y);

}

2	4	0
2	2	0
2	1	0
2	0	1

$$Q/P \Rightarrow -16$$

$$\begin{array}{r} -9 \\ + \end{array}$$

Solution :-

$$13 = 1 \ 1 \ 0 \ 1$$

$$12 = 1 \ 1 \ 0 \ 0$$

$$13 = 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1$$

$$1's = 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0$$

$$2's = + 1$$

$$(-13)_{10} \quad \overline{(1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1)_2} \quad \textcircled{1}$$

$$12 = 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0$$

$$1's = 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1$$

$$2's = + 1$$

$$(-12)_{10} \quad \overline{(1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0)_2} \quad \textcircled{2}$$

~~Q & P~~

$$-12 = 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0$$

$$-13 = 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1$$

$$x \& y = \boxed{1} \quad 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0$$

$$1's = \boxed{0} \quad 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1$$

$$2's = \frac{+ 1}{0 \ - \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0}$$

$$16 \ 8 \ 4 \ 2 \ 1$$

$$x \& y \Rightarrow -16 \quad \textcircled{3}$$

$$x/y = \boxed{1} \quad 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1$$

$$1's = \boxed{0} \quad 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0$$

$$2's = + 1$$

$$\begin{array}{c|ccccc} 2 & 1 & 3 & R \\ 2 & 6 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 2 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 \end{array} \quad \begin{array}{c|ccccc} 2 & 12 & R \\ 2 & 6 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 2 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 \end{array}$$

0 0 0 1 0 0 1

$$x \mid y = -9 \quad \text{--- (4)}$$

$$x^{\wedge}y = 0 0 0 0 0 1 1 1$$

$$x^{\wedge}y = 7. \quad \text{--- (5)}$$

Example :

```

class Demo12
{
    public static void main(String args[])
    {
        byte x = -13;
        byte y = -16;
        System.out.println(x & y);
        System.out.println(x | y);
        System.out.println(x ^ y);
    }
}

```

O/P $\Rightarrow -16$
 $\begin{array}{r} -13 \\ \hline 3 \end{array}$

Solution :

$$-13 = (1101)_2$$

$$16 = (10000)_2$$

$$13 = 000011101$$

$$1's \approx 111110010$$

$$2's \rightarrow \overline{(11110011)}_2 \quad \text{--- (1)}$$

$$\begin{array}{r}
 13 \\
 16 \\
 \hline
 0
 \end{array}
 \quad \text{R}$$

$$\begin{array}{r}
 16 \\
 8 \\
 \hline
 0
 \end{array}
 \quad \text{R}$$

$$\begin{array}{r}
 16 = 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \\
 1's = 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \\
 2's = + 1 \\
 \hline
 (-16)_{10} = (01010110000)_2 \quad (2)
 \end{array}$$

$$\chi_{+} = 1 - 1 + 1 \quad 0 \quad 0 \quad 1 \quad 1$$

$$y = 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

$$xly = \boxed{1} \quad \begin{array}{r} \\ \uparrow \\ \text{1's} \end{array} \quad \left| \begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \right. + 1$$

$$x \& y = -16. \quad \text{---} \quad ③$$

$$\begin{array}{r}
 x/y = 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \\
 1's = 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \\
 2's \\
 \hline
 -0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1
 \end{array}$$

$$x/y = -12 \longrightarrow (4)$$

$$x^1 y = 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1$$

$$x^1y = +3$$

```
# System.out.println(Integer.toBinaryString());
```

जूस method की help से हम check कर सकते हैं कि हमने binary अही निकाला है या नहीं।

For example:

```
class Demo15
{
    public static void main (String args [])
    {
        System.out.println (Integer.toBinaryString (-5));
    }
}
```

O/p \Rightarrow 11111011

Example:

```
class Demo15
{
    public static void main (String args [])
    {
        String s1 = "Ham";
        String s2 = "Sita";
        System.out.println (s1 - s2);
    }
}
```

error: incompatible type: bad operand type for
binary operator '-'

/ first type : string
Second type : string

Example:

```
class Demo15
{
    public static void main (String args [])
    {
```

```
String s1 = "ram";
int y = 10;
System.out.println(s1 - y);
}
}
```

error: bad operand type for binary operator
's1 - y';

first type: String
Second type: int

Example:

```
class Demo15
{
    public static void main(String args[])
    {
        String s1 = "ram";
        System.out.println(++s1);
    }
}
```

error: bad operand type String for unary
operator '+';

```
System.out.println(++s1);
```

Example:

```
class Demo15
{
    public static void main (String args[])
    {
        int x = 10;
        int y = 20;
    }
}
```

```
System.out.println ("sum = " + x+y); // sum = 10+0  
System.out.println ("sum = " +(x+y)); // sum = 30  
System.out.println ("sub= " + x-y); // error  
System.out.println ("sub=" +(x-y)); // sub = -10
```

y
g

error :

Example :

```
class Demo15 {  
    public static void main(String args[]) {  
        System.out.println(  
            int x=10;  
            String s1 = "ram" + x;  
            System.out.println(s1);  
        }  
    }  
O/p ⇒ ram 10
```

Example :

```
class Demo15 {  
    public static void main (String args[]) {  
        int x=10;  
        int y=20;  
        String s1 = "ram" + x+y;  
        System.out.println(s1);  
    }  
}
```

O/P \Rightarrow Ram 10 & 0

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        int x=10;
        int y=20;
        String s1 = "ram" + x-y;
        System.out.println(s1);
    }
}
```

error: bad operand type for binary operator '-';
first type : String
second type : int

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        int x=10;
        int y=20;
        String s1 = "ram" + (x-y);
        System.out.println(s1);
    }
}
```

O/P \Rightarrow ram - 10

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        int x = 10;
        int y = 20;
        int z = 30;
        String s1 = "x+y+z+ " + x+y+z;
        System.out.println (s1);
    }
}
```

O/p \Rightarrow 60

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        int x = 10, y = 20, z = 30;
        String s1 = x+y+z + " " + x+y+z;
        System.out.println (s1);
    }
}
```

O/p \Rightarrow 60 10 20 30

Example : for converting String to int -

```
class Demo15
{
    public static void main (String args[])
    {
        String s1 = "10";
    }
}
```

```
int x = Integer.parseInt(s1);
System.out.println(x+x);
System.out.println(s1+s1);
}
```

O/p ⇒ 20
1010

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        String s1 = "ab";
        int x = Integer.parseInt(s1);
    }
}
```

Error : number format exception for input string
"ab"

- # Integer.parseInt String को integer में convert करता है but String में only number (digit) होना चाहिए otherwise convert नहीं होगा।
- # java has total 53 reserved word. In which 50 are keywords and the other three are literals.
- # We can not use keyword at the place of variable name.

Literal :- जावा में variable का अनु-प्रयोग - value of store कर सकता है, वो उसके literal जैसा होता है।

For example : boolean has two literals. One is true and the other is false.

From 50 keyword we can only use 48 keyword java ~~without~~ gave us permission to use only 48 keyword. means, we can not use 2 keywords in java that are const and goto.

This two keywords we can not use at the place of variable name. because we can not use reserved words at the place of variable name.

Data type operator :-

short, byte, long, float, double, boolean, int, char

Program control statement :- (10 keywords)

if, else, switch, case, break, default, do, while, for, continue.

OOPS (17 keyword) :-

class, new, public, protected, private, this, abstract, extend, interface, implement, package, import, void, return, static, super, final.

Exception handling (5 keywords) :-

try, catch, finally, throw, throws

other (8 keywords) :-

assert, volatile, native, synchronized, strictfp,
enum, transient, instance of.

strictfp :—(strict floating point)

Example :

```
class Demo15
{
    public static void main (String args [])
    {
        String assert = "ab";
    }
}
```

error : as of release 1.4, 'assert' is a keyword
and may not be used as an identifier.

Example :

```
class Demo15
{
    public static void main (String args [])
    {
        System.out.println (1000);
        System.out.println ("1000");
        System.out.println (10.8);
        System.out.println ('A');
        System.out.println (false);
        System.out.println (true);
    }
}
```

O/p \Rightarrow

1000
1000
10.8
A
false
true

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        System.out.println(1000);
        System.out.println(A);
    }
}
```

error : can not find symbol.
System.out.println(A);
 ^

• parenthesis में भी all data type के corresponding value pass कर सकते हैं, because internally उन्होंने method overriding का use किया है।

Example :

```
class Demo15
{
    public static void main (String args[])
    {
        boolean b1 = 10;
        System.out.println(b1);
    }
}
```

error: incompatible type: int can not be converted to boolean.

- we can use ~~not~~ Logical NOT (!) operator ~~is~~ more than one time in continuous order.

Example:

```
class Demo15
{
    public static void main (String args[])
    {
        boolean b1 = true;
        System.out.println('!'+'!'+'!'+'b1'); // false true → false
    }
}
```

O/p ⇒ false

Example:

```
class Demo15
{
    public static void main (String args[])
    {
        float x = 10.8;
        ++x;
        System.out.println(x);
    }
}
```

error: incompatible type: possible lossy conversion from double to float.

B7
Example :

```
class Demo15
{
    public static void main (String args [])
    {
        double x = 10.8;
        ++x;
        System.out.println(x);
    }
}
```

O/p \Rightarrow 11.8

Example :

```
class Demo15
{
    public sum (String ar[])
    {
        float x = 10.8f;
        System.out.println(x);
    }
}
```

O/p \Rightarrow 10.8

Example :

~~```
class Demo15
{
 public sum (String ar[])
 {
 int x = 10;
 int y = 15;
 }
}
```~~

~~(x > y) && printf~~

Example : A C program to find largest of two number without program control statement.

```
#include <conio.h>
#include <stdio.h>
void main()
{
 int x = 10;
 int y = 20;
 (x > y) && printf("x is largest\n");
 (y > x) && printf("y is largest\n");
 getch();
}
```

O/p  $\Rightarrow$  y is largest

Example : C program to find largest among three numbers without program control statement.

```
#include <conio.h>
#include <stdio.h> void main()
{
 int x = 10;
 int y = 15;
 int z = 8;
 ((x > y) && (x > z)) && printf("x is largest\n");
 ((y > x) && (y > z)) && printf("y is largest\n");
 ((z > x) && (z > y)) && printf("z is largest\n");
 getch();
}
```

O/p  $\Rightarrow$  y is largest

### • Example :

```
#include <stdio.h>
#include <conio.h>
void main()
{
 1 11. printf ("yam");
 0 11. printf ("sita");
}
yam
Sita
```

O/p  $\Rightarrow$  Sita

### Example :

```
#include <stdio.h>
#include <conio.h>
void main()
{
 1 && printf ("yam");
 0 && printf ("sita");
 getch();
}
```

O/p  $\Rightarrow$  Yam

### Example :

```
class Demo15
{
 psvm (String arr[])
 {
 int x=10;
```

```
x++ ++x;
System.out.println(x);
}
}
```

error : not a statement

```
x++ + ++x;
^
```

### Example :

```
class Demo15
{
 psvm (String args[])
 {
 int x=10;
 boolean b=false;
 System.out.println(b==x);
 }
}
```

error : incomparable

type : int and boolean

### Example :

```
class Demo15
{
 psvm (String arr[])
 {
 String s1 = "yam";
 boolean b1 = false;
 System.out.println(b1==s1);
 }
}
```

error : bad operand type for binary operator (==)  
System.out.println(s1 == b);

first type : string  
second type : boolean

Example :

```
class Demo
{
 ps v m (String arr[])
{
 String s1 = " ram ";
 boolean b = " false ";
 s.o.println(100);
}
```

error : incompatible types

string can not be converted to boolean.

Example :

Example :

```
class Demo
{
 ps v m (String arr[])
{
 s.o.println(1 & 1);
 s.o.println(1 & 0);
}
```

error : bad operand types  
for binary operator  
(&)

Example :

```
class Demo
{
 ps v m (String arr[])
{
 int x = 1;
 int y = 1;
 s.o.println(x & y);
}
```

O/p  $\Rightarrow$  1

Example :

```
{
 s.o.println(true & false);
}
```

- we can not compare into int and void.

Example :

```
{
 int x = 1;
 int y = 1;
 x & s.o.println(10);
}
```

Error : not a statement

Example :

```
{
 int x = 1;
 int y = 1;
 int z = x & s.o.println(10);
}
```

Error : void type not allowed here.

# s.o.println is of void return type.

Example :

```
int {
 int x = 100;
 x = printf ("is");
 printf ("%d", x);
}
O/P => 2
```

Example :

```
{
 int x = 1;
 int y = 10;
 int z = x & s.o.printf("%d", y);
}
```

Error : bad operand type for binary operator.

first type : int

Second type : printstream

- Return type of printf is printstream.

Example :

```
{
{
int x=2;
switch(x)
{
case 1:
 s.o.println ("softwaves-1");
 break;
case 2:
 s.o.println ("softwaves-2");
 break;
case 3:
 s.o.println ("softwaves-3");
 break;
}
}
}
O/p => Softwaves-2
```

break;

case 3:

s.o.println ("softwaves-3");

break;

}  
}

O/p => blank screen

Example :

```
{
{
int x=23;
switch(x)
{
case 1:
 s.o.println ("softwaves-1");
 break;
case 2:
 s.o.println ("softwaves-2");
 break;
case 3:
 s.o.println ("softwaves-3");
 break;
default:
 s.o.println ("Invalid Case");
}
}
}
```

int x = 23;

switch (x)

{

case 1 :

s.o.println ("softwaves-1");

break;

case 2 :

s.o.println ("softwaves-2");

break;

case 3 :

s.o.println ("softwaves-3");

break;

default :

s.o.println (" Invalid Case");

}

}

}

O/p => Invalid Case.

Example :

```
{
{
int x=23;
switch(x)
{
case 1:
 s.o.println ("softwaves-1");
 break;
case 2:
 s.o.println ("softwaves-2");
 break;
}
```

Example :

```
{
 int x=2;
 switch(x)
 {
 case 1:
 s.o.println("Softwaves-1");
 break;
 case 2:
 s.o.println("Softwaves-2");
 break;
 case 3:
 s.o.println("Softwaves-3");
 break;
 default:
 s.o.println("invalid case");
 break;
 }
}
```

error: duplicate case label

- we can not create duplicate case.

Example :

```
{
 int x = 2;
 switch(x)
 {
```

case 1:

```
s.o.println("Softwaves-1");
```

```
break;
```

case 2:

```
s.o.println("Softwaves-2");
```

```
break;
```

case "ram":

```
s.o.println(" Softwaves-3");
```

```
break;
```

```
{
```

```
{
```

```
{
```

error: incompatible type:  
String can not be  
converted to int.

- if we pass integer value in switch / variable .  
So, it is compulsory to be integer value in case .

Example :

```
{
```

```
{
```

```
byte x=2; // -128 to 127
```

```
switch(x)
```

```
{
```

case 10:

```
s.o.println("Softwaves-1");
```

```
break;
```

case 100:

```

S.o.println("Softwaves-2");
break;
case 1000:
S.o.println("Softwaves-3");
break;
default:
S.o.println("Invalid case");
break;
}
}
}

error:

```

- It is compulsory to pass the value according to data type.

Example:

```

{
byte x=12;
switch (x)
{
case 10:
S.o.println("Softwaves-1");
break;
case 012:
S.o.println("Softwaves-2");
break;
case 100:
S.o.println("Softwaves-3");
break;
}

```

default:  
S.o.println("Invalid case");  
break;  
}  
}  
}  
error: duplicate case label.  
case 012:  
^

Example:

```

by tc x=10;
switch(x)
{
case 8:
S.o.println("softwaves-1");
break;
case 012:
S.o.println("softwaves-2");
break;
case 100;
S.o.println(" softwaves-3");
break;
default:
}
```

S.o.println(" Invalid case");  
break;

}

O/P → Softwaves-2

Example :

```
{
byte x=10;
switch(x)
{
case 8:
 s.o.println("softwaves-1");
 break;
case 012:
 s.o.println("softwaves-2");
 break;
case default:
 s.o.println("softwaves-3");
 break;
default:
 s.o.println("Invalid case");
 break;
}
}
}
```

error: duplicate default label.

Example :

```
{
byte x=2;
switch(x)
{
case 1:
 s.o.println("softwaves-1");
 break;
case 2:
 s.o.println("softwaves-2");
 break;
}
```

```
s.o.println("softwaves-2");
case 3:
 s.o.println("softwaves-3");
default:
 s.o.println("Invalid case");
}
}
}
O/P=> Softwaves - 2
Softwaves - 3
Invalid case
```

- for executing particular case . we use break.

Example :

```
{
byte x=22;
switch(x)
{
default("Invalid case"),
 s.o.println("Invalid case");
case 1:
 s.o.println("softwaves-1");
case 2:
 s.o.println("softwaves-2");
case 3:
 s.o.println("softwaves-3");
}
```

O/p → Invalid case  
Softwaves-1  
Softwaves-2

Example :

```
{
{
byte x = 2;
switch(2)
{
case 1:
 System.out.println("Softwaves-1");
 break;
case 2:
 System.out.println("Softwaves-2");
 break;
case 3:
 System.out.println("Softwaves-3");
 break;
 break;
default:
 System.out.println("Invalid case");
}
}
```

Output : Unreachable Statement.

• In Java we can not use more than one break  
# In Java, if we are making any software than the range of integer is sufficient for that. That's why we can not use ~~long~~ the data type of long data type. because we can only pass the cases by the range of integer data type.

Example :

```
{
{
long x = 2;
switch(x)
{
case 1:
 System.out.println("Softwaves-1");
 break;
case 2:
 System.out.println("Softwaves-2");
 break;
case 3:
 System.out.println("Softwaves-3");
 break;
}
```

```
}
```

error :

Example :

```
{
{
float x = 2.1;
switch(x)
{
case 1:
S.o.println("softwares-1");
break;
case 2;
S.o.println("Softwares-2");
break;
case 3:
S.o.println("Softwares-3");
break;
default:
S.o.println("Softwares-4");
}
}
}
error :
```

Example :

```
{
{
boolean x = false;
switch(x)
{
case true:
S.o.println("S-1");
break;
case false:
S.o.println("S-2");
break;
default:
S.o.println("I.C.");
break;
}
}
}
```

error: incompatible b type:  
boolean can not be  
converted to int.

switch(x)

- \* we can not use float and double data type in switch case statement. and also we can not use boolean data type.

30/11/21

# Java program to break the program without using break statement.

```
{
 int x=1;
 switch(1)
 {
 case 1 → System.out.println("S-1");
 case 2 → System.out.println("S-2");
 }
}
O/P ⇒ S-1
```

Dot(.) in cmd command:-

compile syntax → javac -source12 --enable-preview Demo15.java.

Run syntax → java --enable-preview Demo15

# preview feature की facility हैं java version 12  
को provide करती है।

Example :

```
{
 int x=12;
 switch(x)
 {
 case 1 → S.O.Println("S-1");
 case 2 → S.O.Println("S-2");
 default → S.O.Println("I.C.");
 }
}
O/p → I.C.
```

Example :

```
{
 int x=1;
 switch(x)
 {
 case 1, 6: S.O.Println("S-1");
 case 2, 3: S.O.Println("S-2");
 default: S.O.Println("I.C.");
 }
}
O/p → S-1
 S-2
 I.C.
```

# multiple case label is a part of previous feature.

Example :

```
{
 int x=8;
 switch(x)
 {
 case 1:
 case 3:
 case 5:
 case 7:
 case 9:
 S.O.Println("no. is odd");
 break;
 case 2:
 case 4:
 case 6:
 case 8:
 case 10:
 S.O.Println("no. is even");
 break;
 default:
 S.O.Println("Invalid case");
 }
}
O/p → no. is even.
```

# Java program according  
to the latest version  
of Java.

```
{
int x=7;
switch(x)
{
case 1,3,5,7,9 → System.out.println("no. is odd");
case 2,4,6,8,10 → System.out.println("no. is even");
default → ("Invalid case");
}
}
```

~~o/p~~ O/p ⇒ no. is odd.

Example :

```
{
int x=7;
switch(x)
{
case 1,3,7,5,9 → System.out.println("no. is odd"); break;
case 2,4,6,8,10 → System.out.println("no. is even");
default → ("Invalid case.");
}
}
```

error : case, default or '}' expected.  
break;

Example :

```
{
int x = 7;
switch (x)
{
case 1, 3, 5, 7, 9 → S.O.Println("no. is odd");
case 2, 4, 6, 8, 10 → S.O.Println("no. is even");
default : ("Invalid case");
}
}
```

error: different case kinds used in the switch

Example :

```
{
int x = 1;
switch (x)
{
case 1, 3 →
S.O.Println("S-1");
S.O.Println("S-11");
case 2, 6 →
S.O.Println("S-2");
default →
S.O.Println("I.C.");
}
```

Advantage of previous feature

- we can break a statement without using break keyword.
- we can write more than 1 cases for same statement. means, we can pass multiple case labels separated by comma.
- we can also use curly brackets for more than one statements.

} error: case, default, or '}' expected.

# We can not write more than one statement in case. # Java has provided to use the facility to use string in switch case from 7.0 version

If we want to write more than one statement in case. So we have to use curly braces ( { } )

for example :

```
{
{
int x=1;
switch(x)
{
```

case 1, 3 →

```
{
System.out.println("S-1");
System.out.println("S-11");
- Note since int case is
```

case 2 →

```
System.out.println("S-2");
default →
```

```
System.out.println("I.C.");
}
```

with same logic alternative  
execution of the

O/p ⇒ S-1  
S-11

# Java programming have 4 types of loops :-

i) while

ii) do while

iii) for

iv) for each

Example :

```
{
{
int i=1;
while (i)
{
System.out.println("S-1");
i++;
}
}
}
```

Output :

but in C :-

```
int i=1;
while (i)
```

1/12/21

```
{
 s.o.pf("%d\n", i);
 i++;
}
}
```

O/P  $\Rightarrow -32767$  to  $-1$   
(in the range of %d)

# we can not pass any thing in ~~loop~~ while loop apart from boolean value. because it is compulsory to pass only true or false in loops in java.

Example :

```
{
{
 for(int i=1; i<=10; i++)
 {
 s.o.println(i);
 s.o.println(i);
 }
}
```

error : can not find symbol.

Example :

```
{
{
 int i;
 for(i=1; false; i++)
 {
 s.o.println(i);
 s.o.println("yam");
 }
}
```

error : Unreachable statement.

Example :

```
{
{
 int i;
 for (i=1; i>=10 ;i++)
 {
 s.o.println(i);
 s.o.println("yam");
 }
}
```

O/P  $\Rightarrow$  yam.

# variable के case में checking  
runtime पर होती है।

# constant के case में checking  
compile time पर होती है।

Example :

```
{
 int i;
 for(i=1; true; i++)
```

```
{
 s.o.println(i);
}
```

```
{
}
```

```
{
}
```

O/p  $\Rightarrow \infty$  times

Example :

```
{
 int i;
```

```
 for(i=1; i<=10; i++)
```

```
{
 s.o.println(i);
}
```

```
{
 s.o.println("ram");
}
```

```
{
}
```

```
{
}
```

error : Unreachable statement

s.o.println("ram");

Example :

```
{
 int i;
```

```
 for(i=1; i<=10; i++)
```

```
{
 s.o.println(i);
}
```

```
{
}
```

```
{
}
```

O/p  $\Rightarrow \infty$  times

Example :

```
{
 int i;
 for(i=1; i<=10; i++)
 {
 s.o.println(i);
 }
 s.o.println("ram");
}
```

O/p  $\Rightarrow$  ram.

error : unreachable statement

Example :

```
{
 int i;
 for(i=1; i<=10; i++)
 {
 s.o.println(i);
 }
 s.o.println("ram");
}
```

error : unreachable statement

s.o.println("ram");

Example:

```
{
 int i;
 for(i=1; i<=10; i++)
 {
 s.o.println(i);
 i--;
 }
 s.o.println("ram");
}
```

O/p  $\Rightarrow$  10 times 1

Example:

```
{
 int i;
 for(i=0; i<=10; i++)
 {
 i++;
 s.o.println(i);
 }
 s.o.println("ram");
}
```

O/p  $\Rightarrow$  1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
ram

Example:

```
{
 int i;
 for(i=1; i<=5; i++)
 {
 s.o.println(i);
 s.o.println("ram");
 }
}
```

O/p  $\Rightarrow$  1  
2  
3  
4  
5  
ram  
ram  
ram  
ram  
ram

Example:

```
{
 int i;
 for(i=1; i<=5; i++)
 {
 s.o.println(i);
 s.o.println("ram");
 }
}
```

O/p  $\Rightarrow$  1  
2  
3  
4  
5  
ram

# यदि किसी curly braces का use  
नहीं करते हैं, तो by default  
पूरी loop की body 1<sup>st</sup> semicolon  
तक मानता है।

Example :

```
{
 int i;
 for(i=1; i<=5; i++)
 ;
 cout<<i;
 cout<<"ram";
}
O/P => 2
 3
 4
 5
 6.
```

Example :

```
{
 int i=1;
 for(; i<=5;)
 ;
 cout<<i;
}
O/P => 2
 3
 4
```

∞ times

Example :

```
{
 int i;
 for(i=1; i<=10; i++);
 {
 cout<<i;
 }
}
O/P => 11
```

5

Example :

```
{
{
int i = 1;
for(;);
{
i++;
}
s.o.println(i);
}
}
```

error: Unreachable statement  
s.o.println(i);

Example :

```
{
{
int i = 1;
for(i = 1; i <= 10; i++; i++)
{
s.o.println(i);
}
}
```

error:

Example :

```
{
{
int i = 1;
for(i = 1; i <= 10; i++, i++)
{
}
```

s.o.println();

}

}

}

O/p  $\Rightarrow \frac{1}{3} 5$

7

9

# We can use comma ,) ~~but~~ for  
than using more than two  
Semicolon (;).

Example :

```
{
{
int i = 1;
int x = 1;
for(i = 1; i <= 10; i++)
{
if(x++ < 5 && x++ < 10)
{
}
}
s.o.println(x);
}
```

O/p  $\Rightarrow 13.$

O/P  $\Rightarrow$

$i = 1; i <= 10 \Rightarrow \text{true}$

$1 < 5 \& 2 < 10 \Rightarrow \text{true}$

$i = 2; i <= 10$

$3 < 5 \& 4 < 10 \Rightarrow \text{true}$

$i = 3 ; 3 <= 10$

$5 < 5 \Rightarrow \text{false}$

$i = 4 ; 4 <= 10$

$6 < 5 \Rightarrow \text{false}$

$i = 5 ; 5 <= 10$

$7 < 5 \Rightarrow \text{false}$

$i = 6 ; 6 <= 10$

$8 < 5 \Rightarrow \text{false}$

$i = 7 ; 7 <= 10$

$9 < 5 \Rightarrow \text{false}$

$i = 8 ; 8 <= 10$

$10 < 5 \Rightarrow \text{false}$

$i = 9 ; 9 <= 10$

$11 < 5 \Rightarrow \text{false}$

$i = 10 ; 10 <= 10$

$12 < 5 \Rightarrow \text{false}$

$i = 11 ; 11 <= 10 \Rightarrow \text{false}$

$\boxed{x = 12}$

In the case of logical  $\&$  operator, when 1 condition is false than it will not check 2<sup>nd</sup> condition and the o/p will be false.

Example :

{

int i = 1;

int x = 1;

for (i = 1 ; i <= 10 ; i++)

{

if (x++ < 5 || x++ < 10)

{ }

}

s.o.println(x);

{ }

}

O/p  $\Rightarrow 17$

Sol -

$i = 1 ; 1 <= 10$

$1 < 5 \rightarrow \text{true}$

$i = 2 ; 2 <= 10$

$2 < 5 \rightarrow \text{true}$

$i = 3 ; 3 <= 10$

$3 < 5 \rightarrow \text{true}$

$i = 4 ; 4 <= 10$

$4 < 5 \rightarrow \text{true}$

$i = 5 ; 5 <= 10$

$5 < 5 \& 6 < 10 \rightarrow \text{true}$

$i = 6 ; 6 <= 10$

$7 < 5 \& 8 < 10$

$i = 7 ; 7 <= 10$

$9 < 5 \& 10 < 10$

```

i=8; 8<=10
11<5 || 12<50
i=9; 9<=10
13<5 || 14<10
i=10; 10<=10
15<5 || 16<10
i=11; 11<=10 → false.

```

$\boxed{x=17}$

In the case of logical  $\text{||}$  operator, when 1<sup>st</sup> condition is true then the o/p will be true. So if 1<sup>st</sup> condition is true so it will not check the 2<sup>nd</sup> condition.

Example :

```

{
 int i=1;
 int x=5;
 for(i=1; i<=10; i++)
 {
 if(x++<5 || x++<10)
 }
 s.o.println(x);
}
O/p = 25

```

Sol :-

|                          |                                                   |
|--------------------------|---------------------------------------------------|
| $i=1, x=5$               | $i=1; 1<=10$                                      |
| $5<5 \text{    } 6<10$   | $i=2; 2<=10$                                      |
| $7<5 \text{    } 8<10$   | $i=3; 3<=10$                                      |
| $9<5 \text{    } 10<10$  | $i=4; 4<=10$                                      |
| $11<5 \text{    } 12<10$ | $i=5; 5<=10$                                      |
| $13<5 \text{    } 14<10$ | $i=6; 6<=10$                                      |
| $15<5 \text{    } 16<10$ | $i=7; 7<=10$                                      |
| $17<5 \text{    } 18<10$ | $i=8; 8<=10$                                      |
| $19<5 \text{    } 20<10$ | $i=9; 9<=10$                                      |
| $21<5 \text{    } 22<10$ | $i=10; 10<=10$                                    |
| $23<5 \text{    } 24<10$ | $i=11; 11<=10 \rightarrow \text{condition false}$ |

$\boxed{x=25}$

Example :

```

int x = 5;
int i = 5;
for (i = 1; i <= 10; i++)
{
 if (x++ < 5 && x < 10)
}
S.o.pn(x);

```

O/p  $\Rightarrow$  ~~15~~ 15

Sol :-

$i = 5, x = 5$

$5 < 5 \rightarrow \text{false}$ .

$i = 2 ; 2 <= 10$

$6 < 5$

$i = 3 ; 3 <= 10$

$7 < 5$

$i = 4 ; 4 <= 10$

$8 < 5$

$i = 5 ; 5 <= 10$

$9 < 5$

$i = 6 ; 6 <= 10$

$10 < 5$

$i = 7 ; 7 <= 10$

$11 < 5$

$i = 8 ; 8 <= 10$

$12 < 5$

$i = 9 ; 9 <= 10$

$10 < 5$

$i = 10 ; 10 <= 10$

$11 < 5$

$i = 11 ; 11 <= 10 \rightarrow \text{false}$

$\boxed{i = 12}$   
 $x = 15$

Example :

```

int i;
for (i = 1; i <= 10; i++)
{
 S.o.pn(i);
}

```

O/p  $\Rightarrow$  11

Example :

```

int i = 1;
for (i = 1; i <= 10; S.o.pn(i))
{
 i++;
}

```

O/p  $\Rightarrow$

|    |    |   |   |   |   |
|----|----|---|---|---|---|
| 1  | 2  |   |   |   |   |
| 3  | 4  | 5 | 6 | 7 | 8 |
| 9  | 10 |   |   |   |   |
| 11 |    |   |   |   |   |

Example:

```
{
 int i = 1;
 for(s.o.println(i); i<=10; s.o.println(i))
 i++;
}
```

O/P  $\Rightarrow$  1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Example:

```
{
 int i = 1;
 for(s.o.println(i); i<=5; s.o.println(i))
 i++;
}
```

O/P  $\Rightarrow$  1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Example:

```
{
 int i = 1;
 for(s.o.println(i); s.o.println(i); s.o.
 println(i))
 i++;
}
```

error: void can not be converted to boolean  
for(s.o.println(i); s.o.println(i); s.o.println(i),  
 ^

Example:

```
{
 int i = 1;
 for(s.o.println(i); i<=5; i++,
 s.o.println(i));
 i++;
}
```

O/P  $\Rightarrow$  1  
2  
3  
4  
5  
6.

Example:

```
{
 int i = 1;
 for(s.o.println(i); i<=5; i++,
 s.o.println(i), i++)
```

3/12/21

• Labeled break & continue statement :-

Example :-

{  
}

int i;

for (i=1; i<=10; i++)

{  
}

if (i==5) break;

so.println(i);

{  
}  
}

O/P ⇒ 1  
2  
3  
4

Example :-

{  
}

int i;

for (i=1; i<=10; i++)

{  
}

if (i==5) continue;

so.println(i);

{  
}  
}

O/P ⇒ 1 7  
2 8  
3 9  
4 10  
6

i++;  
2  
3

O/P ⇒ 1  
3  
6  
5

Example :-

{  
}

int i=1;

int x=1;

for (i=1; i<=10; i++)

{  
}

if (x++<5 && x++<10)

{  
}

so.println(x);

{  
}

O/P ⇒ 1 3

~~Ex~~

Example:

```
{
 int i;
 for(i=1; i<=10; i++)
 {
 if (i<=5) continue;
 cout << i;
 }
}
```

O/P => 6  
7  
8  
9  
10.

Example:

```
{
 {
 cout << "S-1";
 cout << "S-2";
 }
 k:
 cout << "S-3";
 cout << "S-4";
}
```

O/P => S-1  
S-2  
S-3  
S-4

Example:

```
{
 int f=5;
}
```

k:

```
s.o.println("Softwares-1");
s.o.println("Softwares-2");
if (i==5) continue;
s.o.println("Softwares-3");
s.o.println("Softwares-4");
}
```

Error: continue outside of loop.

Example:

```
{
 int i=5;
 k:
 cout << "Softwares-1";
 cout << "Softwares-2";
 if (i==5) continue k;
 cout << "Softwares-3";
 cout << "Softwares-4";
}
```

Error: not a loop label: k.

# we can use continue in only loop.

# we can not use break in if-else (program control statements)

Example: Labelled break:

```
{
 int i=5;

 K:
 {
 S.o.println("softwaves-1");
 S.o.println("softwaves-2");
 if(i==5) break K;
 S.o.println("softwaves-3");
 S.o.println("softwaves-4");
 }
}
```

O/p  $\Rightarrow$  softwaves-1  
softwaves-2

Example: Normal break:

```
{
 int i=5;

 K:
 {
 S.o.println("softwaves-1");
 S.o.println("softwaves-2");
 if(i==5) break;
 S.o.println("softwaves-3");
 S.o.println("softwaves-4");
 }
}
```

Note: break outside switch or loop

Example:

```
{
 int i=5;

 K:
 {
 S.o.println("softwaves-1");
 S.o.println("softwaves-2");
 if(i==5) break K;
 S.o.println("softwaves-3");
 S.o.println("softwaves-4");
 }
}
```

O/p  $\Rightarrow$  softwaves-1  
softwaves-2  
softwaves-4

Example:

```
{
 int i=5;

 K:
 {
 S.o.println("softwaves-1");
 S.o.println("softwaves-2");
 if(true) break K;
 S.o.println("softwaves-3");
 S.o.println("softwaves-4");
 }
}
```

O/p  $\Rightarrow$  softwaves-1  
softwaves-2

Example :

```
{
{
int i=5;
k:
{
S.o.println("Softwaves-1");
S.o.println("Softwaves-2");
if (false) break k;
S.o.println ("Softwaves-3");
S.o.println ("Softwaves-4");
}
}
```

O/p → Softwaves-1  
Softwaves-2  
Softwaves-3  
Softwaves-4

Example :

```
{
{
int i=5;
k:
{
S.o.println ("Softwaves-1");
S.o.println ("Softwaves-2");
break k;
}
}
S.o.println ("Softwaves-3");
S.o.println ("Softwaves-4");
}
}
}
```

O/p → Softwaves-1  
Softwaves-2  
Softwaves-3  
Softwaves-4

Example :

```
{
{
int i=5;
k:
{
S.o.println (" Softwaves-1 ");
S.o.println (" softwaves-2 ");
break;
}
}
S.o.println (" Softwaves-3 ");
S.o.println (" softwaves-4 ");
}
```

error : break outside switch  
or loop .

Example :

```
{
{
for (int i= 1 ; i<=10 ; i++)
{
for (int j= 1 ; j<=10 ; j++)
{
if (j == 5) break;
S.o.println(j+ " ");
}
S.o.println();
}
}
```

O/p → 1 2 3 4 5 6 7 8 9 10 times

Example : labelled break in loop .

4/12/21

```

class demo
{
 ps v m (String L1ar)
}
C_N or obj-name = new C N ();
A a, = new A ();

a1.show();
a1.show();

}
}

O/p => ram
 ram

```

- यदि हमें class के अंदर का date access करना है, तो class का object लाना कर ready करना होगा।

# Array :- Collection of homogeneous type of data is called array.

Uses :-

Example:

## class A

1

```
void show()
{
```

`s.o.println("ram");`

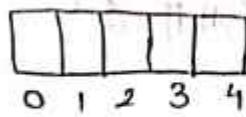
9  
2

## Syntax:

```
D.T. var-N[] = new D.T.[size];
```

## Example:

```
int x[] = new int[5];
```



## Example:

```
{
}
```

```
int x[] = new int[5];
```

```
s.o.println(x[0]);
```

```
s.o.println(x[1]);
```

```
s.o.println(x[2]);
```

```
s.o.println(x[3]);
```

```
s.o.println(x[4]);
```

```
{
}
```

O/P → 0  
0  
0  
0  
0

# Java में अचेही अवधि में memory छन कर ready होती है, तो उसमें fix zero ही आकर store हो जाता है।

# जब हम float का use करते हैं, तो default value 0.0 आकर store ही जाता है।

## Example :

```
{
}
```

```
float z[] = new float[5];
```

```
s.o.println(z[0]);
```

```
s.o.println(z[1]);
```

```
s.o.println(z[2]);
```

```
s.o.println(z[3]);
```

```
s.o.println(z[4]);
```

```
{
}
```

```
{
}
```

O/P ⇒ 0.0

0.0

0.0

0.0

0.0

# boolean data type के case में bydefault false आकर store ही जाता है।

## Example:

```
{
}
```

```
boolean z[] = new boolean[5];
```

```
s.o.println(z[0]);
```

```
s.o.println(z[1]);
```

```
s.o.println(z[2]);
```

```
s.o.println(z[3]);
```

```
s.o.println(z[4]);
```

```
{
}
```

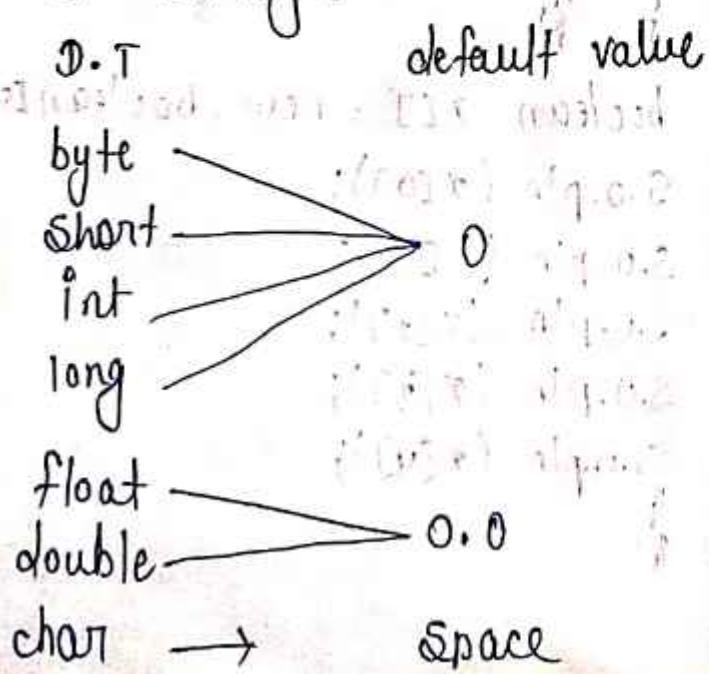
O/p → false  
false  
false  
false  
false.  
false.

# character के case में space  
print लिया है।

Example:

```
{
char x[] = new char[5];
s.o.println(x[0]);
s.o.println(x[1]);
s.o.println(x[2]);
}
O/p → _ _ _ _
```

• In Array :-



O/p → boolean → false

# In Java, we can make  
array of ~~obj~~ class.

# String के case में default  
value null आती है।

Example:

```
{
String x[] = new String[5];
s.o.println(x[0]);
s.o.println(x[1]);
s.o.println(x[2]);
}
O/p → null
null
null
```

Example:

```
{
int x[] = new int[5];
int i; i=0; i<5; i++)
{
s.o.println(x[i]);
}
O/p → 0
0
0
0
0
```

## I/O → Input / output

# जितनी time लम्बे array का index declare किया है, उतने ही time data fetch होना चाहिए।

Example :

```

int x[] = new int[5];
int i;
for (i=0; i<6; i++)
{
 System.out.println(x[i]);
}

```

O/P ⇒

```

0
0
0

```

जहाँ index out of bound exception : Index 5 out of bound at class Demo.

# अब पहली exception आती है, उसके बाद exception नहीं आती है। पहली exception आते ही program terminate हो जाएगा।

- System.out → Standard output device screen.
- System.in → Standard input device keyboard

• keyboard से data fetch करने के लिए java ने हमें एक class provide की है, जिसका नाम ISR है।

- ISR :- Input Stream Reader
- Stream means पाइप लाईन की तरह।

ISR के पास कौन सी functionality नहीं है।

- ISR से data हमें BR के पास fetch करवाना है।

• BR :- BufferedReader

- BR class के पास read method है and read method single character को read करता है and उसकी integer value को return करता है।

## Object syntax:

Input Stream Reader  
i = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(i);

int x = br.read();

S.O. println(x);

- read method exception को throw करती है। इस जरूर नहीं  
read method का use करने से एक exception error  
आकर display होगी।

→ package → It is just like header file in java

• C has header file.

• Java have packages. In this packages have the predefined classes.

• Read method throw exception.

• handle with try catch method or through

with throws keyword.

error?

Unreported exception IOException must be caught or declared to be thrown.

int x = br.read();

Example :

Package                      Class

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
class Demo
{
 public void m (String str) throws IOException
 {
 InputStreamReader i = new InputStreamReader(System.in);
 BufferedReader br = new BufferedReader(i);
 System.out.println("Enter Any character.");
 int x = br.read();
 System.out.println("Character is = " + x);
 }
}
```

# Java का एक ऐसा package है। जो by default import हो जाता है उस package का नाम java.lang है।  
string, System, ~~etc~~ Integer, classes java.lang में रखा है।  
# read method का written type int है।

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
class Demo
{
 public void main(String ar[])
 {
 InputStreamReader i = new InputStreamReader(System.in);
 BufferedReader br = new BufferedReader(i);
 System.out.println("Enter Any character");
 char x = br.read();
 System.out.println("Character is = " + x);
 }
}
```

error: incompatible types: possible lossy conversion  
from int to char.

Example:

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
class Demo
{
 public void main(String ar[])
 {
 InputStreamReader i = new InputStreamReader(System.in);
 BufferedReader br = new BufferedReader(i);
 System.out.println("Enter Any character");
 char x = br.read();
 System.out.println("Character is = " + x);
 }
}
```

```
InputStreamReader i = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(i);
System.out.println("Enter Any Character");
char x = (char)br.read();
System.out.println("Character is = "+x);
}
```

7/12/21

**import java.io.\*;** → means, with the help of this we can access all classes.

Example :

```
import java.io.*;
class Demo {
 public void main (String args) throws IOException {
 InputStreamReader i = new InputStreamReader(System.in);
 }
}
```

```
BufferedReader br = new BufferedReader(i);
System.out.println("Enter Any Number");
char x1 = (char)br.read();
char x2 = (char)br.read();
char x3 = (char)br.read();
char x4 = (char)br.read();
```

```
S.o.println("Character is=" + x1);
S.o.println("Character is=" + x2);
S.o.println("Character is=" + x3);
S.o.println("Character is=" + x4);
{
}
}
```

#br.readLine(); :- It reads whole line. BufferedReader  
Example:

```
import java.io.*;
class Demo
{
 public void main (String ar[]) throws IOException
 {
 }
```

```
 InputStreamReader i = new InputStreamReader (System.in);
 BufferedReader br = new BufferedReader (i);
 S.o.println ("# Enter name ");
 String s1 = br.readLine();
 S.o.println ("Name = " + s1);
}
```

Example:

```
import java.io.*;
```

```
class Demo
```

```
{
```

    public void main(String args) throws IOException

```
{
```

```
 InputStreamReader i = new InputStreamReader(System.in);
```

```
 BufferedReader br = new BufferedReader(i);
```

```
 System.out.println("Enter 2 No.");
```

```
 int x = br.read();
```

```
 int y = br.read();
```

```
 System.out.println("Sum = " + (x+y));
```

    } // End of class

Enter a No.

ASCII

22 78 //  $a = 48$   $b = 50 \Rightarrow 100$

Sum = 100

Enter a No.

10 87 //  $i = 49 + 0 = 48 \Rightarrow 97$

Sum = 97

Enter a No.

25 77 //  $a = 51 + 5 = 52 \Rightarrow 103$

Sum = 103

Enter a No.

20 78 //

Sum = 98.

Enter a No.

Sum = 106.

55 51

Sum = 106.

Because read method only it reads only 1 character.

# Headline of written type string & |

Example: Sum of two numbers by taking input from user.

```
InputStreamReader i = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(i);
System.out.println("Enter No. 1");
String s1 = br.readLine();
System.out.println("Enter No. 2");
String s2 = br.readLine();
int x = Integer.parseInt(s1);
int y = Integer.parseInt(s2);
System.out.println("Sum = " + (s1 + s2));
System.out.println("Sum = " + (x + y));
```

}

}

Enter no. 1

ab

Enter no. 2

20

Exception.

ab  
20  
= 660

Java  
Simple  
Calculator  
Program

Example :-

```
import java.io.*;
class Demo {
 public void main (String args[]) throws IOException {
 InputStreamReader i = new InputStreamReader (System.in);
 BufferedReader br = new BufferedReader (i);
 System.out.println ("Enter Name");
 String s1 = br.readLine ();
 System.out.println ("Enter G");
 char g = (char) br.read ();
 System.out.println ("Enter add");
 String s2 = br.readLine ();
 char
 System.out.println ("name = " + s1);
 System.out.println ("g = " + g);
 System.out.println ("add = " + s2);
 }
}
```

In cmd :-

Enter Name Harshita  
Enter G F  
F add =

Example :

```

{
 int x = '10';
 int y = '11';
 S.o.println(x);
 S.o.println(y);
}
}
O/p => 10
 11

```

Example :

```

class import java.io.*;
class Demo
{
 public void main (String ar[])
 throws IOException
 {
 ISR i = new ISR (System.in);
 BR br = new BR (i);
 S.o.println ("Enter Name");
 char a = (char) br.read();
 char b = (char) br.read();
 char c = (char) br.read();
 char d = (char) br.read();
 S.o.println (a);
 S.o.println (b);
 S.o.println (c);
 }
}

```

```
s.o.println(d);
```

```
}
```

```
when we enter 4 characters शो, it
```

```
headline \n शो read शोता हि
```

```
value of enter key is 13. means \r.
```

# enter key press करने पर दो  
character generate होते हैं।

- i) \r → 13  
ii) \n → 10

9 b c d → input  
O/p → 97  
98  
99  
100

Example:

```
import java.io.*;
```

```
{
{
```

```
s.o.println("Enter name");
int a = br.read();
int b = br.read();
int c = br.read();
int d = br.read();
```

```
s.o.println(a);
```

```
s.o.println(b);
```

```
s.o.println(c);
```

```
s.o.println(d);
```

```
}
}
```

abc → input  
O/p → 97  
98  
99  
13

ab → input  
O/p → 97  
98  
13  
10.

a → input  
b  
O/p → 97  
98 13  
10.

= → input  
O/p → 13  
10  
13  
10

# यदि हम only one character enter करते हैं, तो  
वो fourth character को लिए wait करता है।

# read method का तक wait करती है यदि तक कि  
उसको all character ना मिल जाए।

Example:

```
import java.io.*;
```

{

```
ISR i = new ISR (System.in);
```

```
BR br = new BR (i);
```

```
S.o.println ("Enter name");
```

```
String s1 = br.readLine();
```

```
S.o.println ("Enter G1");
```

```
char g = (char) br.read();
```

```
int g1 = br.read();
```

```
int g2 = br.read();
```

```
S.o.println ("Enter add");
```

```
String s2 = br.readLine();
```

```
S.o.println ("name = " + s1);
```

```
S.o.println ("G1 = " + g);
```

```
S.o.println ("add = " + s2);
```

```
S.o.println ("g1 = " + g1);
```

```
S.o.println ("g2 = " + g2);
```

}

input :—

Enter name.

Manisha

Enter G1

f

Enter odd

Indore

O/p =

Harshita

f

Indore

13

10

Q:- splitting the number from space and sum of them.

Example:

```
{
{
String s1 = "10 20";
int x = Integer.parseInt(s1);
System.out.println(x+x);
}
}
```

exception: java.lang.NumberFormatException: for input string: "10 20"

Reason: → bcz of space. bcz space is a character & we can only pass integer value in string to convert it into int.

- # StringTokenizer :— with the help of this we can split the string according to our requirement. It is in util name package.
- # parameter में लम्बी पास करना है, जिसे space के split करना है।
- # util ⇒ utility.
- # ~~Object~~ Token जो head करने के purpose के StringTokenizer ने हमें nextToken(); provide की है।
- # return type of nextToken(); is String.

Example :

```
import java.io.*;
```

```

String s1 = "my name";
StringTokenizer st = new StringTokenizer(s1);
String s2 = st.nextToken();
String s3 = st.nextToken();
System.out.println(s2);
System.out.println(s3);
```

O/p → My  
name

If nextToken method point the data which are stored in StringTokenizer method.

```
Example : import java.util.*;
{
String st = "My name";
StringTokenizer st = new StringTokenizer(st);
String s1 = st.nextToken();
String s2 = st.nextToken();
String s3 = st.nextToken();
String s4 = st.nextToken();
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
System.out.println(s4);
}
}
```

Exception :- NoSuchElementException.

# Example :

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

```
import java.io.*;
```

```
{
```

```
{
```

```
ISR i = new ISR(System.in);
```

```
BR br = new BR(i);
```

```
System.out.println("Enter any msg");
```

```
String st = br.readLine(); // my
```

```
StringTokenizer st = new StringTokenizer(st);
```

```
String s1 = st.nextToken();
```

```
String s3 = st.nextToken();
```

```
s.o.println(se);
```

```
s.o.println(s3);
```

```
{
```

```
}
```

input  $\Rightarrow$  My name

O/p  $\Rightarrow$  My ~~name~~

input  $\Rightarrow$  My <sup>name</sup> name is @ite

O/p  $\Rightarrow$  my ~~name~~

input  $\Rightarrow$  My <sup>name</sup>.

Exception: NoSuchElementException

9/12/21

# Addition of two no: take input from user.

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

```
import java.io.*;
```

```
{
```

```
{
```

```
s.o.print("Enter 2 no.");
```

```
String s1 = br.readLine();
```

```
StringTokenizer st = new StringTokenizer(s1);
```

```
String s2 = st.nextToken();
```

```
String s3 = st.nextToken();
```

```
int x = Integer.parseInt(s1);
```

```
int y = Integer.parseInt(ss);
System.out.println("sum = " + (x+y));
}
```

Case 1 : Input  $\Rightarrow$  10 20

O/P  $\Rightarrow$  sum = 30

Case 2 : Input  $\Rightarrow$  10 20 50

O/P  $\Rightarrow$  sum = 30

Case 3 : Input  $\Rightarrow$  10 abc

Exception : no. format Exception for input string  
'abc'.

Case 4 : Input  $\Rightarrow$  10

Exception : NoSuchElementException.

Case 5 : Input  $\Rightarrow$  10,20

Exception : NoSuchElementException  
(Demo 21.java:13)

# St. nextToken()  $\Rightarrow$  When you want to split the string as per. according to your need.

Example :

```
import java.util.*;
import java.io.*;
String s1 = br.readLine();
StringTokenizer st = new StringTokenizer(s1, " ");
String s2 = st.nextToken(); nextToken();
String s3 = st.nextToken(); nextToken();
int x = Integer.parseInt(s2);
int y = Integer.parseInt(s3);
System.out.println("sum = " + (x + y));
```

case 1 : Enter 2 No.

15, -25

O/P : 40

case 2 : Enter. 2 No.

15 15

# Exception : NoSuchElementException  
(Demo21.java : 13)

case 3 : Enter 2 No.

15, -25

: NumberFormatException  
(Demo21.java : 15)

# StringTokenizer st = new StringTokenizer(si, ", -");  
If we write like this so we can use ,  
as well as space .

# st.countTokens() :— with this we find the  
total number of Tokens and  
also we can find the remaining no. of  
Tokens.

written type of countTokens() method is Integer

# ① → Count the Tokens

{  
{

ISR i = new ISR (System.in);

BR br = new BR (i);

S.o.println("Enter 2 No.");

String s1 = br.readLine();

StringTokenizer st = new StringTokenizer(s1, ", -");

S.o.println(st.countTokens());

String s2 = st.nextToken();

String s3 = st.nextToken();

S.o.println(st.countTokens());

String s4 = st.nextToken();

S.o.println(st.countTokens());

String s5 = st.nextToken();

S.o.println(st.countTokens());

{  
}

Input : 10 20 30

O/p → Exception : NoSuchElementException  
Line - 15

# St.hasMoreTokens(); if we have tokens, so it will return true and if we dont have any token so, it will return false.

Example:

{  
}

ISR i = new ISR (s.in);

BR br = new BR (i);

s.o.println ("Enter a No.");

String s1 = br.readLine();

StringTokenizer st = new StringTokenizer (s1, ", -");

s.o.println (st.hasMoreTokens());

String s2 = st.nextToken(); // true

String s3 = st.nextToken(); // true

s.o.println (st.hasMoreTokens());

String s4 = st.nextToken(); // t

s.o.println (st.hasMoreTokens());

String s5 = st.nextToken();

{  
}

case : 1

Input  $\Rightarrow$  10 20 30

true

true

false

case : 2

Input  $\Rightarrow$  10 20 30 40

true

true

true

case : 3

Input  $\Rightarrow$  10 20

true

false

Exception : NoSuchElementException.

Q + Addition of numbers and take input from user.

```
import java.io.*;
import java.util.*;
{
{
```

```
ISR i = new ISR (System.in);
```

```
BR br = new BR (i);
```

```
System.out.println ("Enter number");
```

```
String s1 = br.readLine();
```

```
StringTokenizer st = new StringTokenizer (s1, "+ -");
```

```
int s = 0;
```

```
while (st.hasMoreTokens ())
```

Input : Enter no.  
10 20 30 40

```
{
String s3 = st.nextToken();
S = S + Integer.parseInt(s3);
}
s.o.println("Sum = " + S);
}
}
}
```

```

import java.util.*;
import java.io.*;

{
ISR i = new ISR(s.in);
BR br = new BR(i);
s.o.println("enter number");

String s1 = br.readLine();
StringTokenizer st = new StringTokenizer(s1, ", -");
int S = 0;
while (st.countTokens())
{
String s3 = st.nextToken();
S = S + Integer.parseInt(s3);
}
s.o.println("sum = " + S);
}
}
```

Error: incompatible types: int cannot be converted  
to boolean  
while (st. countTokens())  
^

# The method countTokens() does not take any parameter.

Use :- The method is used to return the number of tokens remaining in the string using the current delimiter set.

# delimiter  $\Rightarrow$  In Java, delimiters are the character that split (separate) the string into tokens. Java allows us to define any character as a delimiter.

Q Addition of numbers by countTokens() method.  
take input from user.

{  
{

```
ISR i = new ISR(s.in);
BufferedReader br = new BR(i);
S.o.ph("Enter number");
String s1 = br.readLine();
StringTokenizer st = new StringTokenizer(s1, "-");
int s = 0;
```

```
while (st. countTokens () > 0)
{
 String s3 = st. nextToken ();
 s = s + Integer.parseInt (s3);
}
s. o. println ("sum = " + s);
```

10/12/21

# यह स्ट्रिंग टॉकनाइजर का परामिती को पास करता है, जोकि डेलिमिटर कहते हैं।  
# Addition of n numbers with from st. countTokens() method: take input from user.

```
{
 s. o. println ("Enter numbers");
 String s1 = br. readLine ();
 StringTokenizer st = new StringTokenizer (s1, ", -");
 int s = 0, j = 0;
 int n = st. countTokens ();
 while (j < n)
 {
 String s3 = st. nextToken ();
 s = s + Integer.parseInt (s3);
 j++;
 }
 s. o. println ("sum = " + s);
```

- # Scanner class is in util package.
- # Scanner class has next method.

Ex: take input from user:

```
import java.util.*;
class Demo<?>
{
 public void m(String [] ar)
 {
```

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

```
 System.out.println("Enter name");
```

~~```
        String.out.println(
```~~

```
        String s1 = sc.next();
```

```
        System.out.println("name = " + s1);
```

```
}
```

```
}
```

case 1:

input : Enter name

Ram

O/P : name = Ram.

case 2:

input : Enter name

Ram ji

O/P : Ram

next method only space का जो सीधा data
head करती है, and space को split करती है।

If return type of next method is String.
Example:

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

```
@ class demo
```

```
{
```

```
public void m (String [] ar)
```

```
{
```

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

```
S.o.println ("Enter 2 no.");
```

```
String s1 = sc.next();
```

```
String s2 = sc.next();
```

```
int x = Integer.parseInt (s1);
```

```
int y = Integer.parseInt (s2);
```

```
S.o.println ("Sum = "+(x+y));
```

```
}
```

```
}
```

~~nextInt~~ nextInt() \Rightarrow ① method Integer value
की गेड करता है।

Example:

```
import java.util.*;  
  
class Demo  
{  
    public static void main (String args)  
    {
```

```
        Scanner sc = new Scanner (System.in);  
        System.out.println ("Enter 2 no.");  
        int x = sc.nextInt ();  
        int y = sc.nextInt ();  
        System.out.println ("Sum = " + (x+y));  
    }  
}
```

Scanner class के पास nextInt, next~~Char~~, nextDouble,
nextLong, nextShort, nextByte, nextFloat methods हैं।

Example :

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

```
{
```

```
}
```

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

```
s.o.println("Enter 2 no.");
```

```
int x = sc.nextInt();
```

```
int y = sc.nextInt();
```

```
s.o.println("Sum = " + (x+y));
```

```
}
```

```
input:
```

case 1 : 12 98

Exception : InputMismatchException.

- Example :

```
{
```

```
{
```

```
int x[] = { 10, 20, 30, 40, 50 };
```

```
for (int i = 0; i < 6; i++)
```

```
{
```

```
s.o.println(x[i]);
```

```
}
```

```
}
```

O/P ⇒ 10 40
 20 50

Exception : ArrayIndexOutOfBoundsException

foreach loop :- automatic executable loop is called
Ex:- foreach loop.

purpose of foreach loop is to display / traverse
Example : data of array.

{
}

int x[] = {10, 20, 30, 40, 50, 60, 70};

for (int i : x)

{
}

System.out.println();

{
}
}
}
}

O/P ⇒ 10
20
30
40
50
60
70

Explanation :

x पहले index no. zero की value i को देगा । and i उसको display करवा देगा । then x index no. 1 की value i को देगा and i उसको display करवा देगा ।
ऐसी ही foreach loop index no. zero से last index तक की value i को देगा । and i उसको display करवा देगा ।

Example :

{
}

String s1[] = {"ram", "sita", "gita", "abhu"};

for (String i : s1)

{

System.out.println(i);

}

O/p ⇒ ram
sita
gita
abhu

Collection of 1D array is called 2D array.

for storing the data in order of rows and columns than we have to use 2D array.

for presenting the data in the form of matrix. we have to use 2D array.

For memory allotting :

D.T. V.O.N. [] [] = new D.T. [R][C];

int x[][] = new int [4][3];

| | | |
|----|----|----|
| 50 | 51 | 52 |
| 60 | 61 | 62 |
| 70 | 71 | 72 |
| 80 | 81 | 82 |

- # indexing is very important in array.
- # collection of values is called 1D array.
- # in case of 1D array . we do indexing of values.
- # indexing :

| | | | | |
|-----|----|----|----|----|
| 0 → | 00 | 01 | 02 | 03 |
| 1 → | 10 | 11 | 12 | 13 |
| 2 → | 20 | 21 | 22 | 23 |
| 3 → | 30 | 31 | 32 | 33 |

Example :

{

{

int x[3][3] = {

{50, 59, 52},

{60, 61, 62},

{70, 71, 72},

{80, 81, 82},

};

S.o. pln (x[0][0]),

O/p ⇒ 52
70

Exception :

AIOOB

```
S.o.println(x[2][0]);  
S.o.println(x[4][1]);
```

{

}

Example : Addition of number take input from user.

```
import java.util.*;  
import java.io.*;  
class hasMoreTokens  
{  
    public void main (String args) throws IOException  
{  
        InputStreamReader i = new InputStreamReader (System.in);  
        BufferedReader br = new BufferedReader (i);  
        System.out.print ("Enter number of your choice ");  
        String s1 = br.readLine();  
        StringTokenizer st = new StringTokenizer (s1, "-,\n\t");  
        int s = 0;  
        while (st.hasMoreTokens())  
        {  
            String s3 = st.nextToken();  
            s = s + Integer.parseInt (s3);  
        }  
        System.out.println ("sum = " + s);  
    }  
}
```

Case 1:

Input : Enter no. of your choice

10 20 30

O/P \Rightarrow Sum = 60

Case 2:

Input : Enter no. of your choice

10\n20 30

O/P \Rightarrow Sum = 60

Case 3:

Input : Enter no. of your choice

10\n20\t30

O/P \Rightarrow Sum = 60

Case 4:

Input : Enter no. of your choice

O/P \Rightarrow Sum = 0

11/12/21

Example :

{
}

int x[] = {10, 20, 30, 40, 50};

int y[] = new int[5];

int i;
for (~~i=0~~; i<5; i++)

{

y[i] = x[i];

```
for(i: x)
```

```
{
```

```
s.o.println("array y => ");
```

```
s.o.print(i);
```

```
{
```

```
{
```

```
{
```

O/p \Rightarrow 10 20 30 40 50

Example :

```
{
```

```
{
```

```
int x[] = {10, 20, 30, 40, 50};
```

```
int y[] = new int[5];
```

```
y = x;
```

```
s.o.println("array y => ");
```

```
for(int i:y)
```

```
{
```

```
s.o.println(i);
```

```
{
```

```
{
```

```
{
```

O/p \Rightarrow array y \Rightarrow p

10 20 30 40 50

~~scratch~~

Example :

```
{  
{  
int x[] = { 10, 20, 30, 40, 50 };  
int y[] = new int[5];  
y = x;  
S.o.pn ("array y => ");  
for(int i : x)  
{  
S.o.pn (i + " ");  
}  
}  
}  
O/p => 10 20 30 40 50
```

Example :

```
{  
{  
int x[] = { 10, 20, 30, 40, 50 };  
int y[] = new int[4];  
y = x;  
S.o.pn ("array y => ");  
for(int i : y)  
{  
}
```

```
S.o.println(i + " ");  
}  
}  
}
```

integer type array
can not be
converted to normal
variable.

O/P → 10 20 30 40 50

Example:

```
{  
{  
    int x[] = {10, 20, 30, 40, 50};  
    int y;  
    y = x;  
    for (int i : y)  
    {  
        S.o.println(y);  
    }  
}  
}
```

error: incompatible types: int[] cannot be converted
to int.

y = x;

Example:

```
{  
{  
    int x[] = {10, 20, 30, 40, 50};  
}
```

```
int y[];  
y = x;  
for (int i : y)  
{
```

```
s.o.println(i);  
}  
}
```

O/P \Rightarrow 10, 20 30 40 50

Example of Imp

```
{  
{  
int x[] = {10, 20, 30, 40, 50};  
int y[] = new int[5];  
for (int i = 0; i < 5; i++)  
{  
    y[i] = x[i];  
}
```

x[2] = 55;

s.o.println("array x is ");

```
for (int i : x)  
{
```

```
s.o.println(i + " ");
```

{

```
s.o.println(1);
```

```
s.o.println();
```

```
s.o.println("array y =>");
```

```
for (int i : y)
```

{

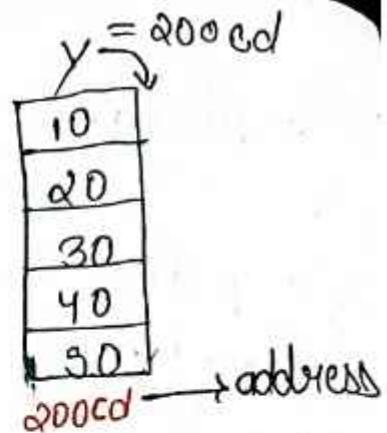
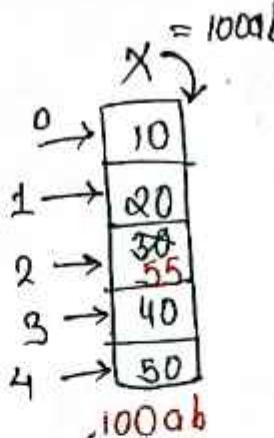
```
s.o.println(i + " ");
```

{

{

O/p \Rightarrow array x \Rightarrow 10 20 30 40 50

array y \Rightarrow 10 20 30 40 50



address $x \Rightarrow$ 10 20 30 40 50
y \Rightarrow 10 20 30 40 50

Example : Imp

{
{

```
int x[] = {10, 20, 30, 40, 50};
```

```
int y[] = new int [5];
```

```
y = x;
```

```
x[2] = 55;
```

```
s.o.println("array x =>");
```

```
for (int i : x)
```

{

```
i.o.println(i + " ");
```

```
}
```

```
i.o.println();
```

```
i.o.println();
```

```
i.o.println("array y =>");
```

```
for (int i : y)
```

```
{
```

```
i.o.println(" " + i + " ");
```

```
}
```

O/p \Rightarrow array x \Rightarrow 10 20 55 40 50

array y \Rightarrow 10 20 55 40 50

Example:

```
{
```

```
}
```

```
int x[3][2] = {{10, 20, 30}, {40, 50, 60}};
```

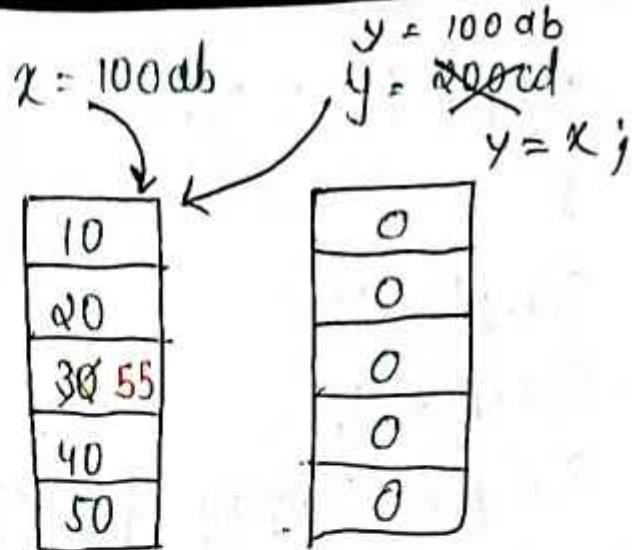
```
for (int i : x)
```

```
{
```

```
i.o.println(i);
```

```
}
```

```
}
```



error: incompatible types: int x[] can not be converted to int.

Example: display 2D array from foreach loop.

```
{  
    {  
        int x[][] = {{10, 20, 30}, {40, 50, 60}};  
        for (int i[] : x)  
        {  
            for (int j : i)  
            {  
                System.out.print(j + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

O/P ⇒ 10 20 30
 40 50 60

18/12/12

Irregular Array :- जिसके dimensions fix नहीं होती, तो irregular array

होती है।

case 1 :

{
}

int x [][] = new int [3][5];

{
}

case 2 :

int x [][] = new int [·][5];

case 3 :

int x [][] = new int [3][·];

Case 3 will be execute.

case 2 will not be executed. because 2D array उनाने पर 1D array बनाना compulsory होता है, अब भी define करना compulsory हो।

list of 1D array is called as 2D array. So, 1D array \rightarrow 2D array

case 4 :

int []_x[] = new int [3][5]; { will be execute }

case 5 :

int [][] x = new int [3][5]; { will be execute }

case 6 :

int [][]_x = new int [3][5]; { will be execute }

case 7 :

int _ [][]x = new int [3][5]; { will be execute }

case 8 :

int _ []x[] = new int [3][5]; { will be execute }

यदि 1st argument दिया गया लेकिन 2nd argument नहीं दिया गया है, तो runtime पर exception आ जाएगी।

{ }

int x[][] = new int [3][];

for (int i[] : x)

{ }

for (int i : i)

{ }

S. O. P (i + "\t");

}

```
s.o.p();
```

```
{  
}  
}
```

Exception :

Example :

```
{  
}  
}
```

```
int x [][] = new int x[3][];
```

```
x[0] = new int [2];
```

```
x[1] = new int [5];
```

```
x[2] = new int [3];
```

```
for (int i[] : x)
```

```
{  
}
```

```
for (int j : i)
```

```
{  
}
```

```
s.o.p(j + " );
```

```
{  
}
```

```
s.o.p();
```

```
{  
}  
}
```

O/p → 0 0
0 0 0 0 0
0 0 0

Syntax of Singly array :-

int x[] = new int[3];

- ये syntax java ने हमें कैसीली प्रोवाइड करता है, ताकि हम सिंगलर एरेय को संहेस्प-
ेंडिंग मेमोरी क्रेट कर सकें।

In 1D array :-

Syntax :

i) int x[] = new int[3]; ये will be execute ?

ii) int []x = new int[3]; ये will be execute ?

iii) int[] x = new int[3]; ये will be execute ?

iv) int[] x = new int[] {10, 20, 30, 40, 50};
ये will be execute ?

Example :

{
 {
 int []x = new int[] {10, 20, 30, 40, 50};
 ^ (dimension)

 for (int i : x)

 {
 System.out.print(i + " ");
 }

}

O/P → 10
20
30
40
50

Example : example of anonymous

```
{  
    int [] x = new int [5] {10, 20, 30, 40, 50};  
}
```

```
for (int i : x)
```

```
{  
    System.out.println(i);  
}
```

error : array creation with bad dimension expression and initialization.

we can not pass anything ~~any~~ in dimension.

Example :

```
{  
    int [] x = {50, 51, 52, 53, 54};  
    System.out.println(x);  
}
```

$\text{o/p} \Rightarrow [1024046046]$. \rightarrow ~~एक System में o/p different
माना है।~~

Example :

```
float [] - y = {50, 51, 52, 53, 54};
```

```
S.o.println(y);
```

Example :

```
byte [] - y = {50, 51, 52, 53, 54};
```

```
S.o.println(y);
```

Example :

```
char[] x = {50, 51, 52, 53, 54};  
long[] y = {50, 51, 52, 53, 54};  
// boolean[] z = {50, 51, 52, 53, 54};  
short[] z = {50, 51, 52, 53, 54};  
  
S.o.println(x);  
S.o.println(y);  
S.o.println(z);  
S.o.println(x1);  
y  
z  
x
```

Syntax of anonymous array :

```
int [] x = new int [] {10, 20, 30, 40, 50};
```

Example : length of array

{
}

```
int x[] = new int [5];
```

```
S.o.pln (x.length);
```

{
}

O/p \Rightarrow 5

for finding the length of array . we have to use variable.length .

Example :

{
}

```
int y [] [] = new int [3][5];
```

```
S.o.pln (y.length);
```

{
}

O/p \Rightarrow 3

Example :

```
{  
int x[] = new int[5];  
int y[][] = new int[3][5];  
S.o.println(x.length);  
S.o.println(y.length);  
S.o.println(y[0].length);  
}  
O/p => 5  
      3  
      5
```

Example :

```
{  
int y[][] = new int[3][5];  
S.o.println(y[0][0].length);  
}
```

~~error: int can not be dereferenced.~~

S.o.println(y[0][0].length);

Err only single element of pass off ~~we can~~.
~~we~~ we can know the size of array.

Example : display triangular array with for loop.

```
{  
    int i, j;  
    int y [][] = {{10, 20, 30, 40, 50}, {2, 5, 8, 9}, {1, 3}};  
    for (i = 0; i < y.length; i++)  
    {  
        for (j = 0; j < y[i].length; j++)  
        {  
            System.out.print(y[i][j] + " ");  
        }  
        System.out.println();  
    }  
}
```

Output
10 20 30 40 50
2 5 8 9
1 3

Example:

```
{  
    int i, j;  
    int x[3][5] = { { 10, 20, 30, 40, 50 }, { 6, 7, 8, 9, 3 }, { 3, 5, 3 } };
```

```
for (i = 0; i < 3; i++)
```

```
{
```

```
    for (j = 0; j < 5; j++)
```

```
{
```

```
        cout << x[i][j];
```

```
}
```

```
    cout << endl;
```

```
{
```

```
{
```

```
{
```

O/P \Rightarrow 10 20 30 40 50

2 5 8 9 Exception AIOOBET

```
{
```

```
{
```

```
#int i, j;
```

```
int y[3][3] = { { 1, 2, 3, 4, 5 }, { 6, 7, 8, 9, 3 }, { 3, 1, 3 } };
```

```
cout << y[0].length();
```

```
cout << y[1].length();
```

```
cout << y[2].length();
```

3
3

O/p \Rightarrow 5
4
2

In 1D array we can store more than 1 value in one variable.

3D array: — list of 2D array is called 3D array.

Ex -

| | | | | |
|---|---|-----|-----|-----|
| 0 | 0 | 000 | 001 | 002 |
| 1 | 1 | 010 | 011 | 012 |
| 2 | 2 | 100 | 101 | 102 |

| | | | | |
|---|---|-----|-----|-----|
| 1 | 0 | 100 | 101 | 102 |
| 2 | 1 | 110 | 111 | 112 |

| | | | | |
|---|---|-----|-----|-----|
| 2 | 0 | 200 | 201 | 202 |
| 3 | 1 | 210 | 211 | 212 |

3D array में indexing 2D array की होती है।

Example of 3D array:

{
}

Put $y[][][] = \{$

$\{ \{ 60, 61, 62 \}, \{ 63, 64, 65 \} \},$

$\{ \{ 60, 61, 62 \}, \{ 63, 64, 65 \} \},$

{ { 70, 71, 72 }, { 73, 74, 75 } }

S.o. p1, (y [1][0][2]);

S.o. p1, (y [2][1][1]);

}

}

O/p \Rightarrow 62
74

Example :

{

{

int y [] [] [] = {

{ { 50, 51, 52 }, { 53, 54, 55 } },

{ { 60, 61, 62 }, { 63, 64, 65 } },

{ { 70, 71, 72 }, { 73, 74, 75 } }

};

S.o. p1, (y [1][0].length);

S.o. p1, (y [2].length);

S.o. p1, (y.length);

}

}

O/p \Rightarrow 3
2
3

memory allocation of 3D array.

int x[3][3][3] = new int [3][3][3];

no. value of 3D array
no. value of 2D array
value of 1D array in 2D array
value of 3D array

Example : display 3D array with for loop.

{
{

int x[3][3][3] = {

{50, 51, 52}, {53, 54, 55},

{60, 61, 62}, {63, 64, 65},

{70, 71, 72}, {73, 74, 75}

} ;

for (int i=0; i<3; i++)

{

for (int j=0; j<3; j++)

{

for (int k=0; k<3; k++)

{

S.o.p(x[i][j][k]);

}

S.o.println();

{

S.o.println();

S.o.println(); } }

O/P \Rightarrow 50 51 52
53 54 55

60 61 62
63 64 65

70 71 72
73 74 75

Example: display 3D array with foreach loop.

{
}

```
int x[3][3][3] = {  
    {{50, 60, 70}, {80, 90, 100}},  
    {{61, 62, 63}, {64, 65, 66}},  
    {{71, 72, 73}, {74, 75, 76}}  
};
```

```
for (int i[3][3] : x[3])
```

{

```
    for (int j[3] : i)
```

{

```
        for (int k : j)
```

{

```
            S.o.p(k + " ");
```

{

```
            S.o.println();
```

{

```
            S.o.println();
```

{

```
            S.o.println();
```

{

```
            S.o.println();
```

{

O/p \Rightarrow 50 60 70
80 90 100

61 62 63

64 65 66

71 72 73

74 75 76

O/p \Rightarrow

Command Line Argument

java demo26 123 abc fkjh. यह program will
 O/p ⇒ ram. definitely run in this situation.

अगर कमांड time का भी value पास जरूर हो तो
 ने अ के पास आकर space से separate होने
 चाहे हो भागी।

Example :

```
class demo26 {
```

```
    public void main (String [] args) {
```

```
        System.out.println ("ram");
```

```
        System.out.println ("args.length");
```

```
}
```

case 1 : java demo26 abc 123 hd

O/p ⇒ ram
 3

case 2 : java demo26

O/p ⇒ ram
 0

Example :

```
class demo26  
{
```

```
    public static void main (String [] ar)  
{
```

```
        String s1 = ar[0];
```

```
        String s2 = ar[1];
```

```
        System.out.println ("name = " + s1);
```

```
        System.out.println ("pass = " + s2);
```

```
}
```

case 1 : java demo26 ram 123 *at run time*

O/p \Rightarrow name = ram
pass = 123

case 2 : java demo26 ram

Exception : AIOOBE

Example : Addition of two numbers from command Line Argument.

```
{  
{
```

```
    String s1 = ar[0];
```

```
    String s2 = ar[1];
```

```
int x = Integer.parseInt(s1);
int y = Integer.parseInt(s2);
System.out.println("Sum = " + (x+y));
```

case 1 : java demo26 10 20
O/p \Rightarrow Sum = 30

Exam : Addition of n numbers from CLA.

```
int i, s=0;
for (i=0; i<ar.length; i++)
{
    String s1 = ar[i];
    int x = Integer.parseInt(s1);
    s = s + x;
}
System.out.println(s);
```

case 1 : java demo26 10 20 30 40
O/p \Rightarrow Sum = 100

case 2: java demo 26 10 20 40

O/p \Rightarrow 70

Case 3: java demo 26 10 20

O/p \Rightarrow 30

case 4: java demo 26 50

O/p \Rightarrow 50

Example :

```
{  
{
```

```
int i, s=0;
```

```
for (i=0; i<ar.length; i++)
```

```
{
```

```
s = s + Integer.parseInt(ar[i]);
```

```
}
```

```
s.out(s);
```

```
}
```

```
}
```

case 1: java demo 10 20 30 40

O/p \Rightarrow 100

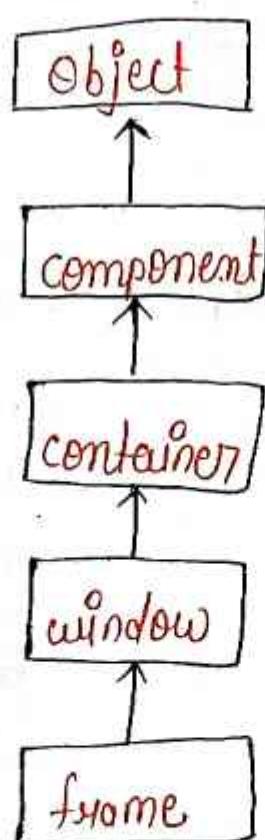
~~String~~ ~~int~~ ~~100~~.

Graphics

GUI → Graphical User Interface

AWT → Abstract window Toolkit.

- A AWT provide classes and Interface. and with the help of that classes and Interface, we write code in Graphics.
- All tools ~~are~~ of graphics are available in awt.
- Super class of java is object.



→ Top level classes are called object.

Frame → If by default frame is invisible.

If we want to use frame. So, you have to inherit the frame class.

- iii) Extend means inherit object
- iv) with the help of extends keyword. we can inherit the class.
- v) frame class is predefined in awt package.

Import java.awt.frame;

```

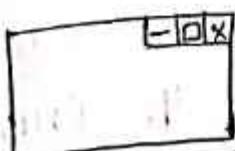
class FDemo extends Frame {
}
FDemo() {
}
}

class demo6 {
}
public void main (String ar[]) {
}
FDemo f = new FDemo();
f.setVisible (true);
// f.setSize (w,h);
f.setSize (500,300);
// f.setLocation (x,y);
f.setLocation (100,200);
}
}
}

// we use ctrl c for
close the frame.

```

O/P ⇒



→ f. setVisible (true) → It is used to visible the frame.

by default inside f. setVisible (false) is false. and by default the frame is invisible. for visibility we have to use true inside f. setVisible (true);

→ f. setSize (w, h); → It is used for set the size width height of frame.

→ f. setLocation (x, y); → It is used for set the location. In which axis you want to set the location of frame.

→ ~~Paint~~ method → when frame is loaded ~~paint~~ method is automatically called.

i) ~~paint~~ method को explicitly call करने की जरूरी नहीं होती।

ii) ~~paint~~ method is provided by Frame class.

#

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Color;

→ user defined class

```
class FDemo extends Frame {  
    public void paint(Graphics g) {  
        g.drawString("Xam", 100, 200);  
    }  
}  
  
class demo {  
    public static void main(String args) {  
        FDemo f = new FDemo();  
        f.setVisible(true);  
        f.setSize(500, 300);  
        f.setLocation(100, 200);  
        f.setBackground(Color.red);  
        f.setForeground(Color.white);  
    }  
}
```

→ Set Background → It is used for set the background color.

→ SetForeground → It is used for set the color.

- Graphics class is in awt package.
- Graphics class has drawString method.

16/12/21

```
import java.awt.Frame;  
import java.awt.Graphics;  
import java.awt.Color;  
  
class FDemo extends Frame  
{  
    FDemo()  
    {  
        setVisible(true);  
        setSize(500, 300);  
        setLocation(100, 200);  
        setBackground(Color.red);  
        setForeground(Color.white);  
    }  
    public void paint(Graphics g)  
    {  
        g.drawString("ram ji", 100, 100);  
    }  
}  
class Demo26  
{  
    public static void main()  
    {  
    }  
}
```

```
FDemo f = FDemo();  
{  
}
```

① g.drawString() →

Example :

```
import java.awt.Frame;  
Graphics;  
Color;  
Font;
```

```
class FDemo extends Frame  
{
```

```
FDemo()  
{
```

```
Font f = new Font("Black-  
Kader ITC", Font.ITALIC,  
50);
```

```
setFont(f);
```

```
}
```

```
public void paint(Graphics g)  
{
```

```
g.drawLine(100, 100, 300, 300);
```

```
g.drawRect(100, 100, 200, 200);
```

```
g.drawRoundRect(100, 100, 200, 200, 50, 50);
```

```
g.drawoval(x, y, w, h);
```

② setFont :- It is used to increase the size, and changing the font language. Font class is in awt.Font package.

Syntax :

```
Font f = new Font("Black-  
adder ITC.", Font.ITALIC  
50);
```

```
SetFont(f);
```

```
g.drawLine(100, 100, 200, 200, 90,  
          270);
```

③ → g.drawLine(); ⇒ It is used to print diagonal.

```
g.drawInt x[] = {100, 200, 300};
```

Syntax:

```
g.drawLine(x1, y1, x2, y2)
```

```
int y[] = {200, 100, 400};  
int n = 3;
```

```
g.drawPolygon(x, y, n);
```

```
int x[] = {200, 300, 400, 300, 200,  
          100};
```

```
int y[] = {100, 100, 200, 300, 300,  
          200};
```

```
int n = 6;
```

```
g.fillPolygon(x, y, n);
```

If we write $n=7$, then at run time an exception will occur.

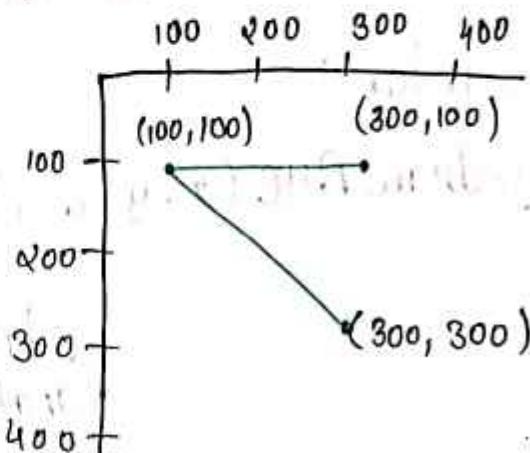
Exception:

If we write $n=5$, then

fill n if will be filled. */

```
g.setColor(Color.Blue);
```

```
g.fillRect(100, 100, 300, 300);
```



④ → g.drawRect() ⇒ It is used to print rectangle.

Syntax:



```
g.drawRect(x, y, w, h);
```

width height

⑤ → g.drawRoundRect() ⇒ It is used to print rounded rectangle

Syntax:



```
g.drawRoundRect(x, y, w,  
                 h, xR, yR)
```

⑥ → g.drawoval () ⇒ It is used to print circular shape.

Syntax :

g.drawoval (x, y, w, h);



⑦ → drawArc() ⇒ It is used to print arc.

Syntax :

g.drawArc (x, y, w, h, s_a, m_a);

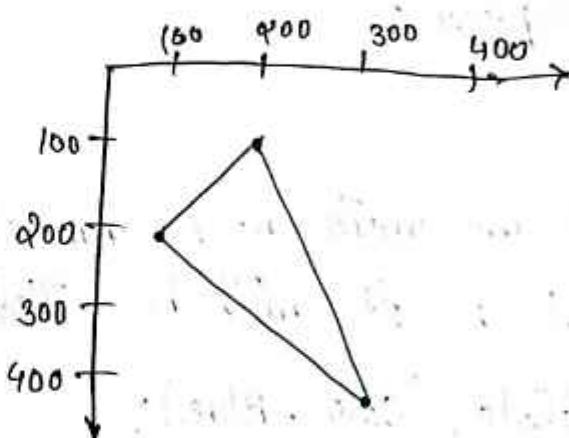


Starting angle moving angle

⑧ → g.drawPolygon →

Syntax :

g.drawPolygon (x[], y[], n);



⑨ → g. ~~draw~~ g. fillPolygon() →

Syntax :

g. fillPolygon(x, y, n);

⑩ → g. fillRect() → It is used for filling the rectangle :

Syntax :

g. fillRect(x, y, w, h);

→ g. color() → for filled shape.

Syntax :

g. Color (color. Color name);

for making color : —

Color c = new Color (r, g, b);

Color c = new Color (176, 88, 0);

⑪ → g. setColor(c);

g. fillRect (100, 100, 300, 100);

orange.

→ g. fillRoundRect →

→ fill oval →

17/12/21

→ for Arc (Ashok chakra) —

int so = 0;

int ma = 5;

for (int i = 0; i < 24; i++)

{

g.fillArc(100, 100, 200, 200, so, ma);

so = so + 15;

}

#

class Demo

{

p.s.v.m()

{

↳ S.o.println(Math.max(10, 20));

0/p → 20

↳ S.o.println(Math.min(10, 20));

10

↳ S.o.println(Math.cbrt(64));

4

↳ S.o.println(Math.sqrt(64));

8

↳ S.o.println(Math.pow(2, 5));

32

↳ S.o.println(Math.sin(10));

-0.1736

↳ S.o.println(Math.cos(10));

0.1736

↳ S.o.println(Math.tan(10));

1.5574

↳ S.o.println(Math.random());

0.4236

```
L→ S.o.println(Math.round(10.7)); // 11  
S.o.println(Math.round(10.3)); // 10  
S.o.println(Math.round(10.49)); // 10  
S.o.println(Math.round(10.50)); // 11  
S.o.println(Math.round(Math.random()));  
} } }
```

Print random no. b/w 1 to 5 :

```
{  
}
```

```
{  
}
```

```
S.o.println(Math.round(Math.random() * 5));
```

```
{  
}
```

```
}
```

```
/* 0.0 * 5 ⇒ 0  
 0.1 * 5 ⇒ 1  
 0.2 * 5 ⇒ 1  
 0.3 * 5 ⇒ 2  
 0.4 * 5 ⇒ 2.  
 0.5 * 5 ⇒ 2  
 0.6 * 5 ⇒ 3  
 0.7 * 5 ⇒ 4  
 0.8 * 5 ⇒ 4  
 0.9 * 5 ⇒ 5  
 */
```

It is compulsory to write 1st character
in string in drawString method.

written type of round method is long.

~~class~~ import java.awt.*;

```
class FDemo extends Frame {  
    FDemo ()  
    {  
        Font f = new Font ("Blackadder ITC", Font.ITALIC,  
                           50);  
        setFont (f);  
    }  
  
    public void paint (Graphics g)  
    {  
        int i, j;  
        int x = 100;  
        int y = 100;  
        for (i=1; i<=10; i++)  
        {  
            for (j=1; j<=10; j++)  
            {  
                try { Thread.sleep (100); } catch (Exception e) {}  
                int r1 = (int) Math. round (Math. random () * 255);  
                int g1 = (int) Math. round (Math. random () * 255);  
                int b1 = (int) Math. round (Math. random () * 255);  
                g.setColor (new Color (r1, g1, b1));  
                g.drawString ("*", i * 20, j * 20);  
            }  
        }  
    }  
}
```

```
color c = new Color (r1,g1,b1);
```

```
g. setColor(c);
```

```
g. drawString (" " + i + j, x, y);
```

```
x = x + 70;
```

```
}
```

```
y = y + 70;
```

```
x = 100;
```

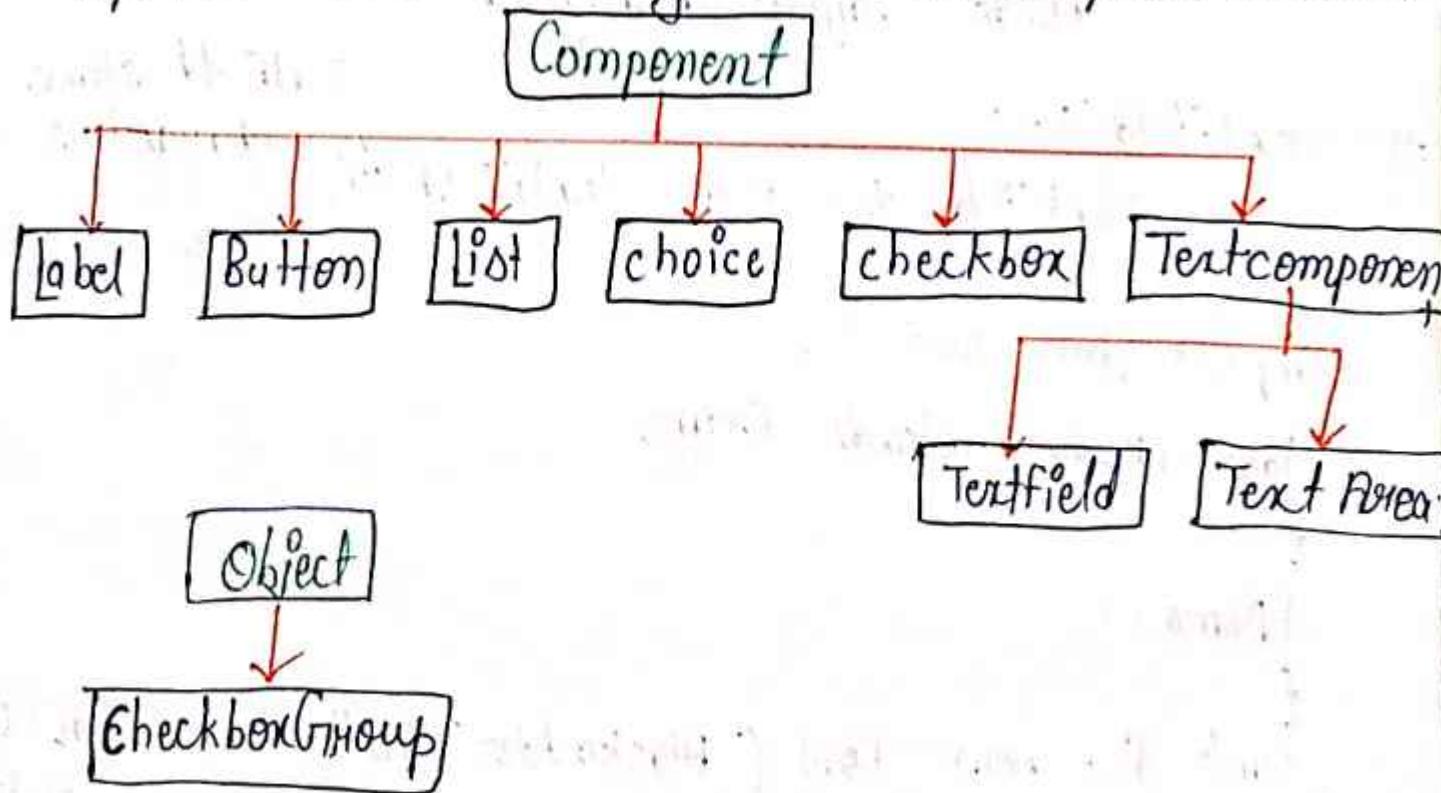
```
}
```

```
}
```

18/012/21

AWT Components :-

Component have a lot of sub classes / child classes.



- # जब भी हम कोई component के बनाते हैं, तो
add का use करना compulsory है। button को
add करने के लिए component को add करना com. है।
- # Frame के अंदर layout default Layout manager
है। border Layout manager होता है, means, जो
- # If component हम layout में add करते हैं
गे border में आकर add हो जाता है। component
- # way of arrange component is called Layout.

FlowLayout() → FlowLayout परिवर्ती content की size
है, उनी एवं @ button add कर देगा।

→ It is compulsory to add component in Frame.

Layout Syntax: setLayout(new FlowLayout)

Layout → default ^{Layout} manager inside Frame is
border Layout manager.

TextField() →
TextField t = new TextField(10);

```
import java.awt.*;  
class FDemo extends Frame
```

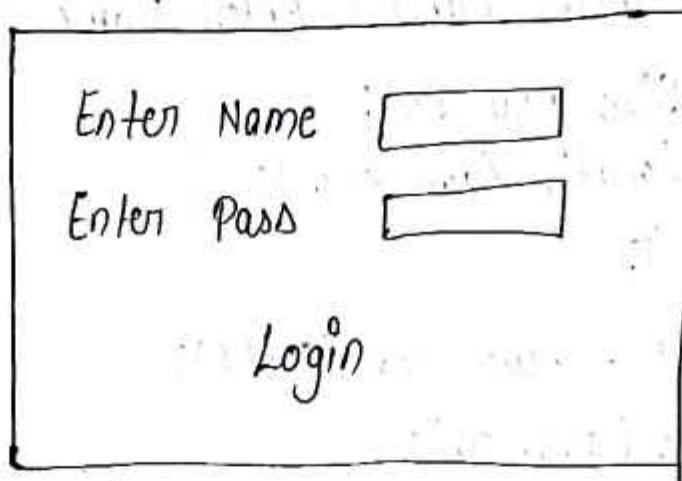
```
{
```

```
FDemo()  
{
```

```
Font f = new Font("Blackadder ITC", Font.ITALIC,  
"50");
```

```
setFont(f);
```

```
setLayout (new FlowLayout ());
Button b1 = new Button ("Login");
add (b1);
TextField t1 = new TextField ();
add (t1);
TextField t2 = new TextField (10);
add (t2);
}
```



→ If you want to count set component according to your need. so setLayout Null.

When you setLayout Null . then size will be zero.

In this case blank screen will be display.

```
import java.awt.*;
class FDemo extends Frame
{
    FDemo()
    {
        setTitle ("Login Page ");
        Font f = new Font ("Blockadder ", Font.ITALIC ,50);
        setFont(f);
        setLayout (null);
        Label un = new Label ("Enter Name ");
        un. setSize (300, 50);
        un. setLocation (100,200 );
        add(un);
        Textfield t1 = new Textfield ();
        t1. setSize (280, 50);
        t1. setLocation (430,200 );
        add(t1);
        Label up = new Label ("Enter pass");
        up. setSize (280,50);
        up. setLocation (100,300 );
        add (up);
        Textfield t2 = new Textfield ();
        t2. setSize (280, 50);
        t2. setLocation (430,300 );
        t2. setEchoChar('*');
```

```
odd(t2);
```

```
Bubble Button b1 = new Button ("Login");
```

```
b1.setSize(180, 60);
```

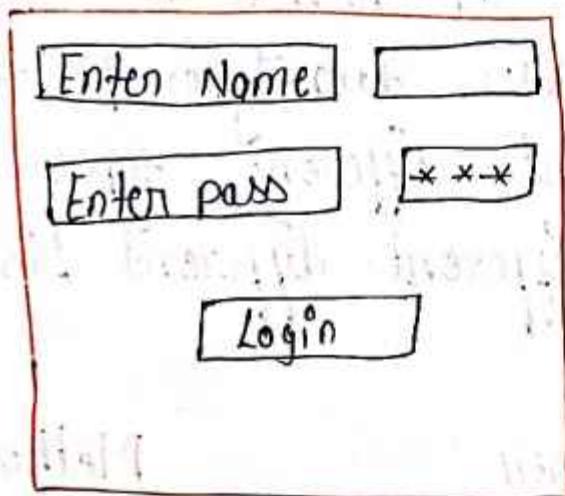
```
b1.setLocation(240, 410);
```

```
add(b1);
```

```
8 setResizable(false);
```

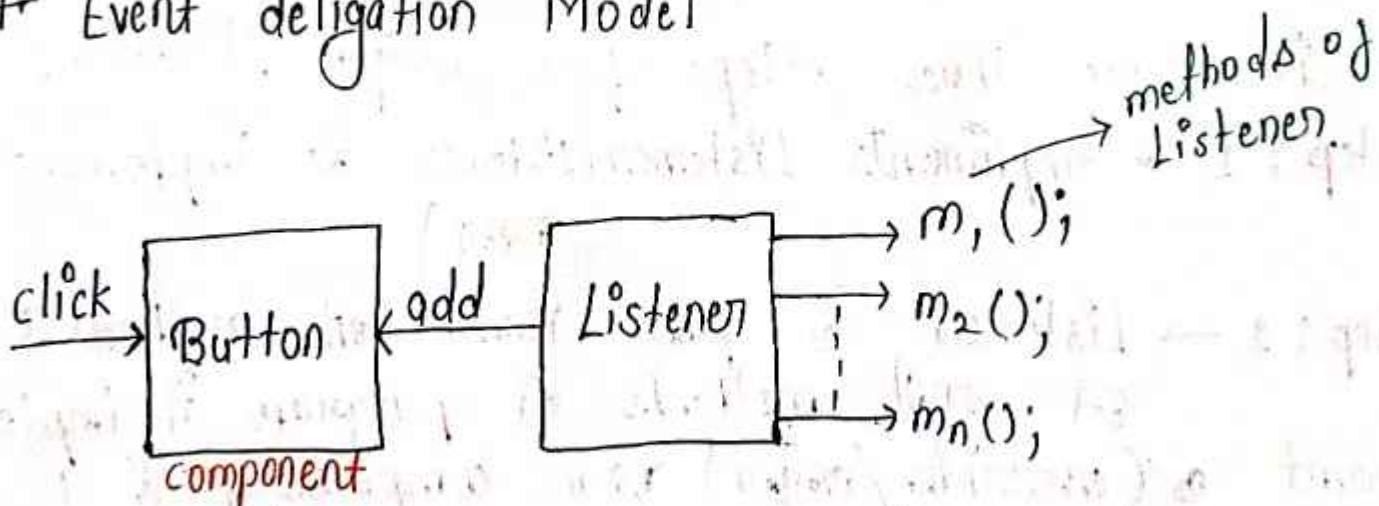
```
}
```

```
}
```



20/12/21

#^{IMP} Event delegation Model



→ Listener has a lot of methods.

→ for performing operation on click on Button
we use Listener.

For clicking on Button, we will have to add the listener to Button. So when we click Button then Listener will be added and method of Listener will be executed. and This diagram is called Event delegation Model.

→ Listener is a interface.

Event handling —

जब अलग - अलग type की values को store करने के लिए डिग्रा - अलग data types provide किए हैं। वैसे ही different - different types के components के लिए different - different Listener provide किए हैं।

| Component | Listener | Method |
|---|----------------|-----------------|
| Button | ActionListener | ActionPerformed |
| → There are three steps for using Listener. | | |
| • step : 1 → Implements Listener (Listener को implement करना) | | |
| • step : 2 → Listener के pass function के methods हैं। उन सारी methods को program में implement (override / बदला) करना compulsory है, otherwise program will not compile. | | |
| • step : 3 → In which component we want to perform operation . we have to add Listener in | | |

that component.

Like we have 3 Button (b₁, b₂, b₃) and we want to perform operation in only two (b₁, b₂). Do we have to add Listener in that 2 Button (b₁, b₃). It is compulsory.

→ ActionListener is in awt.event.*; package.

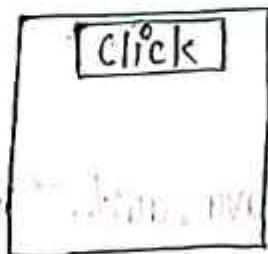
Example :

```
import java.awt.*;
import java.awt.event.*;
class FDemo extends Frame implements ActionListener
{
    FDemo()
    {
        Font f = new Font("Blackadder", Font.ITALIC, 850);
        setLayout(f); setFont(f);
        setLayout(new FlowLayout());
        Button b1 = new Button("click");
        add(b1);
        b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        // System.out.println("ram");
        setBackground(Color.red);
    }
}
```

```

class Demo
{
    public static void main(String ar[])
    {
        FDemo f= new FDemo();
        f.setVisible(true);
        f.setSize(1000, 800);
        f.setLocation(100, 200);
    }
}

```



→ इस जो ~~है~~ ~~कि~~ Button पर click होते हैं, तभी
reference e के पास आ जाता है।

Example:

```

import java.awt.*;
import java.awt.event.*;
class FDemo extends Frame implements ActionListener
{
    Button b1, b2, b3; → Globally declared
    FDemo()
    {
        Font f= new Font("Blackadder ITC", Font.ITALIC, 50);
        setFont(f);
        setLayout(new flowLayout());
    }
}

```

```
b1 = new Button ("RRRR");
add(b1);
b2 = new Button ("G1G1G1G1");
add(b2);
b3 = new Button ("BBB");
add(b3);
```

```
b1.addActionListener(this);
```

b2. addActionListener(this);

b3. addActionListener (this);

3

```
public void actionPerformed(ActionEvent e)
```

5

```
if(e.getSource() == b1)
```

۲۷

```
setBackground(Color.red);
```

3

```
if(e.getSource() == b2);
```

9

set Bockg Hounds (color. green),
2

9

```
    if (e.getBackground() == null) {  
        e.setBackground(Color.blue);  
    }  
}
```

3

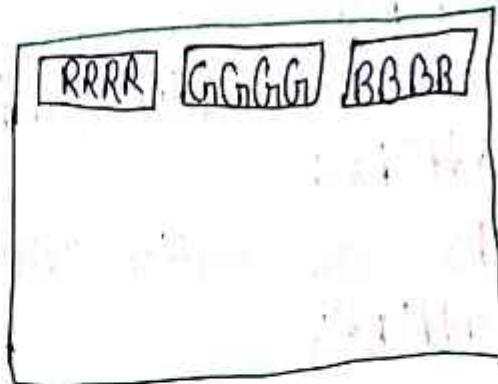
3

class Demo

۹۵

$p \in \text{v.m.}(\text{String arr})$

```
{  
FDemo f = new FDemo();  
f.setVisible(true);  
f.setSize(1000, 800);  
f.setLocation(100, 200);  
}  
}
```



Example :

```
import java.awt.*;  
import java.awt.event.*;  
class FDemo extends Frame implements ActionListener  
{  
Button b1, b2, b3;  
FDemo()  
{  
Font f = new Font("Blackadder ITC", Font.ITALIC, 50);  
setFont(f);  
setLayout(null);  
b1 = new Button(" ");  
set b1.setSize(100, 100);  
b1.setLocation(100, 100);  
add(b1);  
b2 = new Button(" ");  
b2.setSize(100, 100);  
b2.setLocation(200, 100);  
add(b2);  
}
```

```
b1.addActionListener(this);
b2.addActionListener(this);
}
public void actionPerformed(ActionEvent g)
{
if(g.getSource() == b1)
{
b1.setLabel("O");
}
if(g.getSource() == b2)
{
b2.setLabel("X");
}
}
}

class Demo
{
public static void main(String args)
{
FDemo f = new FDemo();
f.setVisible(true);
f.setSize(1000, 800);
f.setLocation(100, 200);
}
}
```

21/12/21

→ repaint() :- for calling paint method (Graphics)

Example :-

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
class fdemo extends Frame implements ActionListener
```

```
{ boolean b = false;
```

```
Button b1, b2, b3;
```

```
fdemo()
```

```
{
```

```
Font f = new Font ("Blackadder ITC", Font.ITALIC, 50);  
setFont (f);
```

```
setLayout (new FlowLayout ());
```

```
b1 = new Button ("rect");
```

```
add(b1);
```

```
b1.addActionListener (this);
```

```
{
```

```
public void paint (Graphics g)
```

```
{
```

```
if (b)
```

```
{
```

```
g.drawRect (100, 100, 100, 100);
```

```
}
```

```
public void actionPerformed(ActionEvent e)
```

```
{
```

```
b = true;
```

```
repaint();
```

```
}
```

```
class Demo
```

```
{
```

```
public static void main(String ar[])
```

```
{
```

```
f demo f = new f demo();
```

```
f. setVisible(true);
```

```
f. setSize(1000, 800);
```

```
f. setLocation(100, 100);
```

```
}
```

→ Text Field की data fetch करने के लिए getText() method

का use करते हैं।

→ Textfield से data लेने के लिए setTextfield() method का use करते हैं।

→ It is compulsory to pass String in setText() method.

Example :

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame & implements ActionListener
{
    Button b1, b2;
    TextField t1, t2;

    FDemo()
    {
        Font f = new Font("BL ITC", Font.ITALIC, 50);
        setFont(f);

        setLayout(new FlowLayout());

        t1 = new TextField(10);
        add(t1);

        b1 = new Button("Click");
        add(b1);

        t2 = new TextField(10);
        add(t2);

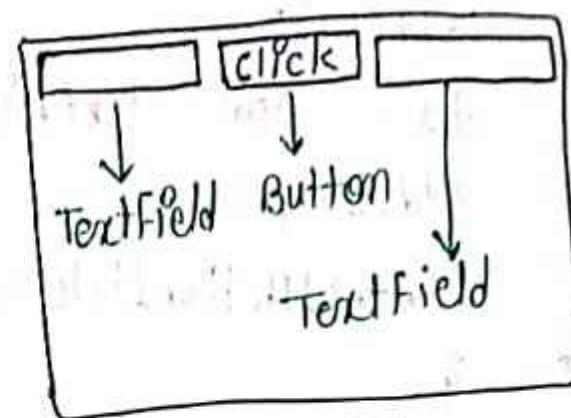
        b1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e)
    {
        String s1 = t1.getText();
        t2.setText(s1);
        t1.setText(" ");
    }
}
```

```

class Button
{
    public void actionPerformed(ActionEvent e)
    {
        Frame fDemo = new fDemo();
        fDemo.setSize(1000, 800);
        fDemo.setLocation(100, 100);
        fDemo.setVisible(true);
    }
}

```



Example :

```

import java.awt.*;
import java.awt.event.*;

class fDemo extends Frame implements ActionListener
{
    Textfield t1, t2, t3;
    Button b1;

    fDemo()
    {
        Font f = new Font("Blackadder ITC", Font.ITALIC, 50);
        setFont(f);
        setLayout(new FlowLayout());
        t1 = new Textfield(10);
        add(t1);
        t2 = new Textfield(10);
        add(t2);
    }
}

```

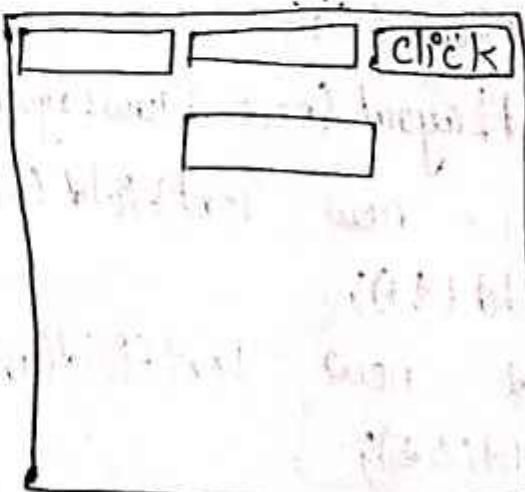
```

b1 = new Button ("Click");
add(b1);
t3 = new TextField(10);
add(t3);
b1.addActionListener(this);
}

public void actionPerformed(ActionEvent e)
{
    String s1 = t1.getText();
    String s2 = t2.getText();
    int x = Integer.parseInt(s1);
    int y = Integer.parseInt(s2);
    z = x + y;
    t3.setText(" " + z);
}

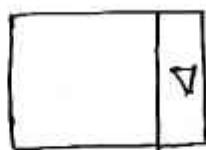
class Demo
{
    public void main (String[])
    {
        FDemo f = new FDemo();
        f.setVisible(true);
        f.setSize(1000, 800);
        f.setLocation(100, 100);
    }
}

```



Example :—

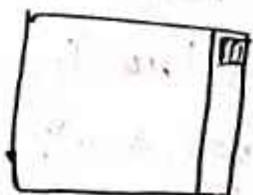
```
import java.awt.*;
import java.awt.event.*;
class FDemo extends Frame implements AL.
{
    Button b1, b2;
    FDemo()
    {
        Font f = new Font("Blackadder ITC", Font.ITALIC, 50);
        setFont(f);
        setLayout(new FlowLayout());
        Choice c1 = new Choice();
        c1.add("java");
        c1.add("Python");
        c1.add("js");
        c1.add("css");
        c1.add("php");
        add(c1);
    }
    class Demo
    {
        public void main(String ar[])
        {
            FDemo f = new FDemo();
            f.setVisible(true);
            f.setSize(1000, 800);
            f.setLocation(100, 100);
        }
    }
}
```



→ List means Scrolling view.

Example :

```
class Import java.awt.*;  
import java.awt.event.*;  
class FDemo extends Frame implements AL  
{  
    FDemo()  
    {  
        Font f = new Font("Blackadder ITC", Font.ITALIC, 50);  
        setFont(f);  
        setLayout(new FlowLayout());  
        List c1 = new List();  
        c1.add("java");  
        c1.add("python");  
        c1.add("php");  
        c1.add("js");  
        c1.add("css");  
        add(c1);  
    }  
}  
class FDemo  
{  
    public void main()  
    {  
        FDemo f = new FDemo();  
        f.setVisible(true);  
        f.setSize(1000, 800);  
    }  
}
```



```
f. setLocation(100,100);
```

```
{}
```

Example :-

```
import java.awt.*;  
import java.awt.event.*;  
class FDemo extends Frame implements AL  
{  
    FDemo()  
{  
        Font f=newFont("Blackadder ITC", Font.ITALIC, 50);  
        setFont(f);  
        setLayout(new FlowLayout());  
        checkbox cb1 = new checkbox("java");  
        add(cb1);  
        checkbox cb2 = new checkbox("python");  
        add(cb2);  
    }  
    class demo  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            FDemo f = new FDemo();  
            f.setVisible(true);  
            f.setLocation(100,100);  
            f.setSize(1000,800);  
        }  
    }  
}
```

java python

Example :-

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
class FDemo extends Frame implements AL  
{
```

```
    FDemo()
```

```
{
```

```
    Font f = new (
```

```
        setFont(f);
```

```
        setLayout(new FlowLayout());
```

```
        Checkbox cb1 = new Checkbox("java");
```

```
        add(cb1);
```

```
        Checkbox cb2 = new Checkbox("python", true);
```

```
        add(cb2);
```

```
}
```

```
}
```

```
class demo
```

```
{
```

```
    public void main (String args)
```

```
{
```

```
    FDemo f = new FDemo();
```

```
    f.setVisible(true);
```

```
    f.setSize(1000, 800);
```

```
    f.setLocation(100, 200);
```

```
    f.setBackground(Color.gray);
```

```
}
```

by default tick after display होता
 Java Python

→ Cols

```
import java.awt.*;
import java.awt.event.*;
class FDemo extends Frame implements AL
{
    Button b1, b2, b3, b4, b5, b6, b0, b;
    Label un, un2;
    TextField t1, t2;
    FDemo()
    {
        Font f = new Font ("blockadder. ITC", Font.ITALIC, 30);
        setFont (f);
        setLayout (new FlowLayout ());
        un = new Label ("Enter no. 1");
        un.setSize (150, 50);
        un.setLocation (100, 200);
        add (un);
        t1 = new TextField (10);
        t1.setSize (200, 50);
        t1.setLocation (300, 200);
        add (t1);
        un2 = new Label ("Enter no 2");
        un2.setSize (150, 100);
        un2.setLocation (100, 300);
        add (un2);
    }
}
```

```
t2 = new TextField(10);
t2.setSize(200,50);
t2.setLocation(300,300);
add(t2);

b1 = new Button("+");
add(b1);

b2 = new Button("-");
add(b2);

b3 = new Button("*");
add(b3);

b4 = new Button("/");
add(b4);

b5 = new Button("%");
add(b5);

b0 = new Button("=");
add(b0);

b = new Button(".");

b.setSize(100,100);
b.setLocation(200,100);
add(b);

b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
b5.addActionListener(this);
b6.addActionListener(this);
b0.addActionListener(this);
b.addActionListener(this);
```

```
public void actionPerformed(ActionEvent e)
{
    String s1 = t1.getText();
    String s2 = t2.getText();
    int x = Integer.parseInt(s1);
    int y = Integer.parseInt(s2);

    if (e.getSource() == b1)
    {
        b.setLabel(" " + (x+y));
    }

    if (e.getSource() == b2)
    {
        b.setLabel(" " + (x-y));
    }

    if (e.getSource() == b3)
    {
        b.setLabel(" " + (x*y));
    }

    if (e.getSource() == b4)
    {
        b.setLabel(" " + (x/y));
    }

    if (e.getSource() == b5)
    {
        b.setLabel(" " + (x%y));
    }
}
```

```
class calci  
{  
    P S V M (St ar[J])  
}  
  
FDemo f = new FDemo();  
f.setVisible (true);  
f.setSize (800,600);  
f.setLocation (100,200);  
f.setBackground (Color.gray);  
}
```

→ shapes :-

```
import java.awt.*;  
import java.awt.event.*;  
  
class FDemo extends Frame implements AL  
{  
    Button b1, b2;  
    boolean b = false, c = false, d = false;  
    Font f  
    setFont (f);  
    setLayout (new FlowLayout());  
    b1 = new Button ("Rect Shape");  
    add (b1);  
    b2 = new Button ("Circle Shape");  
    add (b2);  
}
```

```
b3 = new JButton("Arc shape");  
add(b3);
```

```
b1. addActionListener(this);
```

```
b2. addActionListener(this);
```

```
b3. addActionListener(this);
```

```
}
```

```
public void paint(Graphics g)
```

```
{
```

```
if(b)
```

```
{
```

```
g.fillRect(200, 200, 200, 200);
```

```
}
```

```
if(c)
```

```
{
```

```
g.fillOval(250, 100, 200, 200);
```

```
}
```

```
if(d)
```

```
{
```

```
g.fillArc(500, 100, 200, 200, 90, 270);
```

```
}
```

```
}
```

```
public void actionPerformed(ActionEvent e)
```

```
{
```

```
if(e.getSource() == b1)
```

```
{
```

```
c = false;
```

```
d = false;
```

```
b = true;
```

```
repaint();
```

```
setBackground(Color.blue);
```

```
setForeground(Color.pink);
```

```
}
```

```
if(e.getSource() == b2)
```

```
{
```

```
b = false;
```

```
d = false;
```

```
c = true;
```

```
repaint();
```

```
setBackground(Color.yellow);
```

```
setForeground(Color.red);
```

```
}
```

```
if(e.getSource() == b3)
```

```
{
```

```
b = false;
```

```
c = false;
```

```
d = true;
```

```
repaint();
```

```
setBackground(Color.green);
```

```
setForeground(Color.blue);
```

```
}
```

```
}
```

```
}
```

```
class Shape
```

```
{
```

```
psvm (st or ej)
```

```
{
```

```
FDemo f = new FDemo();  
f.setVisible(true);  
f.setSize(800, 600);  
f.setLocation(100, 60);  
f.setBackground(Color.gray);  
}
```

22/10/21

- What return type of e.getSource() is object.
- Object can never be converted to Button.
- Super class of Button is Object.

Calculator :—

```
import java.awt.*;  
import java.awt.event.*;  
class FDemo extends Frame implements AL  
{  
    FDemo() { int i, j, k = 0; int x = 100, y = 100, w = 100, h = 100;  
        Button b1[] = new Button[9];  
        Font f = new Font("Blackadder.ITALIC", Font.ITALIC, 50);  
        setFont(f);  
        setLayout(null);  
        b1 = new Button[" "];
```

```
for(i=0; i<3; i++)
```

```
{
```

```
for(j=0; j<3; j++)
```

```
{
```

```
b1[k] = new Button(" ");
```

```
b1[k].setSize(w, h);
```

```
b1[k].setLocation(x, y);
```

~~```
 b1.add(b1[k]);
```~~

```
b1[k].addAction.addActionListener(this);
```

```
x+=100; k++;
```

```
}
```

```
y+=100;
```

```
x=100;
```

```
}
```

```
int c=0;
```

```
public void actionPerformed(ActionEvent e)
```

```
{
```

```
Button b=e(Button) e.getSource();
```

```
if(c%2==0)
```

```
{
```

```
b.getsetLabel("O");
```

```
}
```

```
else
```

```
{
```

```
b.setLabel("X");
```

```
}
```

b. removeActionListener(this);

C++ ;

{  
}

Class Demo

{

    p s v m (st ar[ ])

{

    FDemo f = new FDemo();

{

→ with 2D array —

int i, j;

Button b1[ ][ ];

int x=100, y=100, w=100, h=100;

FDemo()

{

    Font f = new ( )  
    setFont(f);  
    setLayout(null);

```
b1 = new Button[3][3];
add
for (i=0; i<3; i++)
{
 for (j=0; j<3; j++)
 {
 b1[i][j] = new Button(" ");
 b1[i][j]. setSize (w, h);
 b1[i][j]. setLocation (x, y);
 add b1[i][j];
 b1[i][j]. addActionListener (AE e)
 x += 100;
 }
 y += 100;
}
```

→ default manager Layout border Layout चलता है

→ for making radio button we have to take help of checkboxGroup();

Example: —

```
import java.awt.*;
class FDemo extends Frame
{
 CheckboxGroup cbg;
 FDemo()
 {
 cbg = new CheckboxGroup();
 Font f = new Font(
 "setFont(f);");
 setLayout(new FlowLayout());
 Checkbox cb1 = new Checkbox("male", true, cbg);
 add(cb1);
 Checkbox cb2 = new Checkbox("female", false, cbg);
 add(cb2);
 }
 class demo
 {
```



→ getState() → यह checkbox check हो रहा है तो वे true return करते हैं otherwise false

Example : — Fetch the value from TextField.

```
import java.awt.*;
import java.awt.event.*;

class Fdemo extends Frame implements ActionListener
{
 TextField t1;
 Button b1;
 Checkbox cb1, cb2;

 Fdemo()
 {
 Font f = new Font("Times New Roman", 1, 16);
 setFont(f);
 setLayout(new FlowLayout());
 cb1 = new Checkbox("java");
 add(cb1);
 cb2 = new Checkbox("python");
 add(cb2);
 t1 = new TextField("20");
 add(t1);
 }

 public void actionPerformed(ActionEvent e)
 {
```

```
string st = " ";
```

```
if (cb1.getState() ==
```

```
{
```

```
s1 = cb1.getLabel();
```

```
}
```

```
if (cb2.getState() ==
```

```
{
```

```
s1 = s1 + cb2.getLabel();
```

```
}
```

```
t1.setText(s1);
```

```
}
```

```
}
```

```
class demo
```

```
{
```

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame implements AL
{
 CheckboxGroup cbg;
 TextField tb, tn, te, tp, tph, tz, ta, tl, td;
 Button b1;

 FDemo()
 {
 Font f = new Font (
 setFont(f);
 setLayout(null);

 setTitle ("REGISTRATION FORM");
 setBackground (Color.gray);
 setBackForeground (Color.black);

 Label un = new Label ("Name");
 un. setSize(100, 30);
 un. setLocation(100, 90);
 add(un);

 Label ud = new Label ("DOB");
 ud. setSize(100, 30);
 ud. setLocation(100, 130);
 add add(ud);
 }
}
```

```
Label ue = new Label ("Email ");
ue. setSize (100, 30);
ue. setLocation (100, 170);
add (ue);
```

```
Label up = new Label ("Password ");
up. setSize (100, 30);
up. setLocation (100, 210);
add (up);
```

```
Label up1 = new Label ("Phone Number ");
up1. setSize (100, 30);
up1. setLocation (100, 250);
add (up1);
```

```
Label g = new Label ("Gender ");
g. setSize (100, 30);
g. setLocation (100, 290);
add (g);
```

```
Label L = new Label ("Language ");
L. setSize (100, 30);
L. setLocation (100, 330);
add (L);
```

```
Label z = new Label ("Pin code ");
z. setSize (100, 30);
z. setLocation (100, 370);
add (z);
```

```
Label o = new Label ("Address");
o. setSize (100, 30);
o. setLocation (100, 470);
add(o);
```

// TextField

```
tn = new TextField ();
tn. setSize (250, 30);
tn. setLocation (250, 90);
add(tn);
```

```
td = new TextField ();
td. setSize (250, 30);
td. setLocation (250, 190);
add(td);
```

```
te = new TextField ();
te. setSize (250, 30);
te. setLocation (250, 170);
add(te);
```

```
tp = new TextField ();
tp. setSize (250, 30);
tp. setLocation (250, 210);
add(tp);
```

```
tph = new TextField ();
tph. setSize (250, 30);
tph. setLocation (250, 250);
add(tph);
```

```
tz = new TextField ();
tz. setSize (250, 30);
tz. setLocation (250, 290);
430
add(tz);
```

```
to = new TextField ();
to. setSize (250, 30);
to. setLocation (250, 340);
470
```

```
cbg = new CheckboxGroup ();
Checkbox cb1 = new Checkbox ("Male", true, cbg);
cb1. setSize (250, 30);
cb1. setLocation (250, 290);
add(cb1);
```

```
Checkbox cb2 = new Checkbox ("Female", false, cbg);
cb2. setSize (250, 30);
cb2. setLocation (250, 320);
add(cb2);
```

```
Checkbox cb3 = new Checkbox ("Other", false, cbg);
cb3. setSize (250, 30);
cb3. setLocation (250, 350);
add(cb3);
```

Choice

c1. add

c1. add

c1. add

c1. add

c1. add

c1. setS

c1. setLo

add(c1);

b1 = new

b1. sets

b1. setLo

add(b1);

b1. add

{

}

class Regis

{

}

```
Choice c1 = new Choice();
```

```
c1.add("Hindi");
```

```
c1.add("Java");
```

```
c1.add("English");
```

```
c1.add("C++");
```

```
c1.add("DS");
```

```
c1.setSize(100, 30);
```

```
c1.setLocation(250, 395);
```

```
add(c1);
```

```
b1 = new Button("Registration");
```

```
b1.setSize(250, 40);
```

```
b1.setLocation(400, 600);
```

```
add(b1);
```

```
b1.addActionListener(this);
```

```
{
```

```
}
```

```
class Regist
```

```
{
```

```
}
```

28/12/21

# Fetch the value from radio button

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame implements AL
{
 TextField t1;
 Button b1;
 Checkbox cb1, cb2;
 CheckboxGroup cbg;

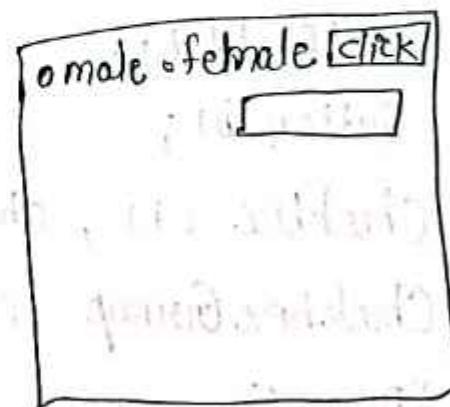
 FDemo()
 {
 Font f = new Font (
 "Times New Roman", 16, 1);
 setFont(f);
 setLayout(new FlowLayout());
 cb1 = new Checkbox("Male", false, cbg);
 add(cb1);
 cb2 = new Checkbox("Female", true, cbg);
 add(cb2);
 b1 = new Button("Click !!");
 add(b1);
 b1.addActionListener(this);
 t1 = new TextField(10);
 add(t1);
 }

 public void actionPerformed(ActionEvent e)
 {
 if(e.getSource() == b1)
 {
 if(cb1.getState())
 t1.setText("Male");
 else
 t1.setText("Female");
 }
 }
}
```

```

public void actionPerformed(ActionEvent e)
{
 String s = " ";
 if (cb1.getState() == true)
 {
 s1 = cb1.getLabel();
 }
 if (cb2.getState())
 {
 s1 = s1 + cb2.getLabel();
 }
 t1.setText(s1);
}

```



# string s1 = cbg.getSelectedCheckbox().getLabel();

→ उस particular group में से जो भी checkbox select हुआ है, वो लिएँ

provide कर देगा।

→ Example:

```

public void actionPerformed(ActionEvent e)
{
 String s1 = cbg.getSelectedCheckbox().getLabel();
 t1.setText(s1);
}

```

# Group of more than one radio button :—

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame implements AL
{
 JTextField t1;
 Button b1;
 Checkbox cb1, cb2, cb3, cb4, cb5;
 CheckboxGroup cbg, cbg2;

 FDemo()
 {
 Font f
 setFont(f);
 setLayout(new FlowLayout());
 cb1 = new Checkbox("male", false, cbg);
 add(cb1);
 cb2 = new Checkbox("female", true, cbg);
 add(cb2);
 cb3 = new Checkbox("sc", false, cbg2);
 add(cb3);
 cb4 = new Checkbox("st", false, cbg2);
 add(cb4);
 cb5 = new Checkbox("ODC", true, cbg2);
 add(cb5);
```



## fetching the data from choice --

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame implements ActionListener
{
 TextField t1;
 Button b1;
 Choice cb1;

 FDemo()
 {
 Font f = new Font("Times New Roman", 1, 16);
 setFont(f);
 setLayout(new FlowLayout());
 cb1 = new Choice();
 cb1.add("java");
 cb1.add("python");
 cb1.add("css");
 cb1.add("php");
 add(cb1);
 b1 = new Button("Click !!");
 add(b1);
 b1.addActionListener(this);
 t1 = new TextField(10);
 add(t1);
 }

 public void actionPerformed(ActionEvent e)
 {
 if(e.getSource() == b1)
 {
 String s = cb1.getSelectedItem();
 t1.setText(s);
 }
 }
}
```

```
 }
 public void actionPerformed(ActionEvent e)
 {
 String s1 = cb1.getSelectedItem();
 t1.setText(s1);
 }
}
```

→ for managing the choices in List. we have write  
getSelectedItems();

→ more than one item select करने की facility  
देना A List की main purpose होता है, and for  
this there is a method called object. getSel-  
ectedItems();

Example :—

```
import java.awt.*;
import java.awt.event.*;
```

```
class FDemo extends Frame
{
 List cb1;
 Button b1;
 TextField t1;
 FDemo
 {
```

```
Font f = new
setFont(f);

setLayout(new FlowLayout());
cb1 = new List(3, true);
cb1.add("java");
cb1.add("python");
cb1.add("css");
cb1.add("html");
add(cb1);

§ b1 = new Button("Click!!");
§ add(b1);
b1.addActionListener(this);

t1 = new TextField(10);
add(t1);

§
public void actionPerformed(ActionEvent e)

{
String s1[] = cb1.getSelectedItems();
String s = " ";
for (String s2 : s1)
{
s = s + ". " + s2;
}
t1.setText(s);
}
```

→ fetch the data from choice but without Button.

for that we have to add the Listener in choice. for adding the Listener in choice. Java gave us ItemListener. ItemListener has single method. That is itemStateChanged() method.

Solution :-

```
import java.awt.*;
import java.awt.event.*;

class FDemo extends Frame implements ItemListener
{
 Choice cb;
 TextField t1;

 FDemo()
 {
 Font f = new Font("Times New Roman", 1, 16);
 setFont(f);
 setLayout(new FlowLayout());
 cb = new Choice();
 cb.add("java");
 cb.add("python");
 cb.add("css");
 cb.add("php");
 cb.addItemListener(this);
 t1 = new TextField(20);
 add(cb);
 add(t1);
 }

 public void itemStateChanged(ItemEvent e)
 {
 if(e.getStateChange() == ItemEvent.SELECTED)
 {
 String s = cb.getSelectedItem();
 t1.setText(s);
 }
 }
}
```

```
add(cb1);
cb1.addItemListener(this);
t1 = new Textfield(10);
add(t1);
}
public void itemStateChanged(ItemEvent e)
{
String s1 = cb1.getSelectedItem();
t1.setText(s1);
}
```

- we can use ItemListener in List, Checkbox, RadioButton and Choice.
- advance version of awt is Swing.

## Oops

24/12/21

- In Java, the data of class inside class by default is private.
- class के अंदर का data ही का class के object को को access की जाना है।

In C++:

```
class A
{
public:
int x = 10;
};

void main()
{
A a;
cout << a.x;
}
```

error: can not initialize  
instance of class  
here.

In Java:

```
class A
{
int x = 10;
};

class Demo
{
public static void main (String args[])
{
A a = new A();
System.out.println(a.x);
}
}

O/P → 10
```

- यहाँ object के variable class के बाहर access नहीं होता जाना। इसीलिए उसे instance variable कहते हैं।
- object का पुस्ता नाम instance है।
- class के अंदर का variable instance पर depend है। इसीलिए वह instance variable कहते हैं।

Example :

```

class A
{
 int x;
}

class Demo
{
 public void main (String args[])
 {
 A a = new A();
 System.out.println(a.x);
 }
}

```

~~Output : O/P~~ → 0

• Local variable को use करने से पहले उसमें value assign करना compulsory है।

• instance variable को use करने से पहले उसमें value pass करना compulsory नहीं है, अगर हम instance variable में कोई value assign नहीं करते हैं, तो O/p default values के according आएगा।

Default values of instance variable.

byte , short, int, long ⇒ 0  
 float, double ⇒ 0.0  
 boolean ⇒ false  
 char ⇒ space  
 String ⇒ null

# Example of instance method

class A

```

{
 void show() // instance method
 {
 System.out.println ("Class A");
 }
}

```

class demo

```

{
 public void main (String args[])
 {
 A a = new A();
 a.show();
 }
}

```

A a = new A();

a.show();

```

{
}

```

O/P ⇒ Class A

Example :-

```
class A
{
 int x, y;
 void show() get(int a, int b)
 {
 x = a;
 y = b;
 }
 void show()
 {
 System.out.println("x=" + x);
 System.out.println("y=" + y);
 }
}

class demo
{
 public static void main()
 {
 A a = new A();
 a.show();
 a.get(10, 20);
 a.show();
 }
}

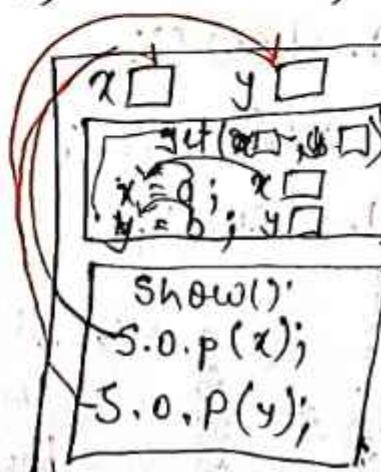
O/P => 0
 0
 10
 20
```

Example :-

```
class A
{
 int x, y;
 void get(int x, int y)
 {
 x = x; // 10
 y = y; // 20
 }
 void show()
 {
 System.out.println("x=" + x);
 System.out.println("y=" + y);
 }
}

class demo
{
 public static void main()
 {
 A a = new A();
 a.get(10, 20);
 a.show();
 }
}

O/P => 0
 0
 10
 20
```



25/12/21

## Example —

class A

{

int x, y;

void get (int a, int b)

    { local variable

int x, y;

x = a;

y = b;

}

void show()

{

System.out.println ("x = " + x);

System.out.println ("y = " + y);

{

class Demo

{

public static void main (String args)

{

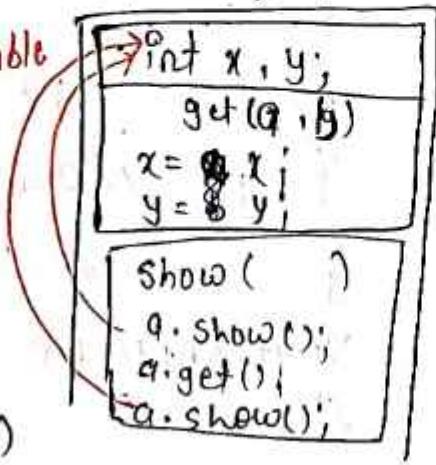
A a = new A();

a.show();

a.get(10, 20);

a.show();

{

Explanation : x and y  
might not have  
been initialized.

→ more than 1 words की  
merge करने के लिए

underscore (-) का  
use करने के लिए

→ hyphen → minus symbol

→ if return type is int  
and you are not  
returning any value  
then it will display on  
error.

→ return type की value  
return करनी होती है।

उसी type की ~~मेहराब~~  
return type हीना चाहिए

→ It is compulsory to  
mention return type.

→ If we want to return  
int value, so return  
type is int.

If we want to  
return array so return  
type be in array form  
i.e., int [] show() {  
    return a; }

for closing the window in awt :-

method → ~~public void~~ f. addWindowListener (new  
WindowAdapter () { public void windowClosing  
(WindowEvent e) { System.exit (0); } } );

→ java के all objects के default value null होती है।

Example :-

```
class A
{
 int x, y;
 void get (int a, int b)
 {
 int x, y;
 x = a;
 y = b;
 }
 void show()
 {
 System.out.println(x);
 System.out.println(y);
 }
}
```

```
psvm(st arq)
A a = new A();
a.show();
a.get(10, 20);
a.show();
O/p => 0
 0
 0
 0
```

class Demo

Example :-

```
class A
{
 int x, y;
 void show()
 {
 System.out.println(x);
 System.out.println(y);
 int x = 1, y = 2;
 System.out.println(x);
 System.out.println(y);
 }
}
```

class Demo

```
{}
public class Demo
{
 public static void main(String args[])
 {
 A a = new A();
 a.show();
 a.show();
 }
}
```

Example :

```
class A
{
 int x, y;
 void show()
 {
 System.out.println(x);
 System.out.println(y);
 x = 1; y = 2;
 System.out.println(x);
 System.out.println(y);
 }
}
```

class Demo

```
{}
public class Demo
{
 public static void main(String args[])
 {
 A a = new A();
 }
}
```

A a = new A();

a.show();

a.show();

```
1
2
1
2
```

Example :

class A

{

string s1;

A s2;

void show()

{

S.o.println(s1);

S.o.println(s2);

}

}

O/p ⇒ null class demo

null

P: S, V M (START)

{

A a = new A();

a.show();

}

}

O/p ⇒ null

null

Example :

class A

{

void cube(int x)

{

return x\*x\*x;

}

}

error : incompatible type:

unexpected value.

return value

Example :

class A

{

int cube(int x)

{

error : missing return statement.

Example :

class A

{

int cube(int x)

{

return 10.8; }

error: incompatible type:  
possible lossy conversion from double to  
int.  
return 10.8

Example:

```
class A
{
 void show(int a[])
{
 for (int i : a)
 System.out.println(i);
}
```

Example:

```
class A
{
 int cube (int x)
{
 return x*x*x;
}

class Demo
{
 public static void main (String args[])
{
 A a = new A();
 System.out.println(a.cube(10));
}
}
```

O/p  $\Rightarrow$  1000

class demo

```
class demo
{
 public static void main (String args[])
{
 int x[] = {10, 20, 30, 40, 50};
 A a = new A();
 a.show(x);
}

O/p \Rightarrow
10
20
30
40
50
```

Example :

```
class A
{
void show(int a[])
{
 a[2] = 555;
}
}

class Demo
{
public void main (String args[])
{
 A a = new A();
 a.show(x);
 for (int i : x)
 {
 System.out.println(i);
 }
}
```

O/p → 10  
20  
5 555  
40  
50

Example :

```
class A
{
void show()
{
}
}

class Demo
{
public void main (String args[])
{
 A a = new A();
 a.show();
}
```

Ans :-

invalid method declaration  
; return type required  
→ addition of & ID array  
in 3rd ID array.

class A

```
{ int [] show(int a[],int b[])
{
 int c[] = new int [5];
 int i;
```

```

for(i=0; i<5; i++)
{
 c[i] = a[i] + b[i];
}
return c;
}

class Demo
{
int x[] = {10, 20, 30, 40, 50};
int y[] = {1, 2, 3, 4, 5};
A a = new A();
int z[] = a.show(x, y);
for(int i: z)
{
 S.o.pln(i);
}
}

O/p => 11
 22
 33
 44
 55

```

```

for(int i: a)
{
 S.o.pln(a[i]);
}

class Demo
{
A a = new A();
a.show(new int[]{10, 20, 30, 40, 50, 60});
}

O/p => 10
 20
 30
 40
 50
 60

```

### → Example

```

class A
{
void show(int a[])
{
}

```

→ class की use करने वाले object का name compulsory हो।

→ String की खाने के लिए हम character array का use कर सकते हैं।

Example : types of making String —

class Demo

{

    public void main (String args[])

}

String s1 = new ("Softwaves-1");

String s2 = new String ();

s2 = "Softwaves-2";

String s3 = "Softwaves-3";

char x[] = { 's', 'o', 'f', 'f', 'w', 'a', 'v', 'e', 's' }

String s4 = new String (x);

String s5 = new String (x, 0, 3);

String s6 = new String (x, 3, 4);

String s7 = new String (x, 3, 14);

s.0.println (s1);

s.0.println (s2);

s.0.println (s3);

constructor

```
S.o.println(s4);
S.o.println(s5);
S.o.println(s6);
S.o.println(s7);
{
{
```

O/P → Softwaves-1

Softwaves-2

Softwaves-3

Softwaves

Sof

twav

(in s7) → exception : string index out of bound exception

# Methods of String class

- Initialization of instance variable :-

- type ! : → assign the value in instance variable

class A

{

int x, y;

void show()

{

S.o.println("x = " + x);

S.o.println("y = " + y);

}

```
3
class Demo
{
 public static void main (String [] args)
 {
 A a = new A ();
 a.x = 10;
 a.y = 20;
 a.show ();
 }
}

O/P => x = 10
 y = 20
```

- Type 2 → of assigning / initialize value in instance variable.

```
class A
{
 void get (int a, int b) {
 x = a;
 y = b;
 }
 void show () {
 System.out.println (x);
 System.out.println (y);
 }
}
```

3  
2

class Demo

{

    public static void main()

{

        A a = new A();

        a.get(1, 2);

        a.show();

{  
}

**explicitly calling  
of get method**

O/P → ?

## # Constructor :-

- i) Constructor is a special type of member function. bcz it has same name as class itself.
- ii) Constructor has no return type.
- iii) Constructor is automatically called when an object is created.
- iv) Constructor is used to initialize the instance variable.
- v) If we do not make any constructor in class so by default compiler will make a default constructor.

Types :- Java has only two type of constructor.

- i) default constructor
- ii) Parameterized Constructor
- iii) No parameterized Constructor (without any parameter)

Example of parameterized Constructor —

```
class A
{ int x, y;
A(int a, int b)
{
x = a;
y = b;
}
void show()
{
System.out.println(x);
System.out.println(y);
}
}

class demo
{ public static void main(String args[])
{ A a = new A(10, 15);
a.show();
}
}
```

Example of no parameterized/  
no argument constructor -

class A

{

A()

{

S.o.p("Indore"),

}

class Demo

{

A a = new A();

A a1 = new A();

A a2 = new A();

}

}

O/p → Indore

Indore

Indore

Example :-

void show()

{

S.o.p("Show method");

}

class Demo

{

A a = new A();

a.show();

A a2 = new A(20, 50);

}

error: Constructor A in class

A cannot be applied  
to given types:

A a = new A();

→ When we make any  
one of constructor in  
class so compiler will  
not make any constructor.

class A

{

A(int x, int y)

{

S.o.p("param cons");

Example of Constructor  
Overloading -

class A

{

A(int x, int y)

```

S.o.println("param cons");
}
A()
{
S.o.println("no arg cons");
}
void show()
{
S.o.println("show method");
}

```

एक class में constructor को diff. diff. parameters के साथ उन्हें ready करने constructor overloading कहते हैं।

```

class Demo {
 void show() {
 System.out.println("show method");
 }
 void show(int a, int b) {
 System.out.println(a + b);
 }
}

```

O/P ⇒ no arg cons  
show method  
param cons.

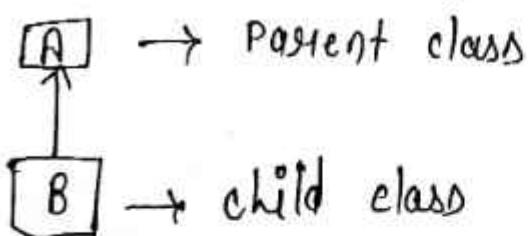
→ zero parameter constructor  
is called default/no argument constructor.

Inheritance :- When one object acquires all the properties and behaviors of its parent object automatically.

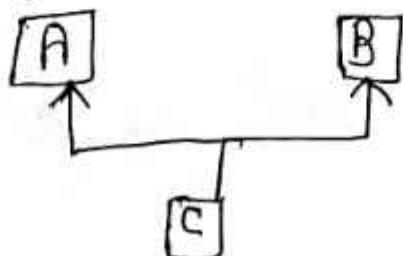
There are 5 types of inheritance :-

- 1) Single Inheritance
- 2) Multiple Inheritance
- 3) Multilevel Inheritance
- 4) Hierarchical Inheritance
- 5) Hybrid Inheritance

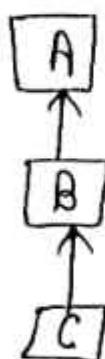
1) Single Inheritance :- Single Inheritance is defined as a inheritance in which a child class is inherited by only one base class.



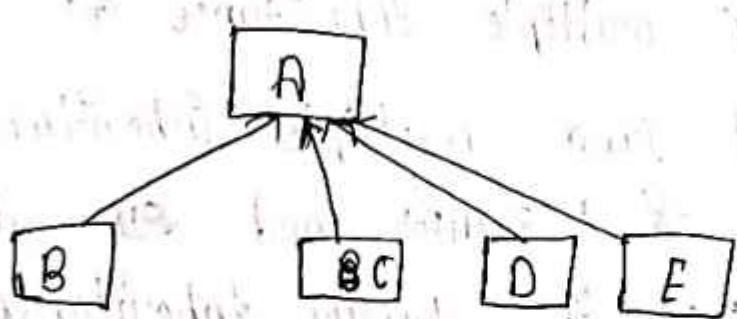
2) Multiple Inheritance :- Multiple Inheritance is defined as a inheritance in which one child class is inherited by more than parent class. but Java does not support <sup>one</sup> Multiple Inheritance



3) Multilevel Inheritance :— when one class inherit another class which is further inherited by another class is called as Multilevel Inheritance.

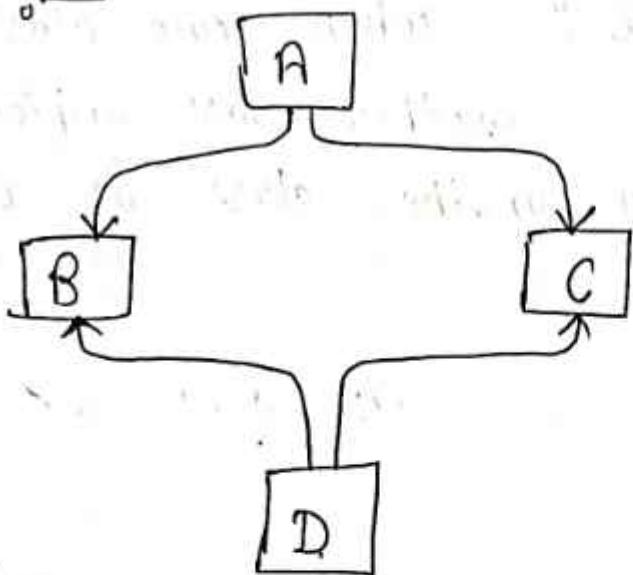


4) Hierarchical Inheritance :— # Hierarchical inheritance is defined as the inheritance in which more than one child classes are inherited by only one parent class. It is opposite of Multiple Inheritance.



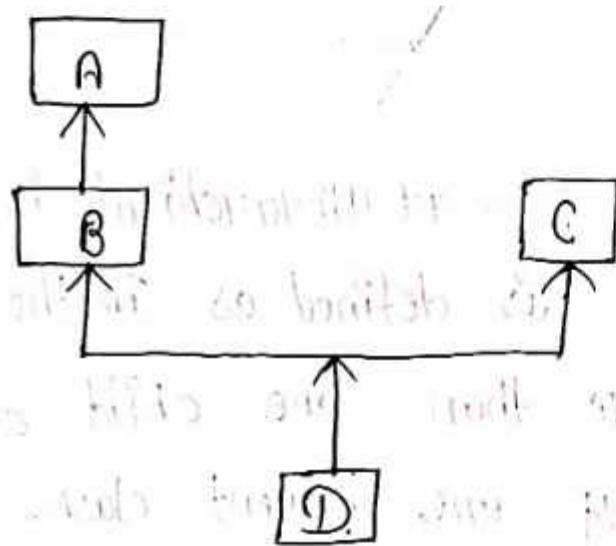
5) Hybrid Inheritance :— Hybrid Inheritance is a combination of more than one type of Inheritance. It is also known as virtual Inheritance.

Ex - 1 :-



$$\begin{array}{rcl} 1 + 4 & = 5 \\ 2 + 4 & = 5 \\ 3 + 4 & = 5 \end{array}$$

Ex - 2 :-



→ Hybrid Inheritance में multiple Inheritance को हीना compulsory है। but Java multiple inheritance को support नहीं करती है। ~~support~~ and ~~without~~ multiple inheritance (4) के hybrid inheritance नहीं होता। that's why Java hybrid inheritance को support नहीं करती है।

## Example :

class A

{  
int x, y;

void get (int a, int b)

{  
x = a;  
y = b;

{  
void sum()

{  
System.out.println ("Sum = " + (x+y));

{  
}

class B

{  
int x, y;

void get (int a, int b)

{  
x = a;  
y = b;

{  
void sub()

{  
System.out.println ("Sub = " + (x-y));

{  
}

class Demo

{

public static void main (String args)

{

A a = new A();

a.get (10, 20);

a.sum ();

B b = new B();

b.get (100, 20);

b.sub ();

{

}

O/P  $\Rightarrow$  Sum = 30

Sub = 80

Example of Single Inheritance  
→ Parent class

class A

{

int x, y;

void get (int a, int b)

{

x = a;

y = b;

{

void sum ()

{

System.out.println ("Sum = " + (x+y));

{

```

class B extends A
{
 void sub()
 {
 System.out.println("Sub = " + (x - y));
 }
}

class Demo
{
 A a = new A();
 a.get(10, 20);
 a.sum();
 B b = new B();
 b.get(100, 20);
 b.sub();
}

O/P => Sum = 80
 Sub = 80

```

Example of Multi-level inheritance

```

class A
{
 void show1()
}

```

```

System.out.println("Class A");
class B extends A
{
 void show2()
 {
 System.out.println("Class B");
 }
}

class C extends B
{
 void show3()
 {
 System.out.println("Class C");
 }
}

class Demo
{
 C c = new C();
 c.show1();
 c.show2();
 c.show3();
}

O/P => Class A
 Class B
 Class C

```

Example of Hierarchical Inheritance :-

```
class A
{
 void show1()
 {
 System.out.println("class A");
 }
}

class B extends A
{
 void show2()
 {
 System.out.println("class B");
 }
}

class C extends A
{
 void show3()
 {
 System.out.println("class C");
 }
}

class demo
{
 B b = new B();
 b.show1();
 b.show2();
}
```

C c = new C();  
c.show1();  
c.show3();  
3  
3

O/p ⇒ class A  
class B  
class A  
class C

Q → why java does not have multiple inheritance?

bcz java does not have multiple inheritance:

because sometimes, multiple inheritance occurs ambiguity / ambiguous error

So, this error msg should not occur in the case of inheritance in java. that's why java does not support multiple inheritance. bcz of ambiguity error.

In C :-

Ex - of Multiple Inheritance.

class A

{

public :

void show()

{

cout << "class A\n";

}

;

class B

{

public :

void show()

{

cout << "class B\n";

}

;

class C : public B, public A

{

public :

;

class Demo : void main()

{

C c = new C();

c. show();

}

ORIG: function is ambi.

In Java :-

class A

{

void show()

{

System.out.println("Class A");

}

;

class B

{

void show()

{

System.out.println("Class B");

}

;

class C extends B, extends A

{

void show()

{

System.out.println("Class C");

}

;

class Demo {

C c = new C();

}

ORIG: '}' expected

class C extends B,

extends A

Example :—

class A

{

A()

{

S.o.println("class A constructor");

}

}

class B extends A

{

B()

{

S.o.println("class B const.");

}

}

class C extends B

{

C()

{

S.o.println("class C const.");

}

}

\* class demo

{

C c = new C();

}

O/P ⇒ class A constructor

class B const.

class C const.

→ Super class का const.  
structure पहले call होता  
है।

Example :

class A

{

A(int a, int b)

{

S.o.println("sum = " + (a+b));

}

}

class B extends A

{

B(int a, int b)

{

S.o.println("sum = " + (a+b));

}

class Demo

{

B b = new B(10, 20);

}

const. A in class A

ज्ञानः— cannot be applied to  
given types;

Super class का पास  
default constructor नहीं  
मिलने पर error msg display

class A

{

}

class B extends A

{

B(int a, int b)

{

System.out.println("sum = " + (a+b));

{

class Demo

{

B b = new B(10, 20);

{

O/P → sum = 30

Example :

class A

{

A()

{

System.out.println("class A Default  
const.");

{

class B extends A

{

B(int a, int b)

{

System.out.println("sum = " + (a+b));

{

class Demo

{

B b = new B(10, 20);

{

→ by default super class

का default constructor ही call होगा और

मात्र super class के

पास default constructor

हो नहीं है, तो क्षमता

condition में error msg

भाकर display हो जाएगी।

29/12/21

- Construction chaining → Sub class या object बनाने के लिए पहले Super class का constructor automatic call हो जाता है, इसी concept की construction chaining कहते हैं।
- Compiler ने भी पहले check करता है कि class में constructor हो या नहीं। whenever super() की जगह वो ही Super class का constructor call होता है। whenever you create a constructor inside sub-class. Compiler will check whether constructor have super or not (inside sub class) if super is available then compiler will not create any super. Or if super() is not found then by default compiler generate super with zero parameters.

Example :

class A

{

A()

{

}

S.o.println("class A Default con.");

}

{

class B extends A

{

B()

{

Super();

Default  
S.o.println(" Class B Super  
const");

{

{

class demo

{

{

B b = new B();

{

O/P, class A Default con  
class B Default const.

Example:

```
class A
{
 A()
}
S.o.println("class A Default Cons.");
```

```
class B extends A
{
 B()
}
S.o.println("C" Super());
 Super();
```

```
S.o.println("class B Default Cons.");
```

```
class demo
```

```
B b = new B();
```

error: call to super must  
be first statement

in constructor:

```
Super();
```

Note → Super always  
constructor में

पहला statement होना  
चाहिए।

Example:

```
class A
```

```
A()
{
```

```
S.o.println("class A D.C.");
```

```
}
class B extends A
```

```
B()
{
```

```
S.o.println("Class B D.C.");
```

```
Super();
```

```
}
class demo
```

```
B b = new B();
```

error: call to super must  
be first statement  
in constructor.

Example: Constructor chaining This keyword → this current object को refer.

class A  
{

A(int a, int b)

}

S.o.println("Sum = " + (a+b));

}

}

class B extends A

{

B(int a, int b)

{ super(a, b);

S.o.println("Sub = " + (a-b));

}

}

class demo

{

B b = B(10, 20);

}

O/p ⇒ 30  
-10

ence को hold करता है।

and instance variable को point करता है।

Example of this keyword—

class A

{

int x = 10;

~~int~~ y = void show()

{

int x = 20;

S.o.println(x);

S.o.println(this.x);

{

class demo

⇒ {

A a = new A();

a.show();

{

O/p ⇒ 20 (10, 20) top, b

10 (20, 20) middle, b

→ Local variable and instance variable की memory

अलग - अलग होती है, दोनों

अलग - अलग memory में

धाकर store होते हैं।

Example :

class A

{

int x;

int y;

void get(int x, int y)

{

this.x = x;

y = y;

}

void show()

{

System.out.println("x=" + x);

System.out.println("y=" + y);

}

}

class demo

{

A a = new A();

a.show();

a.get(10, 20);

a.show();

}

O/P  $\Rightarrow$  x = 0

y = 0

x = 10

y = 0

## Constructor

- Constructor has same name as its class.

- Constructor has no return type.

- Constructor is automatically called. when an object of class is created.

- एक object से constructor  
only एक बार call  
हो सकता है।

## Method

Method has name and class name can ~~not~~ be same or different both.

It is compulsory to specify the return type of Method.

we have to call it explicitly.

एक object से तभी Method  
जो more than one time  
call कर सकते हैं।

31/12/21

Example :

```
class A
{
void A()
{
S.o.println("ram");
}
```

class demo

```
{ A a = new A();
a.A(); }
```

O/P ⇒ ram

# अगर हम method का

return type specify

नहीं करते हैं, तो ऐसा

compiler by default करते

constructor जान सकता है।

Example :

error : invalid method declaration : return type required.  
show() ^

Example :

class A

{

A()

{

S.o.println("Constructor ");

}

void show()

{

S.o.println("method ");

}

```
class Demo
```

```
A a = new A();
```

```
a.A();
```

```
{
```

```
A a = new A();
```

~~a.A();~~ A a1 = new A(); error: can not find symbol

```
a.A()
```

```
,
```

```
a.show();
```

```
a.show();
```

```
a.show();
```

```
{
```

%> constructor

Constructor

Method

Method

Method

Example:

```
class A
```

```
{
```

```
A()
```

```
{
```

```
S.o.pn ("Constructor ");
```

```
{
```

```
void A()
```

```
{
```

```
S.o.
```

Example:

```
class A
```

```
{
```

```
A()
```

```
{
```

```
S.o.pn (" Constructor ");
```

```
{
```

```
void show()
```

```
{
```

```
S.o.pn (" Method ");
```

```
{
```

```
{
```

```
class demo
```

```
{
```

## # Methods of String class :

return type

- ① int
- ② char
- ③ String
- ④ String
- ⑤ String
- ⑥ int
- ⑦ int
- ⑧ boolean
- ⑨ boolean
- ⑩ String
- ⑪ String
- ⑫ boolean
- ⑬ boolean
- ⑭ int
- ⑮ int
- ⑯ String

Method Name

- length();
- charAt(int);
- concat(String);
- toUpperCase();
- toLowerCase();
- CompareTo(String);
- compareToIgnoreCase(String);
- equals(String);
- equalsIgnoreCase(String);
- substring(int);
- substring(int, int);
- startsWith(String);
- endsWith(String);
- indexOf(String);
- lastIndexOf(String);
- trim();

⑭ String ~~replace~~(ch,

replace(char, char);

⑮ String []

split(String);

Example of length and concat method;

class demo

{

String s1 = "soft";

String s2 = "waves";

S.o.println(s1.length());

S.o.println(s2.length());

S.o.println(s1);

S.o.println(s2);

S.o.println(s1.concat(s2));

}

O/P → 4  
5

soft

waves

.softwaves

Example of charAt();  
method:

class demo

{

String s1 = "soft";

S.o.println(s1.charAt(2));

S.o.println(s1.charAt(20));

}

O/P ⇒ f

Exception :

toUppercase() → शब्द

String  
capital letter से convert  
हो जाएगी।

toLowerCase() → शब्द  
String  
small letter से convert  
हो जाएगी।

→ charAt() → particular character fetch करने

के लिए use करो है।

```
class demo
```

```
{
String s1 = "Soft Waves";
S.o.pn(s1);
```

```
S.o.pn(s1.toUpperCase());
S.o.pn(s1.toLowerCase());
```

```
}
O/p ⇒ Soft Waves
SOFTWAVES
softwaves
```

equals() → अगर दोनों  
string का

content same है, तो true  
return करेगा। Otherwise  
false return करेगा।

equalsIgnoreCase() → ये capital,  
small letter  
को ignore करता है।

```
class demo
```

```
{
String s1 = "ram";
```

```
String s2 = "ram";
```

```
String s3 = "Ram";
```

```
S.o.pn(s1.equals(s2));
```

```
S.o.pn(s1.equals(s3));
```

```
S.o.pn(s1.equalsIgnoreCase(
s3));
```

```
}
```

O/p ⇒ true  
false  
true

compareTo() → string  
की comp-

are करने के लिए use  
होते हैं। O के बारे में  
छुल ओपरेटर difference  
आएगा। तो वो आगे  
check करनी होती है।

class demo

{

String s1 = "abc";

String s2 = "abc";

String s3 = "Abc";

S.o.println(s1.compareTo(s2));

S.o.println(s1.compareTo(s3));

}

O/p ⇒ 0

Ques.

Example :

class demo

{

String s1 = "abc";

String s2 = "adzaa";

String s3 = "abcdefgh";

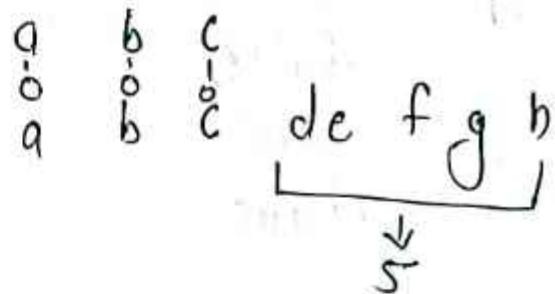
S.o.println(s1.compareTo(s2));

S.o.println(s1.compareTo(s3));

}

O/p ⇒ -2

-5



class demo

{

String s1 = "abc";

String s2 = "adc";

S.o.println(s1.compareTo(s2));

}

O/p ⇒ -2

startsWith()

→ ~~बहु~~ एकल  
return type

boolean है, but क्षमता

String pass करना

compulsory है। अगर

Same name से starting

होगी तो true return

होगा। otherwise false.

Example :-

class demo  
{

String s1 = "my name is  
ram";

s.o.println(s1.startsWith("m"));  
s.o.println(s1.startsWith("M"));  
s.o.println(s1.startsWith("my-"));  
s.o.println(s1.endsWith("m"));

}

O/P  $\Rightarrow$  true

false

true

true

class demo  
{

String s1 = "my name is  
ram is ram";

s.o.println(s1.indexOf("is"));

s.o.println(s1.indexOf("iss"));

s.o.println(s1.lastIndexOf("is"))

}

O/P  $\Rightarrow$  8

-1

15

- substring()  $\rightarrow$  index  $\rightarrow$  content

display दिया देगा {

- indexOf()  $\rightarrow$  दिये देगा {

मिला तो उस

content को index दिया देगा • length में से 1 से start  
दिया देगा अ-otherwise -1 दिया देगा |

Example :-

class demo  
{

String s1 = "my name is  
ram is ram";

दिया देगा |

S.o.println(s1);

S.o.println(s1.substring(4));

S.o.println(s1.substring(4, 9));

S.o.println(s1.substring(2, 10));

S.o.println(s1.substring(2, 100));

3

O/P ⇒ my name is ham is ham

ame is ham is ham

ame i

name is

Exception : SIOOBE

String में आगे को  
and पीछे को space

को ignore कर देगा।

Example :

class demo

3

String s1 = " my name is  
ham ";

S.o.println(s1);

S.o.println(s1.trim());

3

O/P ⇒ my name is ham

my name is ham

capital,

small को

Same मान लेता है।

Example :

class demo

3

String s1 = " abc ";

String s2 = " AbC ";

S.o.println(s1.compareTo(s2));

S.o.println(s1.compareToIgnoreCase(s2));

3

O/P ⇒ 32

0

Example of trim method —

class demo

3

String s1 = " 10 ";

String s2 = " 20 ";

int x = Integer.parseInt(s1.  
trim());

int y = Integer.parseInt(s2.);

```
s.o.println(x+y);
```

3

O/p  $\Rightarrow$  30

$\rightarrow$  content को

replace करने

कुछ फलाई

Example :

```
class demo
```

3

```
String s1 = "my name is ram";
```

```
s.o.println(s1);
```

```
s.o.println(s1.replace('a','z'));
```

3

O/p  $\Rightarrow$  my name is ram

my nzme is rzm

$\rightarrow$  content को

split कराने

कुछ फलाई

Example :

```
class demo
```

3

```
String s1 = "my name is
ram";
```

```
String s[] = s1.split(" ");
```

```
s.o.println(s[0]);
```

```
s.o.println(s[1]);
```

3

O/p  $\Rightarrow$  my  
is

Example :

```
class demo
```

3

```
String s1 = "my name is
ram";
```

```
String s[] = s1.split(" ");
```

~~for (int i=0; i~~

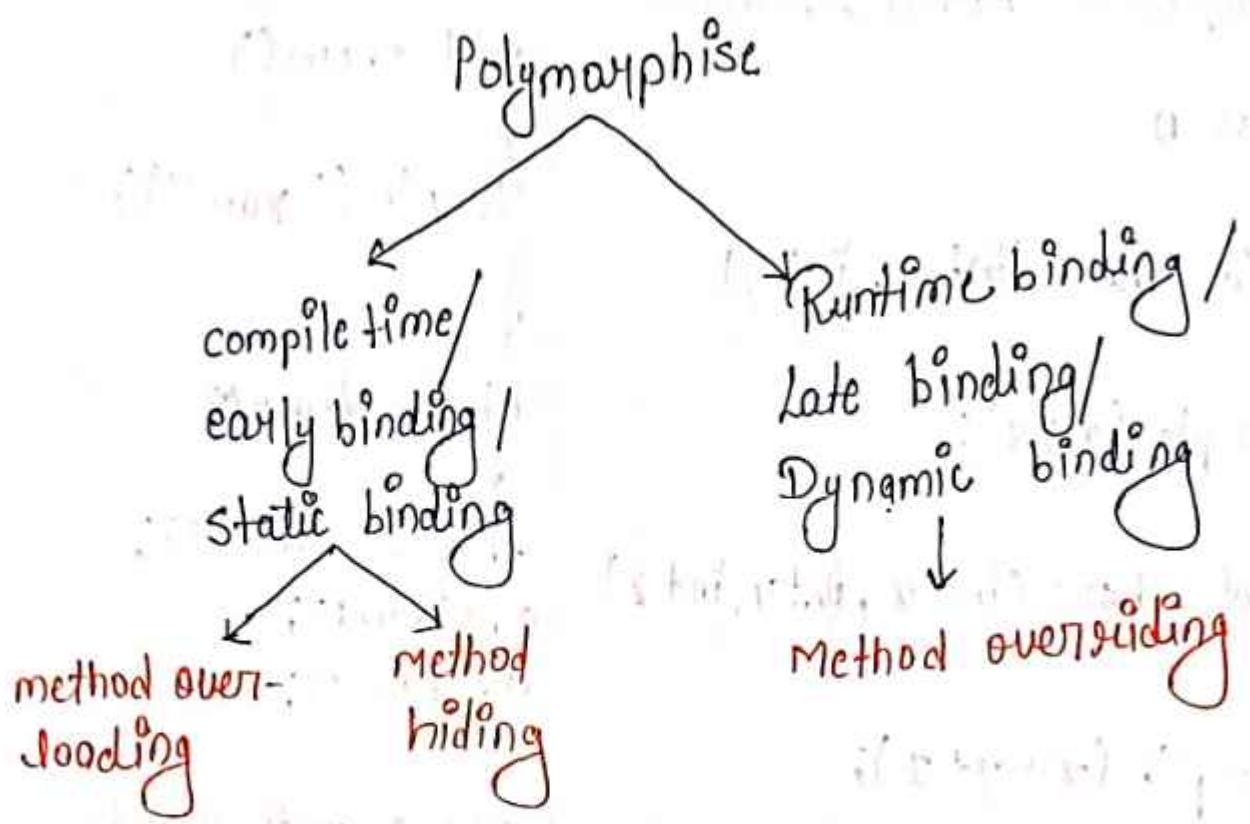
~~& for (String s2 : s)~~

3

```
s.o.println(s2);
```

3

O/p  $\Rightarrow$  my  
name  
is  
ram



→ Note :— we can make method private, static and final. but private, static and final method को कभी गत override नहीं किया जा सकता। लाए private, static and final method को overload कर सकते हैं।

→ Method overloading → एक class में same name but different param. के साथ छना कर सकते हैं।

→ C के pass polymorphism नहीं है। क्षमता, C में method overloading की facility provide नहीं की है।

## Examples of Method Overloading -

```
class A
```

```
{
```

```
void show (int x, int y)
```

```
{
```

```
System.out.println (x+y);
```

```
}
```

```
void show (int x, int y, int z)
```

```
{
```

```
System.out.println (x+y+z);
```

```
}
```

```
class demo
```

```
{
```

```
A a = new A();
```

```
a.show (10, 20);
```

```
a.show (10, 20, 5);
```

```
O/P → 30
35
```

```
void show()
```

```
{
```

```
System.out.println ("ram");
```

```
class demo
```

```
{
```

```
A a = new A();
```

```
a.show();
```

```
a.show();
```

error : method show is already define in class A.

Example :

```
class A
```

```
{
```

```
void show()
```

```
{
```

```
System.out.println ("class A");
```

```
int show()
```

```
{
```

```
System.out.println ("ram");
```

```
return 100;
```

```
class demo
```

```
{
```

Example —

```
class A
```

```
{
```

```
void show()
```

```
{
```

```
System.out.println ("class A");
```

```
}
```

```

A a = new A();
int x = a.show();
S.o.println(x);
}

```

error: method show is already defined in class A

→ यह method name and parameter list same है but return type different है, तो ये method Overloading कहलाएगा।

→ JVM main method की execution time पर check करेगा।

Example :

```

class demo
{
 p.s.v.m (String, int)
 {
 S.o.println ("ram");
 }
}

```

error: please define the method main as :

~~Method~~

class demo

```

{
 p.s.v.m (String... ar)
}
```

```
S.o.println ("ram");
```

}

o/p → ram

Q → ये method के पास parameter different होना compulsory है या type of (different data type) की method overloading होगा या नहीं ?

Ans → yes, type of की method overloading होगा।

Example :

class demo

{

o/p. s.v.m (String

Example of type of —

class A

{

void show (int x, int y)

{

S. O. println ("int...");

}

void show (byte x, byte y)

{

S. O. println (" byte... ");

}

}

class demo

{

A a = new A();

a. show (1000, 2000);

}

O/p  $\Rightarrow$  int...

$\rightarrow$  यहाँ लाए value int के पास  
ही जाएगी।

Example :

class demo

{

A a = new A();

a. show (1000, 2000);

a. show (10, 20);

}

O/p  $\Rightarrow$  int...

int...

Example :

class A

{

void show (long x, long y)

{

S. O. println ("long... ");

}

void show (short x, short y)

{

S. O. println ("short... ");

{

}

class demo

{

A a = new A();

byte x = 1;

byte y = 2;

short a = 11;

short b = 12;

a. show (1000, 2000);

a. show (x, y);

a. show (a, b);

{

O/p  $\Rightarrow$  long

short

short

$\rightarrow$  O/p  $\Rightarrow$  according to order  
of type casting.

like : byte → short → int → Example — of ambiguous error

long → float → double

Example :

class A

{

void show(~~float~~ float x, float y)

{

System.out.println("float");

}

void show(Short x, Short y)

{

System.out.println("Short");

}

class demo

{

A a1 = new A();

byte x = 1;

byte y = 2;

Short a = 11;

Short b = 12;

a1.show(10, 20);

a1.show(x, y);

a1.show(a, b);

a1.show(10.8, 20.8);

}

error : no suitable method found  
for ~~double~~ show  
(double, double)

class A

{

void show(double x, int y)

{

System.out.println("double...int");

}

void show(int x, double y)

{

System.out.println("int...double");

}

double, int, double दोनों

जो stone के सफल हैं

class demo int, int double दोनों

में उपर उपर दोनों सफल हैं

A a1 = new A();

a1.show(10.8, 20);

a1.show(10, 20.8);

a1.show(10.8, 20.8);

a1.show(10, 20);

O/P ⇒ double...int

int...double

उपर

error : reference to show  
is ambiguous.

→ ~~bcz int - double और int~~

~~int जो उपर दोनों सफल हैं, and double~~

~~int के उपर जो पास नहीं~~

~~होता है।~~

# Example of method overriding -

3/1/2022

class A

{

void show()

{

S.0. pIn ("class A");

}

}

class B extends A

{ }

class demo

{

B b = new B();

b.show();

}

O/P => class A

Example : static <sup>Imp. week.</sup>

class demo

{

int x = 10;

psvm (S[ ] ar)

{

S.0. pIn (x);

}

}

error : ~~not a static variable x~~  
can not be referenced from  
static context.

Example :

class demo

{

int x = 10;

psvm (S[ ] ar)

{

x++;

S.0. pIn (x);

}

}

error : ~~not a statement~~

# static <sup>method</sup> always access static data.

# Mainly variables are of two types :

i) instance variable

ii) static variable

# Rules of static -

- Non static method can access non static and static both data.

• Non static means instance and instance without object access ~~not~~ ~~an~~ object

Example :

```
class demo
```

```
{
```

```
static int x = 10;
```

```
public void main (String args[])
```

```
{
```

```
System.out.println(x);
```

```
{
```

```
Output : 10
```

Example : of non static method.

```
class A
```

```
{
```

```
void show()
```

```
{
```

```
System.out.println("class A");
```

```
{
```

```
class demo
```

```
{
```

```
.show();
```

non static method show can  
not be referenced from  
static context.

- we can not access non-static  
method directly from  
class name.

static is a non-access  
modifier.

# यदि मैंने किसी भी  
method को static decl-  
are किया है तो हमें  
static method के लिए  
object create करने की  
need नहीं होती है, गे  
directly class name से  
access हो जाती है।

Example :

```
class A
```

```
{
```

```
static void show()
```

```
{
```

```
System.out.println("class A");
```

```
{
```

```
class demo
```

```
{
```

```
A.show();
```

```
A.show();
```

```
{
```

```
Output : class A
```

```
class A
```

Example :

```
class A
```

```
{
```

```
int x = 10;
```

```
void show()
```

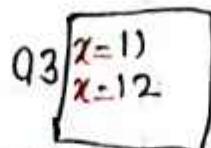
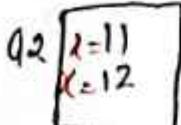
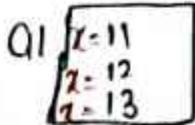
```
{
```

```

x++;
S.o.println(x);
}
}

class demo
{
A a1 = new A();
A a2 = new A();
A a3 = new A();
a1.show();
a2.show();
a1.show();
a2.show();
a1.show();
a3.show();
a3.show();
}

```



O/P  $\Rightarrow$

```

11
11
12
12
13
11
12

```

# Every object has corresponding instance / non-static variable are separate.

~~memory allocate~~ ~~free~~

Example:

Example :

class A

{

static int x = 10;

void show()

{

x++;

S.o.println(x);

}

class demo

{

A a1 = new A();

A a2 = new A();

A a3 = new A();

a1.show();

a2.show();

a1.show();

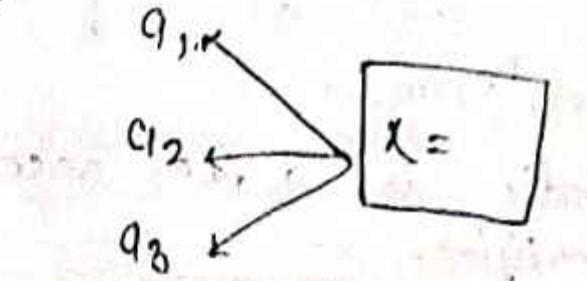
a2.show();

a1.show();

a3.show();

a3.show();

}



0/p → 11  
12  
13  
14  
15  
16  
17

Q + why main method is static  
Ans → JVM कमी भी से object  
create नहीं करता है।  
and main method static फल  
-ए है, ताकि JVM उसको बिन  
object create किए direct.  
main method से किए access  
कर सके।

# static → static variable

object के सभी में एक  
~~variable~~ के corresponding  
common memory allocate  
होती है। इसीलिए एक object  
का effect दूसरे object पर  
पड़ता है।

Syntax of run by JVM :-

demo00.main

\* Difference b/w instance and static variable.

Instance / non-static

i) हर एक object के correspond-  
onding separate memory  
allocate होती है।

ii) heap memory में खाली  
instance variable store  
होती है।

iii) instance variable object  
में किए access हो सकता है;  
इसीलिए इसे object variable  
में कहते हैं।

static

ii) हर एक object के correspond-  
onding common memory  
allocate होती है।

method area memory में खाली  
static variable store होती है।

static variable directly class  
name में access हो जाता है,  
इसीलिए इसे class variable  
में कहते हैं।

## Instance

static

→ There is no keyword for instance variable.

for using static variable we have to declare it static from static keyword

# क्या जॉन - सी requirement है, वह मा static का use करते हैं?

वह मा ऐसा चाहते हैं, कि एक object का effect दूसरे object में show होना चाहिए। तो इस condition में हम static का use करेंगे।

for example : enrollment no. in college / school.

Example of static

```
class Employee
{
 int id; → 4
 double sal; → 8
 long contact; → 8
 char name[10]; → 20
 static char cname[20]; → 40
}
```

Emp Employee e1; → 80  
1000 → 80,000

when static —

Example :

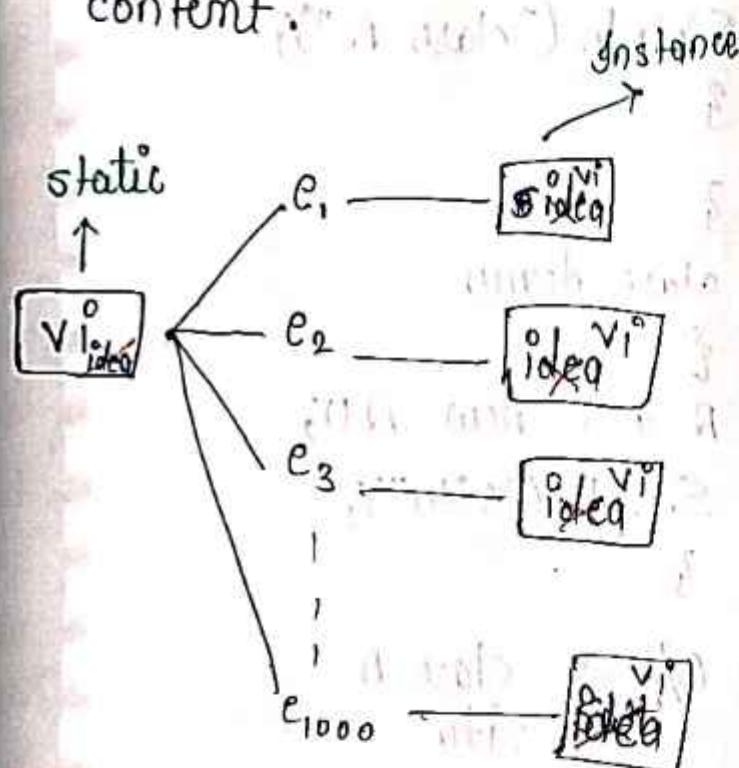
```
class A
{
 static int x=10;
}
class demo
{
 S.O. println(A.x);
}
```

O/p → 10

4/01/22

In the case of static variable:

- i) edit data की easy way
- ii) memory saving
- iii) no need to write repeat content.



Example :

```
class A
{
 void show2()
 {
 System.out.println("Show2 method");
 }
 void show()
 {
 show2();
 }
}
```

```
S.o.pn ("show method");
}
}
class demo
{
 void show();
}
O/p => show2 method
 show method
```

Example :

```
class A
{
 static void show2()
 {
 System.out.println("show2 method");
 }
 void show()
 {
 show2();
 }
}
class demo
{
 void show();
}
O/p => show2 method
 show method
```

Example :

class A

{

void show2()

{

S.o.println("show2 method");

{

static void show()

{

show2();

S.o.println("show method");

{

{

class demo

{

A.show();

{

Output:

Example :

class A

{

S.o.println("class A");

{

class demo

{

A a = new A();

S.o.println("sita");

{

Output: illegal start of type  
S.o.println("class A");  
^

Example : instance block

class A

{

// instance block

S.o.println("class A");

{

{

class demo

{

A a = new A();

S.o.println("sita");

{

O/p.  $\Rightarrow$  class A  
sita

# instance block :—

जैसे ही class का object  
create होता है, तो ही  
instance block automatic  
execute हो जाता है।

Example : of constructor  
& instance block

```
class A
{
 A() // constructor
}
```

```
S.o.println("class A cons.");
}
```

```
{ // instance block
```

```
S.o.println("class A instance
block");
}
```

```
{ // class demo
}
```

```
class demo
{

```

```
A a = new A();

```

```
S.o.println(" site");
}
```

Q ⇒ class A instance block  
class A cons.

Q ⇒ what will execute first  
instance block or constructor?

Ans ⇒ instance block will execute  
first. then constructor will  
be executed.

Example :

```
class A
{

```

```
A()
}
```

```
S.o.println("class A cons.");
}
```

```
{ A(int x)
}
```

```
S.o.println("class A parameter
cons.");
}
```

```
{
```

```
{
```

```
S.o.println("class A instance
block");
}
```

```
{
```

```
{ class demo
}
```

```
{
```

```
A a = new A();

```

```
* A a2 = new A(10);

```

Q ⇒ class A instance block  
class A cons.

class A instance block

class A parameter cons.

# instance block हर का

object के corresponding  
call होता। but construc-  
tor parameter की match

## ऐक्य call होता है।

Q → Difference between instance block and constructor?

### Instance block

### Constructor

- i) we can not pass any parameters into instance block.
- ii) Instance initialization block just have a body without any name or visibility type.
- iii) Instance block executes with every corresponding to every object. but instance block will execute first and then constructor will be executed.
- we can pass parameters into constructor.
- Constructor has the same name as class itself.
- Constructor executes from the match of parameter.

# Can we make more than one instance block in same class?

Ans Yes, we can make more than one instance block in same class.

Example:

```
class A
{
}
```

S.o.println("class A instance block");

5/1/22

O/P ⇒ class A show method

S.o.println("class A instance block"); Example : prove that JVM directly call main method.

class demo

{

    public void m()

{

    A a = new A();

    A a1 = (new A());

{

O/P ⇒

Example :

class A

S.o.println("class A instance block");

static void show()

S.o.println("class A show method");

class demo

. show();

class demo

{

S.o.println("class A instance block");

{

public static void main(st)

{

S.o.println("class A main method");

{

O/P ⇒ class A main method

It is proved that JVM never creates instance block. and directly call main method.

Example : Static block.

```
class demo
{
 static
}
S.o.pn ("Softwaves-1");
}
```

```
P s v m (st ar[])
{
```

```
S.o.pn ("class A main method");
}
```

}

O/p ⇒ ~~class~~ Softwaves-1  
class A main method

Static :-

1) Static block executes at  
the time of class load.

पहले class load होती है,  
then main method execute  
होती है।

2) static block हमेशा highest  
priority based, होता है,  
even main method को भी  
पहले

Q → can we make static  
block after main method  
yes, but all static block  
will execute first.

Example :

```
class demo
```

```
{
 static
}
```

```
S.o.pn ("S-1");
}
```

```
P s v m (st ar[])
{
```

```
S.o.pn ("class A main
method");
}
```

```

static
```

```
S.o.pn ("S-2");
}
```

O/p ⇒ S-1  
S-2

class A main method

Q → can we make more  
than one static block?  
yes,

Example :

```
class A
{
 static
 {
 System.out.println("class A static block");
 }
 void show()
 {
 System.out.println("class A show method");
 }
}

class demo
{
 public static void main(String args[])
 {
 System.out.println("class A main method");
 A a = new A();
 a.show();
 }
}
```

O/P → class A main method  
class A static block  
class A show method

Example :

```
class A
{
}
```

System.out.println("class A instance block");

```

}
static
{
 System.out.println("class A static block");
}
A()
{
 System.out.println("class A default constructor");
}
A(int x)
{
 System.out.println("class A param constructor");
}
```

void show()

System.out.println("class A show method");

}

class demo

```
{
 public static void main(String args[])
 {
 }
}
```

```
A a = new A();
A a2 = new A(10);
a.show();
```

```
S.o.pln("class A main method");
```

3  
3

O/P  $\Rightarrow$  class A static block

class A instance block

class A default constructor

class A instance block

class A param constructor

class A show method

class A main method

# असे हर एक object के  
corresponding instance

block execute होता है, क्योंकि  
हर एक object के corres-  
ponding static block exe-  
cute नहीं होता है। only  
class load होने पर ही  
call होता है, and class  
एक ही बार load होती है।

Example : can we store  
non-static data  
in static method.

class A

{

static void show()

{

int x=10;

S.o.pln ("x=" + x);

{

}

class demo

{

public static void

{

A.show();

{

}

O/P  $\Rightarrow$  x=10.

# we can store non-  
static data in static  
method.

Example :

class A

{

static void show

{

static int x=10;

S.o.pln ("x=" + x);

{

}

```

class demo
{
public void main (st, ar[])
{
 A . show();
}

error : illegal start of expression
static int x = 10;
^

```

local  
we can not make static  
variable to static variable.

Q + why do we not make  
local variable to static  
variable?

Ans bcz it is useless.  
that's why we can not  
make static variable  
as local.

Example :

```

class A
{
void show()
{
 static int x = 10;
 System.out.println ("x = " + x);
}
}

```

class demo

```
{
public void main (st, ar[])
{

```

```

A a = new A();
a . show();
}
}
```

error : illegal start of expr.  
static int x = 10;

6/1/22

- Example :

```

class A
{
 int x = 10;
 static
 {
 A a = new A();
 S.o.println("x = " + a.x);
 }
 void show()
 {
 S.o.println ("show method");
 }
}

class demo
{
 public void m()
 {
 A a1 = new A();
 a1.show();
 }
}

```

O/P : x = 10  
show method

# object से किसी भी type के content को नहीं access कर सकते हैं।  
means, not static and static both, के content की।

Example of cube :-

```

class A
{
 static int x = 10;
 static
 {
 S.o.println("x = " + cube(x));
 }
}

static int cube(int a)
{
 return a*a*a;
}

```

class demo

```

public void m()
{
 A a1 = new A();
 //S.o.println("x = " + cube(x));
}

```

O/P : x = 1000

Q Example without main meth- od.

```
class demo
{
 static
 {
 System.out.println("static block");
 }
}
```

Exception :- please define main method as public

```
sum();
```

Example :

```
class A
{
 void show()
 {
 System.out.println("class A ");
 }
}
```

```
class demo
{
 A a = new A();
 sum()
 {
 show();
 }
}
```

non static variable  
can not be referenced  
in a static context.

ऐसा क्यों - कि विलिंग है,  
परन्तु जावा में without  
main method का प्रोग्राम  
execute हो सकता है?

Yes, till the version 1.0  
but version 1.1 से की  
facility हो दी गई है।

# Can we make object  
static object.

Yes we can make  
object as static.

Example :

```
class A
{
 void show()
 {
 System.out.println("class A ");
 }
}
```

```
class demo
{
 A a;
```

static A a = new A();  
sum()  
{
 a.show();
}

O/p → class A

Example : Imp

```
class A
{
 int x = 10;
}

class B
{
 static A a = new A();
 int y = 25;
}

class Demo
{
 public void m()
 {
 System.out.println(B.a.x);
 System.out.println(B.y);
 }
}
```

Example of printStream method

```
import java.io.*;

class demo
{
 public void m()
 {
 PrintStream p = new PrintStream(System.out);
 p.println("softwaves");
 }
}
```

O/P  $\Rightarrow$  Softwaves

Example :

```
class A
{
 void show()
 {
 System.out.println("class A");
 }
}

class B
{
 static A a = new A();
}

class demo
{
}
```

~~Q~~  $\rightarrow$  println method क्या  
class के पास है। println  
method ~~is~~ PrintStream class  
की instance method है।  
PrintStream class is in  
IO package.

public void()

{

B.a.show();

B.a.show();

}

}

O/P ⇒ class A  
class A

Example :

```
import java.io.*;
class Ram {
 public static void main() {
 System.out.println("Abhi Sir");
 }
}
```

class demo → 1<sup>st</sup> class

{ p s v m () }

Ram.out.println("Abhi Sir");

}

}

O/P ⇒ Abhi Sir

~~internally same as java used.~~

System.out.println("ram");

i) println is the instance method of PrintStream class.

ii) PrintStream class is in io package.

iii) out is the static variable of System class that's why we can write as System.out

iv) out System class or static variable हीने द्वारा लिया जाता है - लिया जाता है PrintStream class का object द्वारा लिया जाता है

v) out.printStream द्वारा लिया जाता है object ही, that's why we can write as out.println and out System class or static variable ही। इसलिए, we can write as System.out.println();

# Java में JDK 1.6 version  
में without main method  
program run कर सकते हैं।

JDK 1.6 version का पहले  
static block execute हो  
जाता था then exception  
DISPLAY होती थी।

JDK 1.7 version से जैसे  
compulsory कर दिया गया  
कि main method present  
होने पर तो program  
execute होगा।

O + P s v m (st ar [ ])

# • main is the starting point of execution of

• program. ← main Example :  
main method never ret. class A  
with any value. that's why  
it's return type is void. ← void

• jvm directly call  
main method with  
class name without  
creating any object.  
that's why we write  
as static. ← static

• public →

- jvm is in c drive.
- jvm can't see main method को बड़ी से भी access कर सके। that's why हम main method को public declare करते हैं।
- Command line argument का purpose को string args [ ] का use किया गया है, ताकि हम उपर से input के सके। ← (string, ar [ ]) ←

main

class A

{

S. o. p () ("class A instance block");

}

A ()

{

S. o. p () ("class A constructor.");

}

class B extends A

C c = new C();

}

S.o.println(" class B instance  
block ");

↳ class A Instance block  
↳ class A constructor  
class B instance block  
class B constructor  
class C instance block  
class C constructor.

B()

}

S.o.println(" class B constru-  
ctor ");

# Constructor can never be  
inherited.

# instance block को compile

ले एक constructor में  
super() के नीचे रख देता  
है। यहाँ भी constructor  
call होगा। उसके पहले  
instance block definitely  
call होगा।

class C extends B

Example :

class demo

{

    public void m()

{

        out.println(" ram ");

}

error : can't find symbol

out.println(" ram ");

}

class demo

{

    public void m()

{

# Topic :- Static import

7/1/22

# हम static को import कर्ये क्योंकि pauseInt, Integer class की static method है।

परन्तु छी class के static data को without object create किए name के रूप के directly access करने के लिए हम static को import करते हैं।

Example :

```
import static java.lang.
System.*;
```

class demo

{

psvm()

{

out.println("ram");

out.println("ji");

{

O/P :- ram  
ji

Example :

```
import static java.lang.
```

```
System.*;
```

class demo

{

psvm()

{

```
int x = Integer.parseInt(ar[0]);
```

```
int y = Integer.parseInt(ar[1]);
```

```
out.println("sum = " + (x+y));
```

{

{

Example :

```
import static java.lang.
```

```
System.*;
```

```
import static java.lang.
```

```
Integer.*;
```

class demo

{

psvm()

```

int x = parseInt(ar[0]);
int y = parseInt(ar[1]);
out.println("sum: " + (x+y));
out.println(Integer.MIN_VALUE);
out.println(Byte.MAX_VALUE);
out.println(Byte.MIN_VALUE);

```

# Integer के class के पास एक हुआ static variable MAX\_VALUE है। जिससे कि इस Integer की maximum value को पता कर सकें।

~~MIN-MAX-VALUE~~ की Integer

class के पास एक हुआ static variable है।

ऐसी लिए हए data type के corresponding दीनी वाली variable इस वेक्तुर के समान है।

```
import static java.lang.
```

```
System.*;
```

```
class demo
```

```
psv m()
```

```
out.println(Integer.MAX_VALUE);
```

O/P :- 2147483647  
-2147483648  
127  
-128

Example :

```
import static java.lang.
```

```
System.*;
```

```
Integer.*;
```

```
class demo
```

```
psv m()
```

```
out.println(MAX-VALUE);
```

```
out.println(MIN-VALUE);
```

```
2147483647
```

```
-2147483648
```

Example: case 3 of ambiguous error:

```
import static java.lang.
System.*;
```

" " "

```
Integer.*;
```

" " "

```
Byte.*;
```

```
class demo
```

```
{ p s v m () {
```

```
out.println(MAX_VALUE);
```

```
out.println(MIN_VALUE);
```

}

}

Error: Reference to MAX-VALUE

is ambiguous

```
out.println(MAX_VALUE);
```

both variable MAX-VALUE  
in Integer and variable  
MAX-VALUE in Byte Match

Same error for MIN-VALUE.

Range of long → 92233720

368548808

to 92233720 36854775

807

Example:

```
import static java.lang.
```

```
System.*;
```

" " "

```
Integer.*;
```

" " "

```
Byte.*;
```

```
class demo
```

```
{ int MAX-VALUE = 100;
```

```
p s v m ()
```

```
{
```

```
out.println(MAX-VALUE);
```

}

}

Error: non static variable

MAX-VALUE cannot

be referenced from a

static context.

Example:

```
import static java.lang.
```

```
System.*;
```

" " "

```
Integer.*;
```

" " "

```
Byte.*;
```

```
class demo
```

```
{ static int MAX-VALUE =
```

```
p s v m ()
```

```
out.println(MAX-VALUE);
```

Output : 100

F Wrapper class has all the strange corresponding to every data type.

Example :

```
import static java.lang.
System.*;

Byte.*;

class demo
{
 public void m()
 {
 MAX-VALUE = 100;
 out.println(MAX-VALUE);
 }
}
```

error : cannot assign a value to final variable MAX-VALUE.

```
MAX-VALUE = 100;
```

Java में यही कृति एक capital letter में होता है, वे const. नहीं हैं। Like in Java Constants are in capital letter.

# we declare constants by final in Java.

# MAX-VALUE static हीने के साथ - साथ final भी है। That's why हम उसकी value को भी भी change नहीं कर सकते।

Example :

```
class demo
```

```
{
```

```
int x = 10;
```

```
public void m()
```

```
{
```

```
S.0. println(x);
```

→ case-1:

```
demo a = new demo();
```

```
S.0. println(a.x);
```

→ case-2:

```
S.0. println(this.x);
```

→ case-3:

Explain: non static variable x can not be referenced from a static context. → case-1:

O/p ⇒ 10 → case-2:

Explain: " " → case-3:

# object से हम किसी भी type के data को access कर सकते हैं। but depend जरूरता है, कि हमने object ऊपर पर create किया है।

# we can not use this, super keyword in static.

Q: ऐसी जॉन-सी condition है। जिसमें main method की manually program द्वारा call कर सके ?  
Ans, we can call main method manually by itself.

# Example of calling main method manually.

```
class demo20
{
 public static void main()
 {
 System.out.println("Hello World");
 }
}
```

```
S.o.pln ("demo20");
```

}

```
class demo21
```

{

```
public static void main()
```

{

```
demo20.main();
```

```
S.o.pln ("demo21");
```

}

}

O/p ⇒ demo20

@ demo21

# It is compulsory to pass string type array in main method.

Example :

```
class demo20
```

{

```
public static void main(String args)
```

{

```
S.o.pln (args[0]);
```

```
S.o.pln (args[1]);
```

```
S.o.pln ("Demo20");
```

}

}

```

class demo21
{
 public static void main(String[] args)
 {
 demo20.main(new String[]
 {
 "aaa", "bbb"
 });
 System.out.println("demo21");
 }
}

O/P → demo20
 aaa
 bbb
 demo2D
 demo21

```

→ points of static

static variable

static method

instance block

static block

System.out.println

public static void main()

static import

call main method manually.

Example of method overriding

class A

{

void show()

{

System.out.println("class A");

}

}

class B extends A

{

void show()

{

System.out.println("class B");

}

}

class demo

{

public static void main()

{

B b = new B();

b.show();

}

}

O/P → class A

class B

# method overriding :- *Upper*

*ent class* *in same method*

*name & same param*

list लेना कर मुश्यम् जूना।

Example :

class A

{

void show (int x, int y);

{

S. O. `println ("sum = " + (x+y));`

}

}

class B extends A

{

void show (int x, int y)

{

S. O. `println ("multi = " + (x*y));`

}

}

class demo

{

P S V m ()

{

B b = new B();

b. show (10, 20);

}

}

O/P → multi = 200

Example corresponding to  
Super keyword.

class A

{

void show (int x, int y)

{

S. O. `println ("sum = " + (x+y));`

}

}

class B extends A

{

void show (int x, int y)

{

Super. `show (x, y);`

S. O. `println ("multi = " + (x*y));`

}

}

class demo

{

P S V m ()

{

B b = new B();

b. show (10, 20);

}

}

O/P → sum = 30  
multi = 200

# we can write super keyword anywhere in method. but super

$$\begin{aligned} O/P \Rightarrow x &= 30 \\ &x = 20 \\ &x = 10 \end{aligned}$$

10/1/22

Should be first statement in constructor.

Example of command line argument :

Example of super & this.

class A

{

int x = 10;

}

class B extends A

{

int x = 20;

void show()

{

int x = 30;

S.0.println("x = " + x);

S.0.println("x = " + this.x);

S.0.println("x = " + super.x);

}

class demo

{

p s v m ()

{

B b = new B();

b.show();

}

class demo

{

p s v m (string [ ] a)

{

S.0.println("S-1");

int x = Integer.parseInt(a[0]);

S.0.println("S-2");

int y = Integer.parseInt(a[1]);

S.0.println(x/y);

S.0.println("S-3");

{

{

case : 1

10

2

O/P ⇒

case : S1

S2

S

S3

case : 2 0 10

S-1

S-2

0

S-3

S-1

S-2

Exception :

case : 3 10 0

# Arithmetic Exception: / by zero → EXCEPTION HANDLING

Case: 4 10 ab

~~o/p~~ o/p ⇒ S-1

S-2

Exception: Number  
format Exception  
for InputString: "ab"

Case: 5 ab 10

O/p ⇒ S-1

Exception: NFEFIS  
"ab";

Case: 6 10

S-1

S-2

Exception: AIOOBE

# यह लाइन में Except-  
-ion message आकर display  
होता है, उसके बाद को  
code execute नहीं होता।

# Exception हमें runtime  
पर होकर होती है,  
Exception कभी नहीं compile  
time पर होती है।

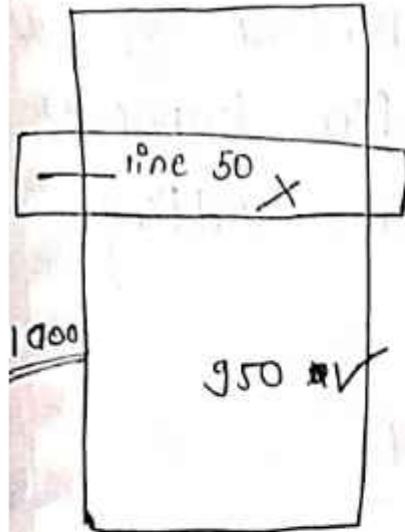
Q + what is Exception?

Ans → An Exception is an unwanted or unexpected event, which occurs during the execution of a program at runtime, that disrupts the normal flow of the program.

Q What is Exception Handling?

Suppose there is 1000 statements in your program and there occurs an exception at statement 50, rest of the code will not be executed i.e. statement 51 to 1000 will not run.

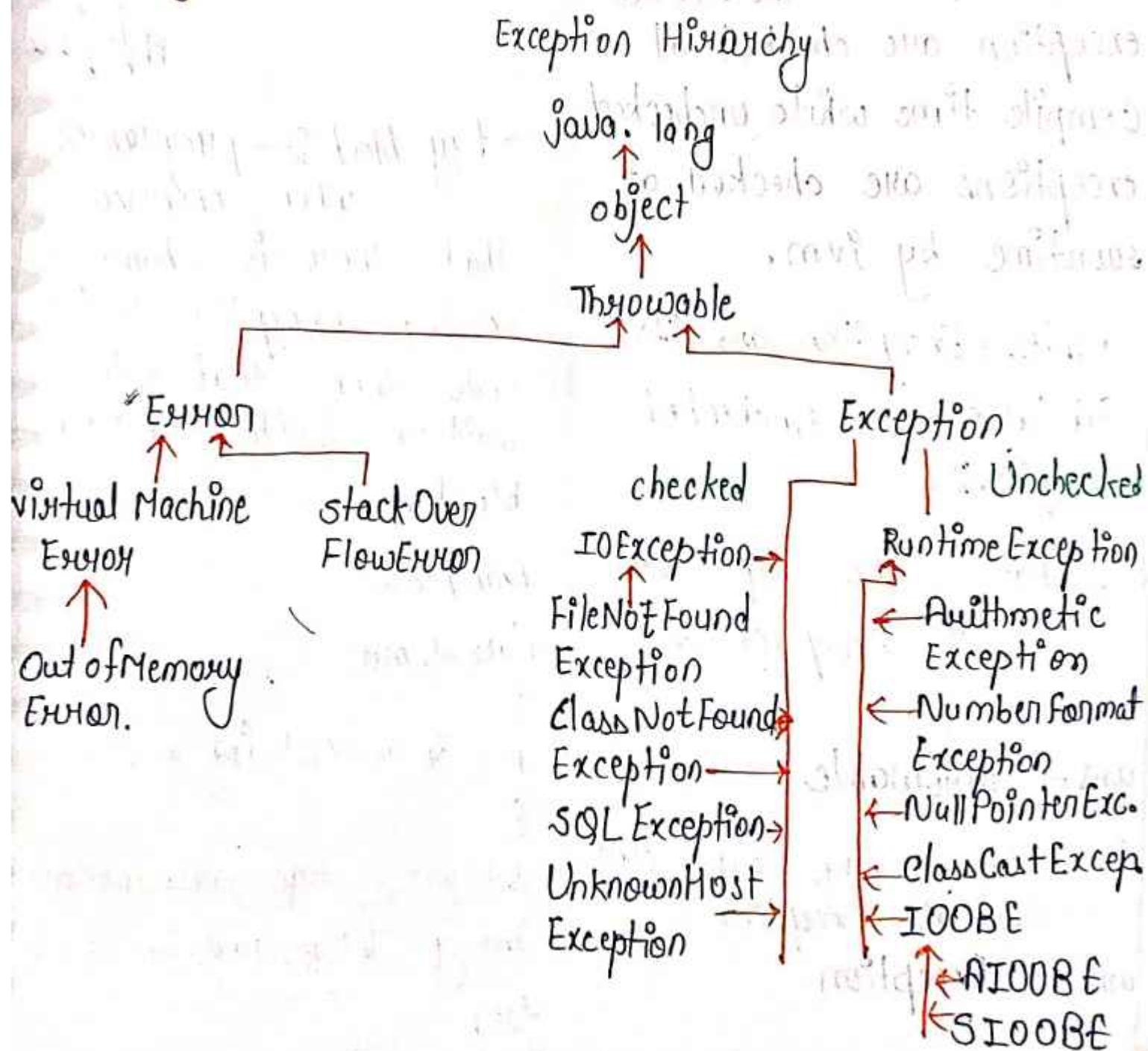
If we perform exception handling, rest of the statement will be executed. That's why we use exception handling in Java.



Qrwh

## # Throwable class lang package में क्या है?

### Exception Hierarchy



~~Q. What is~~

There are two types of Exception :

- i) Checked exceptions and
- ii) Unchecked exceptions

The main difference b/w checked and Unchecked exceptions is that the checked exception are checked at compile time while unchecked exceptions are checked at runtime by jvm.

Runtime Exception and its

Subclasses are Unchecked Exceptions.

Q → What is the super class

for all Exception and Error?

Ans → Throwable

Q → What is the super class

for all Exception?

Ans → RuntimeException

Q → Exception Handling keywords  
There are five keywords in Exception Handling :

→ try  
→ catch, finally  
→ throw  
→ throws  
→ ~~finally~~

11/1/22

→ try block :- programmer when observe

that there is chance of coming exception in the code then that code is written inside the try block.

Example :-

class demo

{

    public static void main(String args[])

{

        int x = Integer.parseInt(args[0]);

        int y = Integer.parseInt(args[1]);

    try

```
S.o.println(x/y);
```

```
S.o.println("softwaves - 1000");
```

```
{}
```

error : 'try' without 'catch',  
'finally' or resource  
declarations  
try

# we cannot use try block

single , we must use try  
with the catch , finally or  
with resource declaration  
otherwise it will give  
error.

catch block :-

Example : →

```
class demo
```

```
{}
```

```
psvm ()
```

```
{}
```

```
int x = Integer.parseInt(a[0]);
```

```
int y = Integer.parseInt(a[1]);
```

try

```
S.o.println(x/y);
```

```
{}
```

```
catch(ArithmeticException e)
```

```
{}
```

```
S.o.println("ok...");
```

```
{}
```

```
S.o.println("S-1000");
```

```
{}
```

```
}
```

Example :

```
class demo
```

```
{}
```

```
psvm ()
```

```
{}
```

```
int x = Integer.parseInt(a[0]);
```

```
int y = Integer.parseInt(a[1]);
```

```
catch(AE e)
```

```
{}
```

```
S.o.println("catch block...");
```

```
{}
```

```
S.o.println("S-1000");
```

```
{}
```

```
}
```

error : catch without try.

```
{}
```

# catch के बारे में जल्द हीना व्यवहार :

compulsory है) and try के साथ catch, finally and resource declaration त्रुटि भी हो सकता है।

Inputs of example-1 :

input 10 2  
o/p 5  
S-1000

input 10 0  
o/p catch block...  
S-1000

input 10 ab  
Exception in thread "main" java.  
lang.NumberFormatException:  
"ab"

# अगर try block में exception आएगी, तभी catch block execute होगा।

catch block →

the catch block will always come after the try block. no statement or block after can come b/w the try and catch block.

3  
catch(Exception-Name e)

5  
statement - - -  
3

when exception is generated in try block then catch block is executed. we can use multiple catch block with one try block.

The type of exception is generated in try block that type of catch block is executed.

Example of multiple catch block with one try block.

```
class demo
{
 public void m()
 {
 try
 }
```

```

int x = I.parseInt(a[0]);
int y = I.parseInt(a[1]);
S.o.println(x/y);
}
catch(AE e)
{
 S.o.println("catch block... ArithmeticException");
}
catch(NumberFormatException e)
{
 S.o.println("catch block... NFE");
}
S.o.println("S-1000");
}

input 10 2.
o/p 5
S-1000

input 10 0
o/p @catch block... AE
S-1000

input 10 ab
o/p catch block... NumberFormatException
S-1000

input 10

```

Exception : AIOOB  
 input : ab  
 o/p      catch block... NumberFormatException  
 S-1000

try block में प्रियंका तरीके  
 line में Exception आएगी।  
 तो वह direct jump होकर  
 catch block के पास चला  
 जाएगा।

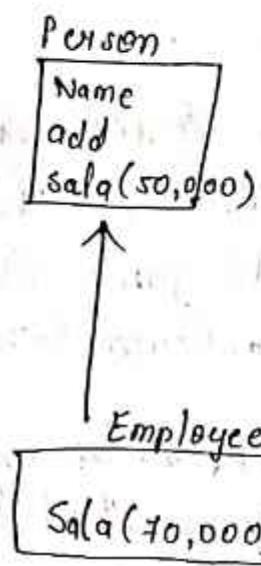
At a time only one  
 exception is generated in  
 try block bcz the  
 program is terminated  
 from that point and  
 if its corresponding  
 catch block is available  
 then it is executed.

→ finally block :- Always  
 executable  
 block. Is called finally  
 block.

12/1/22

## Polymorphism

Inheritance means predefined class के अपनी property को add करना।



# अगर न्में super की कोई method की definition पसंद नहीं आ रही है, तो उसे उसी superclass में override करके अपने according उसकी definition को change कर सकते हैं।

Example :

```

class A
{
 void show1()
}

S.o.pn ("Class A");

```

3  
5  
class B extends A  
8

B.show2()

5

S.o.pn ("class B");

3

class demo

5

P.S.V.m( )

5

B b = B new B();

b.show1();

b.show2();

3

O/P → class A

class B

Final :— final is a keyword of oops.

यदि हमने किसी class की final declare कर दिया तो उसे कभी नहीं able inherit कर सकते।

Example:

~~class A~~ final class A

{ void show1()

{

System.out.println("class A");

}

class B extends A

{

System.out.println("class B");

}

class demo

{

B b = new B();

b.show1();

b.show2();

}

}

error: cannot inherit from  
final A.

# उगरे हम याहते हैं कि  
स्मारी method को कोई  
override न कर सके।  
तो इस situation में  
हम उस method को  
final declare कर देंगे।  
# we cannot override

final method.

class A

{

final void show()

{

System.out.println("class A");

}

class B

{

void show()

{

System.out.println("class B");

}

class demo

{

B b = new B();

b.show();

}

error: show() in B cannot  
override show() in A

# method of super class  
is called overridden and  
method of sub class is  
called override.

# अगर हमने किसी variable को final declare किया है, तो उसकी values एवं भी change नहीं कर सकते।

Example :-

class B

{

```
final int x = 10;
void show()
{
 x = 20;
 System.out.println("x = " + x);
}
```

}

class demo

{

```
B b = new B();
b.show();
```

}

error : cannot assign a value to final variable x.

x = 20;

# final variable की default value क्या भी नहीं होती है वो blank होता है।

Example of private method :

class A

{

private void show()

{

System.out.println("class A");

}

class B extends A

{

class demo

{

B b = new B();

b.show();

}

error : cannot find symbol  
b.show();  
^

Example : not a example of class A overriding :-

{

private void show()

{

System.out.println("class A");

}

class B extends A

{

```

void show()
{
 S.o.println("class B");
}

```

class demo

```

b. show();
}

```

O/P  $\Rightarrow$  class B

# we can not override to private.

# method overriding के case

में class inheritance / super

class का class sub class

में आना compulsory है।

we can redclare that method in sub class. but can not override.

Example:

class A

```

{
 private void show()
}

```

S.o.println("class A");

class demo

```

{
 private void m()
}

```

A a = new A();

a.show();

}

error: show() has private access in A.

a.show();

<sup>^</sup>

Example :

class A

```

{
}

```

int x; // case-1

static int x; // case-2

final int x; // case-3

void show()

```

{
}

```

S.o.println("x = " + x);

}

}

class demo

```

{
}

```

a.show();

}

O/P  $\Rightarrow$  case 1  $\rightarrow$  0

case 2  $\rightarrow$  0

case 3  $\rightarrow$  error: variable x is not initialized in the default constructor

What is blank final variable? Example:

It is a variable that does not have any default value.

class A

{

final int x;

void show()

{

x = 20;

S.O. println(x);

}

class demo

{

A.show();

}

O/P → ~~error~~ error: can not assign a value to final variable x.

Example:

class A

{

final int x;

A()

{

x = 55;

}

void show()

{

S.O. println(x);

}

class demo {

# why ~~default~~ blank final variable?  
instance and static variable  
की तरह final variable की  
वैलिड default value zero फ्रैमिले  
नहीं रखी। क्योंकि अगर final  
variable की default value  
zero रख देते। तो किस की  
use less हो जाता है। इस  
runtime पर final variable में  
value assign कर सकते। फ्रैमिले  
final variable की def की  
blank रखा गया है।

# blank final variable में इस  
value ~~default~~ constructor  
की help से assign कर  
सकते हैं। ~~किसी~~

a. show();

}

O/P → 55

Example :

class A

{

final int x;

A()

{

x = 55;

}

A(int a)

{

S.o.pn ("a = " + (a+a));

}

void show()

{

S.o.pn ("x = " + x);

}

class demo

{

A a = new A();

A a2 = new A(100);

a. show();

~~a. show();~~

Error: variable x might not  
have been initialized.

इन एक्सप्रेसियन में इन्टर्नल  
construction है, जोकि  
सभी ऐसी final variable  
में value assign करना  
compulsory है।

Example :

class A

{

final int x;

A()

{

x = 55;

}

A(int a)

{

x = a;

}

S.o.pn ("a = " + (a+a));

}

void show()

{

S.o.pn ("x = " + x);

}

class demo

{

A a = new A();

A a2 = new A(100);

a. show(); a2. show();

}

O/P  $\Rightarrow$  a = 200  
x = 55  
z = 100

# Can we assign value in blank final variable with the help of instance block?

Example:

class A

{

final int x;

{

z = 55;

}

A()

{

S.o.println("Default cons.");

}

A(int a)

{

S.o.println("Param. const.");

}

void show()

{

S.o.println("x = " + x);

}

class demo

{

A a = new A();

A a1 = new A(100);

a.show(); O/P: Default cons

a1.show();

O/P: Param.cons

x = 55

x = 100

yes, becz instance block अपने code की construction में super के नीचे से पास put हो देता है।

# अगर हमने blank final variable में value की instance block की help से assign कर दिया है, तो उस construction की help से किसी off condition में उसमें value assign नहीं कर सकते।

Example:

class A

{

final int x;

{

x = 55;

}

A()

{

S.o.println("Default");

}

A(int a)

{

x = a;

```
S.o.println("Parameterized");
{
 void show()
 {
 S.o.println(x);
 }
}
```

```
class demo
{
 A a = new A();
}

```

class demo

```
{
 A a = new A();
}
```

```
A a1 = new A(100);
```

```
a.show();
```

```
a1.show();
```

```
}
```

Output:

Example : static blank final variable

class A

```
{
 static final int x;
```

```
A()
```

```
{
```

```
x = 55;
```

```
}
```

```
void show()
```

```
{
```

```
S.o.println("x = " + x);
```

```
}
```

# we can make blank final variable as static.

# we can not assign value in static blank final variable from constructor. but we can assign value from static block.

class A

```
{
 static final int x;
```

```
static
```

```
{
```

```
x = 55;
```

```
}
```

```
void show()
```

```
{
```

```
S.o.println(x);
```

```
}
```

```
class demo
```

```
{
```

A a = new A();

g. show();

3

O/p  $\Rightarrow$  55

# Java has 4 access specifier:

▷ public

▷ private

▷ protected

\* object के corresponding blank final variable का default access specifier होता है।

# The final keyword can be applied with the variable. Q:- Can we declare a constructor that have no value. It is as final?

Ans:- No, bcz constructor can never be inherited.

Use:- If you want to create a variable that is

initialized at the time of creating object and once initialized may not be changed. It is useful for

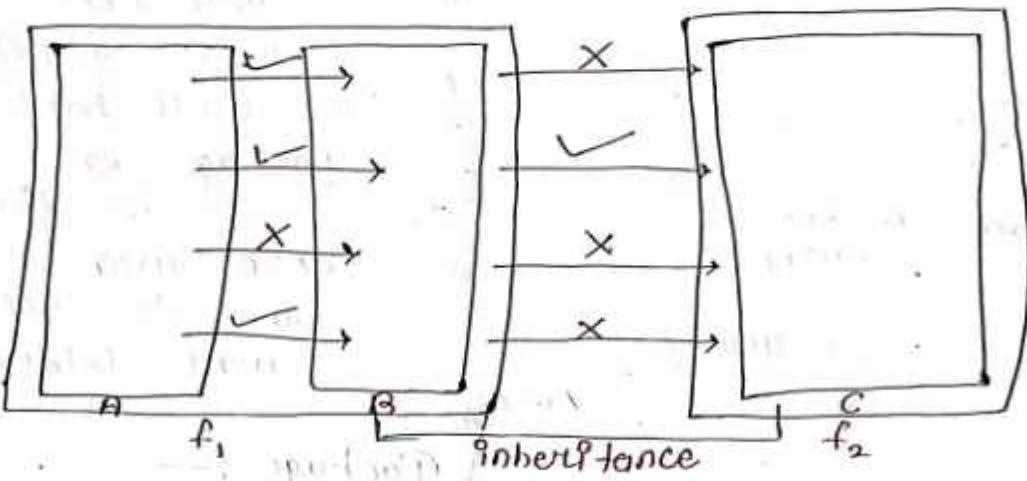
example PAN CARD number of employee.

Blank final variable is used to create immutable objects (object whose members can't be changed once initialized).

# final parameter:- If you declare any parameter as final. You can not change the value of it.

# When files are in same folder. From one class you can access the data of another class.

default  
 public  
 private  
 protected



case 1 :   
 demo.java → folder 1  
 A.java → folder 2

Example : when 2 classes in same folder

```

class A
{
 void show()
 {
 System.out.println("class A");
 }
}

```

```

class Demo
{
 public void m()
 {
 A a = new A();
 a.show();
 }
}

```

$f_1 = \text{folder 1} \quad \begin{matrix} A \\ B \end{matrix}$  } classes  
 $f_2 = \text{folder 2} \quad \begin{matrix} C \\ D \end{matrix}$  } classes

a.show();  
 a.show();

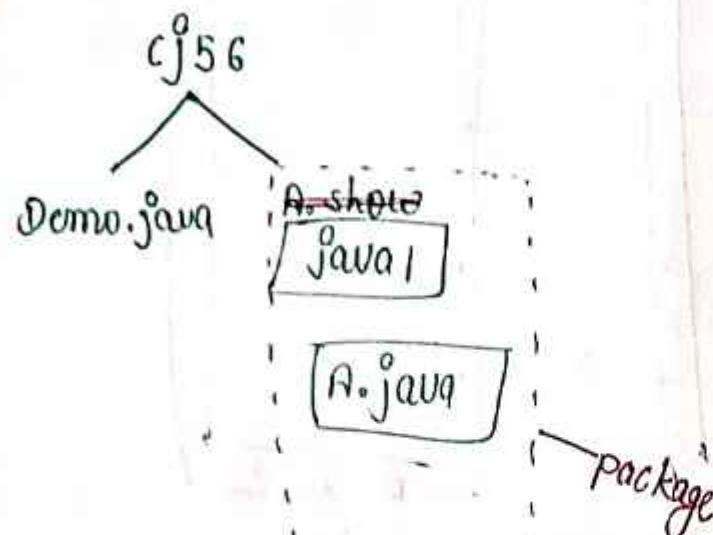
3

O/P → class A  
class A  
class A

# In protected we can not access one folder data into another.

but if we inherit  
 $f_1$  data into  $f_2$ . So it is possible to access the protected data

case - 2 :



If you want to access data of another folder. So you have to create package. eg. in the below of core java is folder. You want to access data of java1. folders.

### Package :-

- 1) package is made by package keyword.
- 2) folder name and package name must be same.
- 3) package is accessed from import keyword.  
import javal.A;

```
import javal.A;
class Demol
{
 public void m()
 {
 A a = new A();
 a.show();
 a.show();
 }
}

package javal;
public class A
{
 public void show()
 {
 System.out.println("class A");
 }
}

O/P => class A
 class A
```

case : 3 →

```

import java.io.A;
class Demol
{
 A a = new A();
 a.show();
 a.show();
}

```

```

package java.io;
public class A
{
 public void show()
 {
 System.out.println("class Nam");
 }
}

```

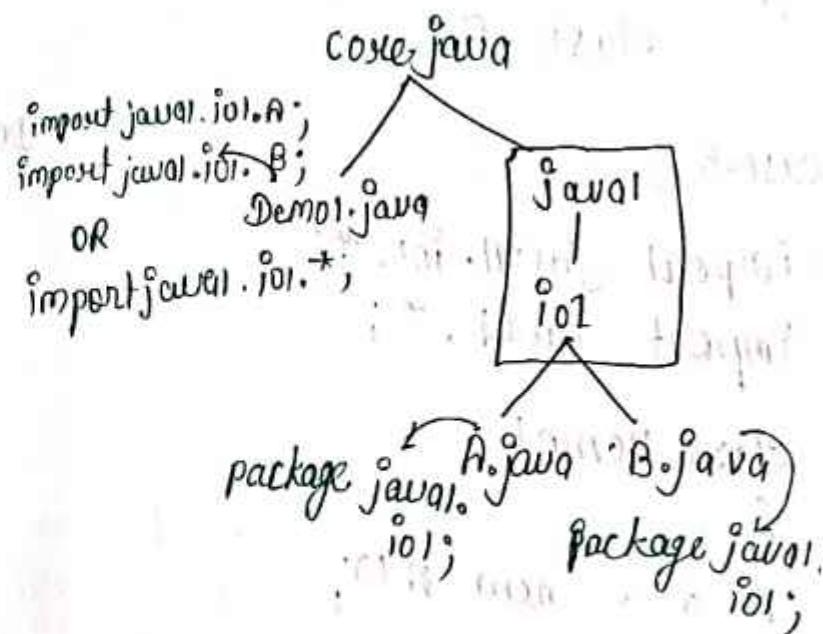
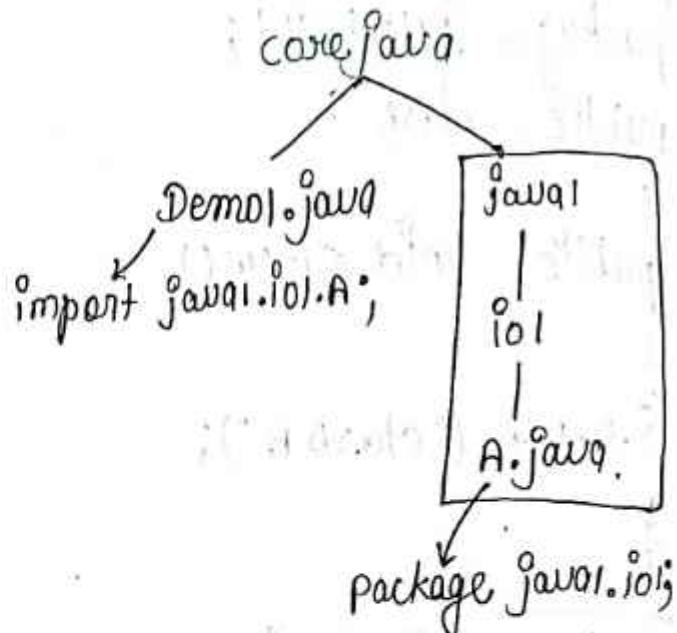
O/p ⇒ class Nam  
class Nam

case - 4 :

```

// import java.io.A;
// import java.io.B;
import java.io.*;
class Demol
{
 A a = new A();
 a.show();
 B b = new B();
 b.show();
}

```



```

package java1.io;
public class A
{
 public void show()
 {
 System.out.println("class A");
 }
}

```

`*`; → means, we can access the classes inside the folder. We can not access the folder.

```

package java1.io;
public class B
{
 public void show()
 {
 System.out.println("class B");
 }
}

```

O/p ⇒ class A  
class B

case-5 :

```

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

```

class Demo1

{

```

A a = new A();

```

a.show();

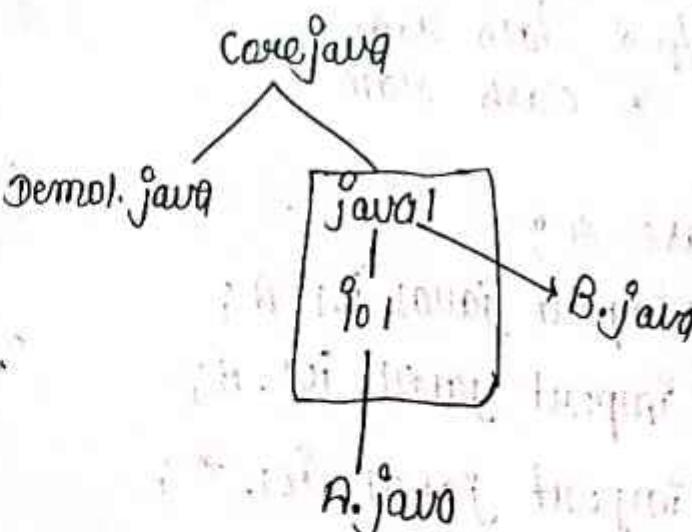
```

B b = new B();

```

b.show();

b.show();



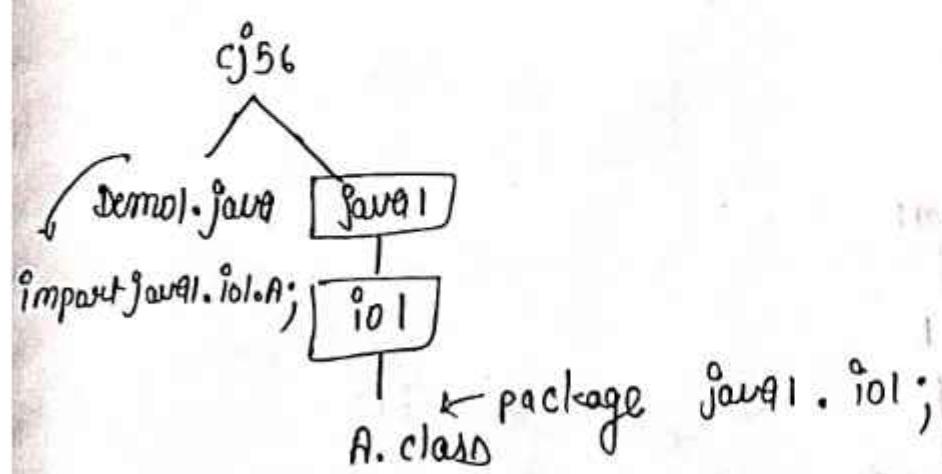
```
package java.io;
public class A
{ public void show()
S.o.println("class A");
}
```

---

```
package java.io;
public class B
{ public void show()
S.o.println("class B");
}
```

O/p → class A  
class B  
class B.

# program की run का class  
file से किया होता है।



Case-6 :

```

package java1.io1;
public class A
{ public void show()
S. o. pln ("class A");
}

```

```

import java1.io1.*;
class Demo1
{
A a = new A();
a.show();
}

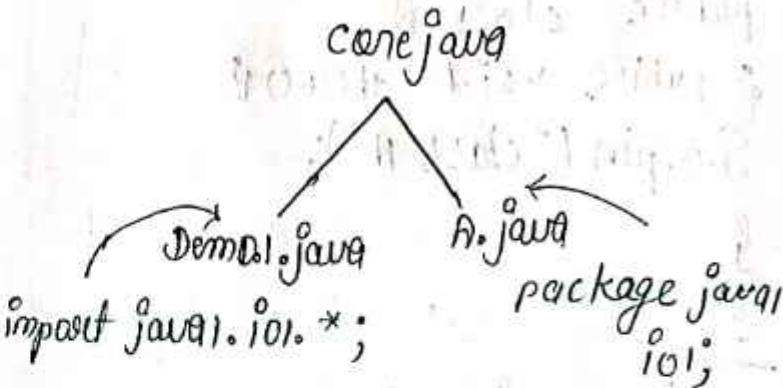
```

error : package java1.io1 does not exist

file does not contain class A.

compile :- javac -d . A.java

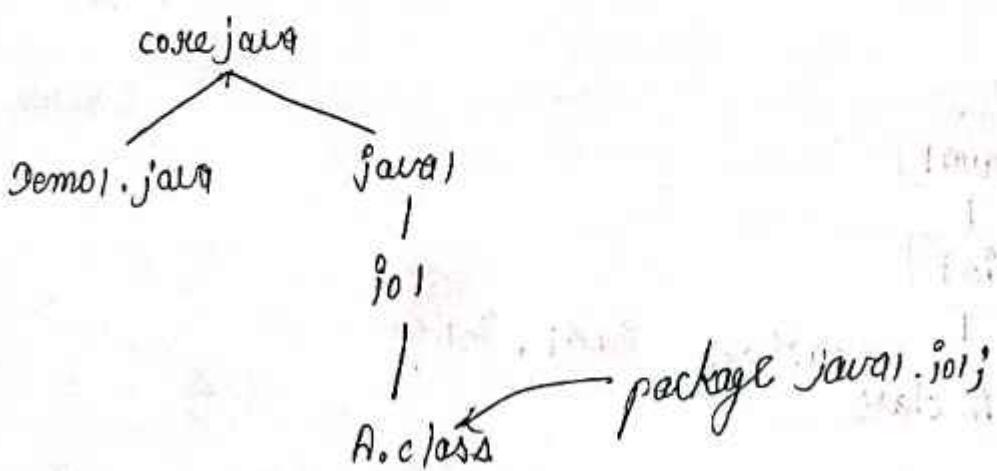
O/P → class A



-d → directory

- → जो path open किया है उसके अंदर से folder बन दूर ready होना चाहिए।

# package में हम जो path लिखेंगे 'अब के folder बन दूर ready हो जाएगा।'



# अगर folder नहीं होगा तो को automatic create कर देगा।  
अगर अगर होते तो उन्हीं को use कर देगा।

case-# : To access a class from one drive to another drive without package import.

class Demo1

{

A a = new A();

a.show();

a.show();

}

public class A

{

public void show()

{

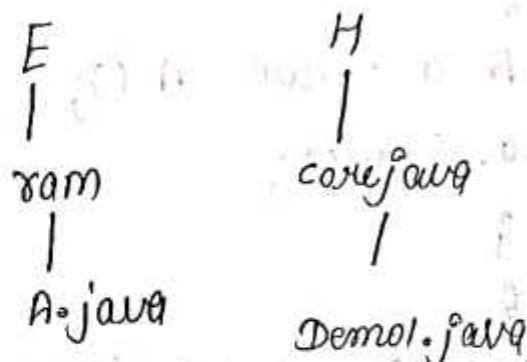
S.o.println("class A 123");

}

\* set classpath = E:\ram;  
.; %classpath%.

compile then run

%> class A 123  
class A 123



# एक drive से दूसरे drive के में class को direct access करने के लिए without package access करने के लिए हमें path set करना होगा।

set classpath = E:\ram;  
.; %classpath%

temporary path  
permanent

Case - 8:

```
import java.util.*;
class Demo1
```

{

```
A a = new A();
a.show();
```

}

```
package java.util.io;
```

```
public class A
```

{

```
public void show()
```

{

```
System.out.println("class A say");
```

}

```
javac -d class1 A.java
```

Case - 9:

```
import java.util.*;
class Demo1
```

{

```
A a = new A();
```

```
a.show();
```

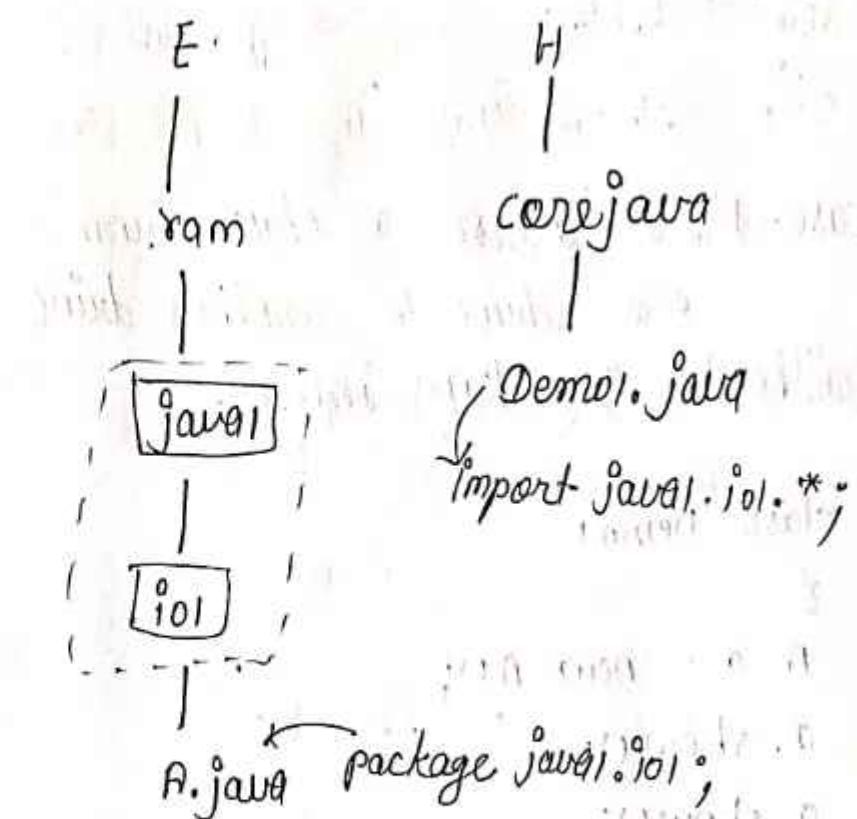
```
a.show();
```

}

```
package java.util.io;
```

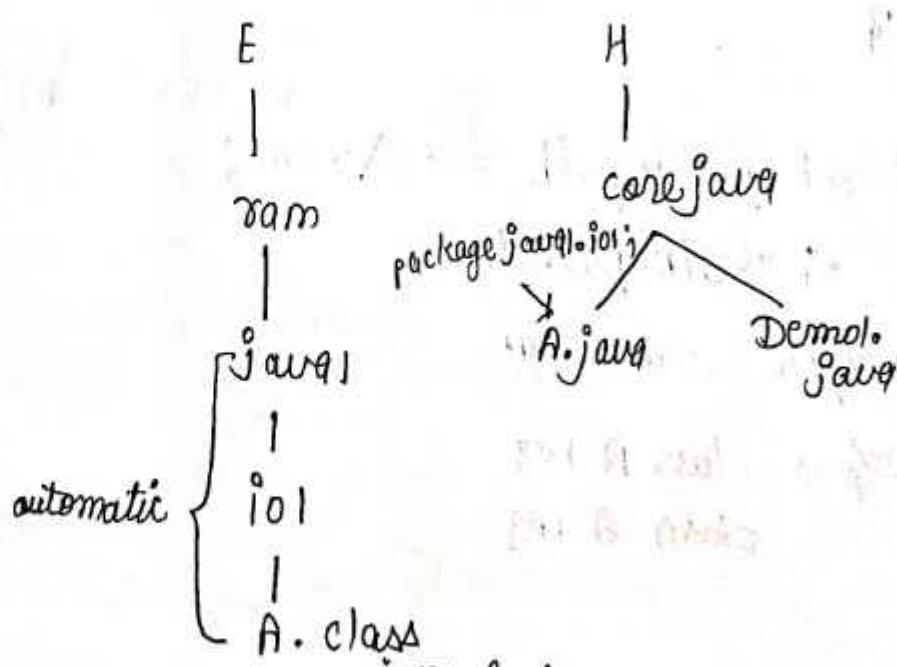
```
public class A
```

{



# program run करने के लिए  
 • class file की need होती है।  
 and for compilation .java file

Case - 10:



command: javac -d A.java

javac -d E:\ram A.java

```
public void show()
{
 D.o.println("class A 321");
}
```

# temporary path cmd के  
class close होने के साथ  
it destroy हो जाता है।  
we can also set  
permanent path.

```
%> class A 321
class A 321
```

17/1/22

## Abstract class or Interface

# Pure Virtual function :-

Abstract class :- Abstract class को abstract keyword की help से declare करते हैं।

2) We can not create object of Abstract class.

Example :-

abstract class A

{

void show()

{

S.o.println("class A ");

{

}

class demo

{

A a = new A();

a.show();

}

Error: A is abstract; can not be instantiated.

Example

abstract class A

{

void show()

{

S.o.println("class A ");

3) abstract class का data access करने के लिए inheritance ~~use~~ का use करना compulsory है। means, abstract class को inherit करना compulsory है।

4) Sub class के object से abstract class का data का access कर सकते हैं।

class B extends A

{

{

class demo

{

B b = new B();

{

b.show();

}

# Abstract method :- method without body is called as Abstract method.

\* Abstract method की abstract keyword से declare करना compulsory है।

Example :

```
class A
{
 void show();
}
```

Error :- missing method body, को declare abstract.

Example :

```
class A
{
 abstract void show();
}
```

Error :- A is not abstract and does not override abstract method show() in class A

~~if~~ sub class में super class की all abstract method की body की define करना compulsory होता है

3) यदि एक class में एक वा abstract method है, तो उस class की abstract declare करना compulsory है।

Example :

```
abstract class A
```

```
{
```

```
abstract void show();
```

```
{
```

```
class demo
```

```
{
```

```
A a = new A();
```

```
{
```

Error : A is abstract ; can not be instantiated

# 5) यदि हमने sub class में super class की all abstract methods को define नहीं किया। तो sub class को abstract declare करना होगा।

Example :

abstract class A

{

abstract void show();

{

class B extends A

{

}

class demo

{

B b = new B();

{

error: B is not abstract  
and does not override  
abstract class A  
method show in

Example :

abst. abstract class A

{

abstract void show();

abstract void show2();

{

Example :

abstract class A

{

abstract void show();

{

class B extends A

{

void show()

{

S.o.pn ("class B");

{

{

class demo

{

B b = new B();

b.show();

{

O/P ⇒ class B

# we can make a method with body in abstract class. as well as we can create a method without body.

class B extends A

void show()  
{

S.o.println("class A");  
}

}

class demo

B b = new B();  
b.show();  
}

error: class B is not abstract and does not override abstract method show() in class A

Example :

~~class~~ abstract class A

{  
static void show()  
{

S.o.println("class A");

}

}

class demo

{  
A.show();

}

O/p :> class A

Q: Can we declare main method as abstract?  
Yes, bcz jvm access main method directly by class name.

abstract class demo

{

S.o.println("class demo main method");

}

O/p :> class demo main method.

### Interface

Example :

abstract class A

{

void show()

{

S.o.println("class A");

}

class B extends A

{

class demo

{

B b = new B();

b.show();

}

}

O/p :> class A

~~O/p~~  $\Rightarrow$  class A

नि भारा ४।

- # Interface is corresponding to abstract class.

Example :

```
class demo
{
 int i;
 for (i=1; i<=10; i++)
 {
 continue;
 S.o.println(i);
 }
 S.o.println("ram");
}

O/p :- 9am
```

error : - Unreachable Statement

- Q + Can we make constructor in abstract class ?

yes,

- # ~~we can't~~ sub class of ~~constructor call~~ ~~so~~ object create ~~so~~ it, it automatically super class of constructor call

Interface

- # Interface के interface keyword की help से बना ज़रूर ready होते हैं।

- # we can not create a method with body in interface.

- # Interface के अंदर only abstract method हो सकते हैं।

Example : -

interface Interf

```
{
 void show();
}
```

{ }

error :

- # we can never create object of interface. if we create object of interface, then it will display an error.

Example:

```
interface Inter1
{
 abstract void show();
}
class demo
```

```
Inter1 i = new Inter1();
```

error: Inter1 is abstract;  
can not be instantiated.

# interface को data को access  
करने के लिए interface को  
implement करना compulsory है,  
interface को implement  
implements keyword से  
जरूरते हैं।

# subclass में @interface  
को पास जितनी भी abstract  
method है उन सभी को  
define करना compulsory है  
अगर इस ऐसा नहीं करते हैं  
तो subclass को abstract  
class बनाना होगा।

Example:

```
interface Inter1
{
 abstract void show();
}
class A implements Inter1
{
}
```

```
A a = new A();
```

error: class A is not  
abstract and does  
not override Inter1

Example: ~~Imp~~

```
interface Inter1
{
 abstract void show();
}
class A implements Inter1
{
 void show()
 {
 System.out.println("class A");
 }
}
class demo
```

A a = new A();

3

~~O/p → class A~~

Error: show() in A cannot implement show() in

→ Interf

void show()

attempting to assign weaker access privileges; was public

# method की public declare  
त करने पर attempting  
to assign error msg  
मात्र display होता है।

Example of functional interface

interface Interf

{

abstract void show();

{

class A implements Interf

{

public void show()

{

S.o.println("class A");

{

{

O/p → class A

Example: way of creating meth in interface

interface Interf

{

|| abstract void show();

|| public abstract void show();

|| public void show();

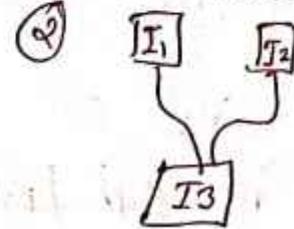
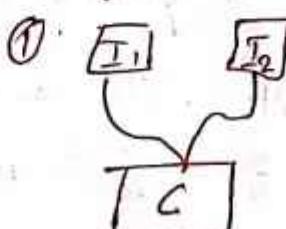
void show();

{

case 4: class A

# with the help of interface we can achieve multiple inheritance.

from two ways we can create multiple interface inheritance



# interface में all the methods inside the interface are by default public and abstract. means, java compiler 3नके आगे public को keyword लिख देता है।

Example :

class A

```
public void show()
{}
```

```
System.out.println("class A");
```

class B extends A

```
{}
void show()
```

```
System.out.println("class B");
```

class demo

```
B b = new B();
```

```
b.show();
```

Output: attempting to assign

Q+ अगर In the case of method overriding, अगर उपर्युक्त class को pub method को public declare किया है and sub class की method default है, तो error msg आजूर display होगा।

Q+ Can we make protected as default?

No

# weaker access privilege

# private → default → protected → public

अगर हम इसका opposite order use करते हैं, तो attempting to assign weaker access privilege error msg आजूर display होगा।

but अगर super में public है, तो sub class में भी public होना चाहिए

Type 1

# Example of ~~multiple inheritance~~ ~~multiple inheritance~~

interface Int1

{

void show1();

}

Interface Int2

{

void show2();

}

class A implements Int1,  
Int2

{

public void show1()

{

S.O. println();

}

}

class demo

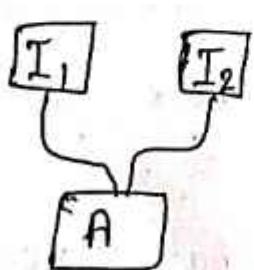
{

A a = new A();

a.show1();

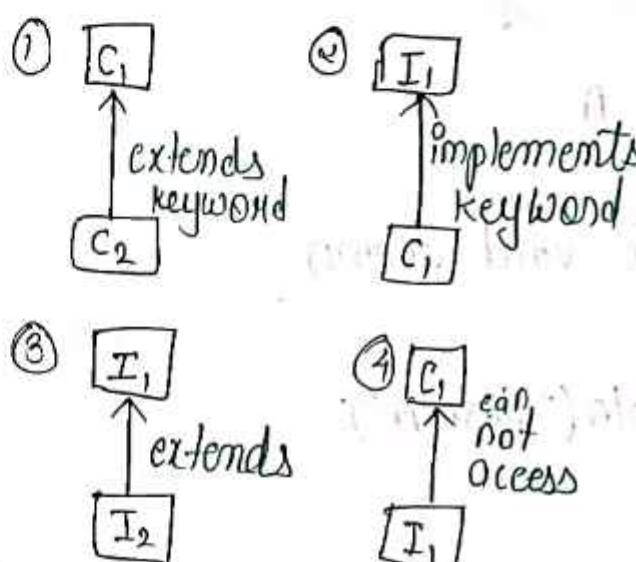
}

error:



If Java में Interface के case में multiple inheritance possible है लेकिन class के case में possible नहीं है।

If class का data interface में access नहीं हो सकता।



Example :

interface Int1

{

void show1();

}

interface Int2 extends Int1

{

void show2();

}

class A implements Int2

{

```

Type-2 interface Inter1
{
 void show1();
}

interface Inter2
{
 void show2();
}

interface Inter3 extends
 Inter1, Inter2
{
 void show3();
}

class A implements Inter3
{
 public void show1()
 {
 System.out.println("Show1");
 }

 public void show2()
 {
 System.out.println("Show2");
 }

 public void show3()
 {
 System.out.println("Show3");
 }
}

```

A a = new A();  
a.show1();  
a.show2();

O/p → show1  
show2

# extends keyword की help से more than one Interface को extends करना possible है। इस तरह से multiple inheritance possible है।

# with the help of interface we can achieve multiple interface.

```

class demo
{
 A a = new A();
 a.show1();
}

```

```
a.show1();
a.show3();
```

}

O/P → show1  
show2  
show3

Example :

```
interface Inter1
```

{  
 int x = 10;

}

class A implements Inter1

{  
 void show()  
 {  
 System.out.println("x = " + x);  
 }

}

class demo

{  
 A a = new A();  
 a.show();  
}

O/P → x = 10

Example :

```
interface Inter1
```

{  
 int x = 10;  
}

}

class A implements Inter1

{  
 void show()  
 {  
 x = 20;

System.out.println("x = " + x);  
 }

}

class demo

{  
 A a = new A();  
 a.show();  
}

Error: can not assign a value  
to final variable x.

Note : इतने से variable  
interface के अंदर  
होते हैं। Java compiler  
by default इनमें आगे नहीं  
key word लिख देता है।  
① public  
② static

③ final

means, a variable inside interface is final. and for it will be constant.

# for accessing the variable of interface is we use the interface's class name.

O/p  $\Rightarrow x = 80$

$x = 80$

$x = 10$

# interface को case ii 8A .class file के बनती है।

# we can not make constructor in interface

interface Intef1

{

int x = 10;

}

class A implements Intef1

{

int x = 20;

void show()

{

int x = 30;

System.out.println("x = " + x);

System.out.println("x = " + this.x);

System.out.println("x = " + Intef1.x);

// System.out.println("x = " + Super.x);

}

}

Example of ambiguity in interface

interface Intef1

{

int x = 10;

}

interface Intef2

{

int x = 20;

}

class A implements Intef1, Intef2

{

void show()

{

System.out.println("x = " + x);

}

}

```
class demo
```

```
{
```

```
A a = new A();
```

```
a.show();
```

```
}
```

error: reference to x is  
ambiguous.

```
S.o.p("x = "+x);
```

Example:

```
interface Inter1
```

```
{
```

```
public void show();
```

```
}
```

```
interface Inter2
```

```
{
```

```
public void show();
```

```
}
```

class A implements Inter1,  
Inter2

```
{
```

```
public void show()
```

```
{
```

"class A"

```
S.o.println("class A");
```

```
}
```

class demo

```
{
```

```
A a = new A();
```

```
a.show();
```

```
}
```

O/p → class A

Example of method hiding

```
class A
```

```
{
```

```
static void show()
```

```
{
```

class B

```
{
```

```
static void show()
```

```
{
```

```
S.o.println("class A");
```

```
}
```

class B extends A

```
{
```

```
* static void show()
```

```
{
```

```
S.o.println("class B");
```

```
}
```

```
}
```

class demo

```
{
```

```
B b = new B();
```

```
b.show();
```

```
}
```

O/p  $\Rightarrow$  class B  
# new A(); it is called object.

# Super class का reference variable subclass के object को hold कर सकता है। like int can hold byte data.

Example : prove

class A

{  
A()  
}

S.o.println("class A");  
}

class demo{  
}

new A();  
}

O/p  $\Rightarrow$  class A

A    x = new A();  
↓              ↓  
reference      object

int x = 10;  
↓      ↓  
variable      value

Example : of method overriding

class A  
{

void show()  
{

S.o.println("class A");  
}

}

class B extends A

{

void show()  
{

S.o.println("class B");  
}

}

class demo  
{

A a = new B();

a.show();  
}

O/p  $\Rightarrow$  class B

Example : method hiding

```
class A
{
 static void show()
 {
 System.out.println("class A");
 }
}
```

```
class B {
 static void show()
 {
 System.out.println("class B");
 }
}
```

class demo

```
A a = new B();
a.show();
}%> class A
```

# method hiding जैसे reference variable को corresponding working होती है।

# method overriding जैसे object को corresponding working होती है।

# why method overriding is called as runtime polymorphism?

ans) object is made at runtime that's why method overriding is known as RTP.

# reference variable is used for working corresponding to address

# we can not create constructor inside interface. bcz all variable inside the interface are static final and public and we are using constructor to initialised the instance variable in interface

instance variable does not exist. so we can not use constructor inside interface.

## Method overriding

- ① It is a Runtime polymorphism.
- ② method overriding के object working होती है।
- ③ object के corresponding runtime पर binding होती है that's why it is called as runtime polymorphism.

## Abstract class

Abstract class can have abstract and non-abstract methods.

2) Abstract class does not support multiple inheritance.

3) abstract class can have final, non-final and non-static variables.

4) The abstract keyword is used to declare abstract class.

## Method hiding

It is a compile time polymorphism.

method hiding में reference variable के corresponding working होती है।

comp reference variable के corresponding compile पर ही binding हो जाती है। that's why it is known as compile time polymorphism.

## interface

Interface can ~~only~~ have only abstract methods.

It supports multiple inheritance.

It can have only static and final variables.

The interface keyword is used to declare interface.

⑤ An abstract class can be extended using extends keyword.

An interface can be implemented using implements keyword.

⑥ A abstract class can have class members like private, protected etc.

Members of interface are public by default.

20/1/22

Example :

class B extends A  
{ }

class A  
{ }

void show()  
{  
S.o.println("class A");  
}

class demo  
{ }

B b = new B();  
b.show();

}

Op → class A

Example :

class A extends A

{ }

void show()

{ }

S.o.println("class A");

{ }

class demo

{ }

A a = new A();

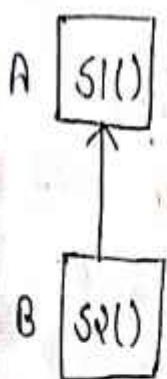
a.show();

{ }

→ error of cyclic inheritance involving A.

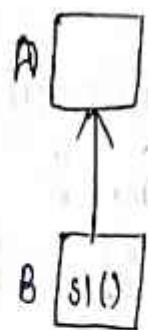
class A extends A

case



or

case 2:



A a = new B();  
 a.show1();  
~~O/p~~ → class B

A a = new A();  
 a.show1();

~~O/p~~ → class B error

case 3:

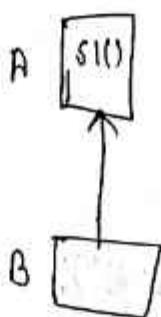


A a = new B();  
 a.show1();  
 a.show2();

~~O/p~~ → class A error.

~~class B~~

case 4:



A a = new B();  
 a.show1();

~~O/p~~ → class A → ~~O/p~~ → class A

~~error~~

Example :-

$\Rightarrow$  class A

class .

# Java by default supports runtime binding.

Note :- java में compile time पर checking reference variable के corresponding होती हैं और runtime पर object के corresponding होती हैं।

Example of case 2 :-

class A

{

}

class B extends A

{

void show()

{

System.out.println("class B");

{

}

class demo

{

A a1 = new B();

a1.show();

{

error :- can not find symbol

a1.show();

Example of case 3 :-

class A

{

}

void show1()

{

System.out.println("class A");

{

}

class B extends A

{

void show2()

{

System.out.println("class B");

{

class demo

{

A a1 = new B();

a1.show1();

a1.show2();

{

error :- can not find symbol

a1.show2();

Example of showcase 4:

```
class A
{
 void show1()
 {
 System.out.println("class A");
 }
}

class B
{
}

class demo
{
 A a1 = new B();
 a1.show1();
}

o/p ⇒ class A
```

Example :

```
interface Intef
{
 void show1()
 {
 System.out.println("class A");
 }
}

class A implements Intef
{
}
```

error :

According to 1.8 version

Example : ~~as~~

```
interface Intef
{
 default void show1()
 {
 System.out.println("class A");
 }
}

class A implements Intef
{
}

class demo
{
 A a1 = new A();
 a1.show1();
}

o/p ⇒ class A
```

version 1.8 ने हमें यह facility provide की है कि हम interface में default method लगा कर steady तरह छानते हैं। default means body वाली method and default लिखना compulsory है।

Example :

interface Intent

{

default void show()

{

S.o.println("class n");

{

{

class A implements Intent

{

void show()

{

S.o.println("class A 123");

{

{

class demo

{

A a1 = new A();

a1.show();

{

error: weaker access  
privilege.

Q → can we override default  
method ?

Yes.

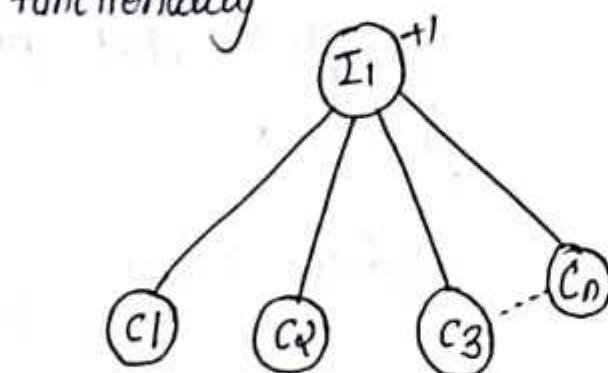
The default is a method  
name in interface,  
but by default method  
is public we declared  
method as default but  
by default method is  
public. That's why  
error msg is displayed.

public void show()

If default methods are also  
known as defender meth-  
ods or virtual extension  
methods.

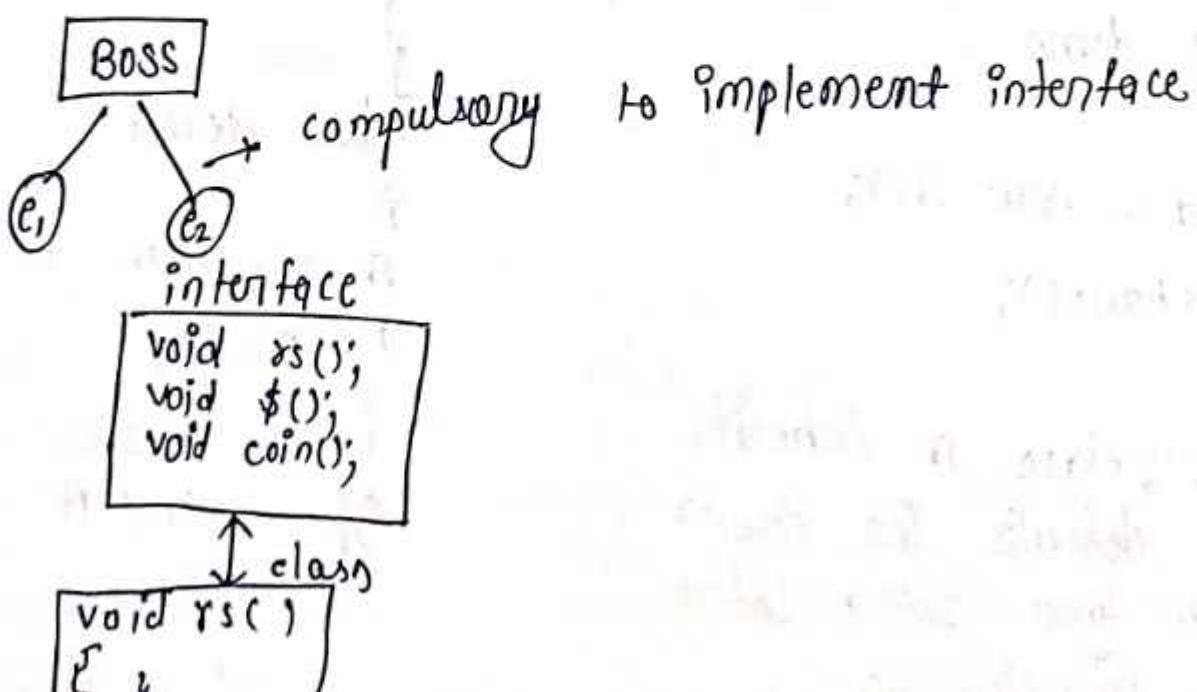
Advantage of default method in Interface :-

without effecting implementing class we can add new features in interface.



The main advantage of default method is without effecting implementation classes we can add new functionality to the interface.

Use of interface → 3rd party vendor के forcefully work करने के लिए हम interface का use करते हैं।  
Let's consider, हमें एक software बनाना है।



Example :

```
interface Inter1
{
 default void show()
 {
 System.out.println("Inter1 method");
 }
}
```

```
interface Inter2
{
 default void show()
 {
 System.out.println("Inter2 method");
 }
}
```

class A implements Inter1,  
Inter2

```
{}
}

class demo
{
 A a = new A();
 a.show();
}
```

Output: class A inherits  
defaults for show()  
from types Inter1, Inter2  
class A implements --

Example :

```
interface Inter1
{
 default void show()
 {
 System.out.println("Inter1 method");
 }
}
```

```
interface Inter2
{
 default void show()
 {
 System.out.println("Inter2 method");
 }
}
```

class A implements Inter1,  
Inter2

```
{}
public void show()
{
 System.out.println("class A");
}
```

```
class demo
{
 A a = new A();
 a.show();
}
```

O/p ⇒ class A

# for access the method of interface we use  
interface name.super.method name.

Example :

interface Inter1

```
{
 default void show();
}
```

```
{
 S.o.println("Inter1");
}
```

```
{
 interface Inter2
}
```

```
{
 default void show();
}
```

```
{
 S.o.println("Inter2");
}
```

class A implements Inter1,

```
{
 public void show() {
 Inter1.super.show();
 }
}
```

```
 Inter1.super.show();
}
```

```
 Inter2.super.show();
}
```

```
 S.o.println("class A");
}
```

```
{
 O/p : Inter1
 Inter2, class A
```

If we can also make static method in Interface

Example :

interface Inter1

```
{
 static void show();
}
```

```
{
 S.o.println("Inter1");
}
```

```
{
 class demo
}
```

```
{
 Inter1.show();
}
```

```
{
 O/p : Inter1
```

Example : program without class

interface Demo

```
{
 public void m() {
 S.o.println("m");
 }
}
```

```
{
 S.o.println("ram");
}
```

```
{
 O/p : ram
```

We can also run a program without class in Java with the help of interface.

Example :

interface demo

```
{
 static void main()
 {
 System.out.println("ram");
 System.out.println("ram");
 }
}
```

O/p → ram  
 ram

# by default method inside Interface is public.

Example :

interface Inter1

```
{
 static void show()
 {
 System.out.println("Inter1 method");
 }
}
```

class A implements Inter1

{ }

class demo

```
{
 A a = new A();
 a.show();
}
```

error: can not find symbol  
 a.show();  
 ^

Example :

Interface Inter1

```
{
 static void show()
 {
 System.out.println("Inter1");
 }
}
```

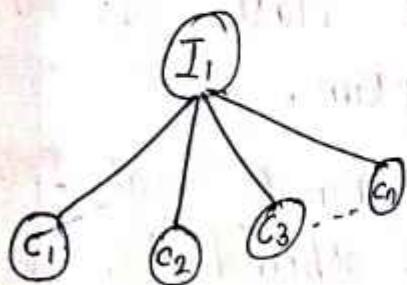
class A implements Inter1

```
{
 void show()
 {
 Inter1.show();
 System.out.println("class A");
 }
}
```

O/p → Inter1  
 class A

According to jdk 1.9 version we can also make private method in interface.

why? Advantage of private method in interface.



interface में 10 default method हो और वे सभी में same data हो वह data का private method में रख सकते हैं। means, now we can put common data in private method. उस data किसी ने show नहीं होगा। means, implementation details show नहीं होगी। ~~and~~ and यह method में उसे उस data की need हो तो उसमें उसे access कर सकते हैं। The introduction of private method avoids the code duplication.

**Advantages**

- Avoiding code duplication
- Ensuring code reusability
- Improving code readability.

Q + want to store data in file.

# we can not pass string in write method.

Example:

```
import java.io.*;
class Demo
{
 public void m() throws Exception
 {
 String s1 = "zoom";
 int i = 0;
 FileOutputStream fi = new
 Fops("aaa.txt");
 while (i < s1.length())
 {
 fi.write(s1.charAt(i));
 i++;
 }
 }
}
```

Example : with old data → You want to add new data in the file. Do FileOutputSteam fi = new FOpS("aaa.txt" true); pass the para true parameter after file name.

default method :-

interface default method are by default available

to all implementation classes.

An implementation

class can use these default method directly or can override. when do override when default methods in implementation classes you omit the default keyword. Example :

# If there is only one public class in a program.

# If class name and file name should be same.

# file का पुरा data read करने पर ending point -1 होता है।

# file का हुआ data read करने के लिए read method provide की है।

# functional interface :- इस interface class में single abstract method हो। इस interface को functional interface कहते हैं।

→ you can not pass string in write method, only single char can be pass.

int i;  
FileInputStream fi = new FIS  
("Demo20.java")

i = fi.read();  
while (i != -1)

i = fi.read();  
System.out.println((char)i);

3 3

24/01/22

## Applet

कुछने के लिए java ने हमें

applet का tool provide

किया है।

① static websites are of two types

① static website :- ऐसी web-site पर जिसमें user

interact नहीं कर सकता। वो # applet बनाने के लिए

static website कहलाती है। class को applet से

② dynamic website :- ऐसी web-site पर जिसमें # there is no main method

user interact कर सकता है। in applet.

वो dynamic websites कहलाती # steps of making applet

है। like facebook, college

portal etc.

① .java file

② .class file

③ .html page बना कर उसमें  
java का code merge करना  
है।

④ program को compile  
and then appletviewer and  
html file का name  
use करना है।

# static means कि एक user

के corresponding name की

view होता है।

# java की html is a

dynamic language and

html is a static

language. So if html की

dynamic बनाने के लिए

java को html में merge

new page

Step 1:

```

import java.applet.*;
import java.awt.*;
public class AppDemo extends
Applet
{
 public void paint(Graphics g)
 {
 g.drawString("nam", 100, 100);
 }
}

```

```

<html>
<body>
<applet code = "AppDemo.
class" width = "200" height
= "200">
</applet>
</body>
<html>

```

command → javac AppDemo.java  
appletviewer abc.html

# applet के program को  
मग उन्हें कैसे work या  
तो browser में install  
plugin करता है या किसी  
applet viewer.

# applet viewer का use testing  
के purpose के किया जाता -  
है।

# applet में भी self body को  
जैसे ही उसे close करना  
compulsory है।

# Applet viewer का use तभी  
file में उन्होंने है, जो प्रसंगे  
html का use code लिखता है।

```

import java.applet.*;
import java.awt.*;
public class AppDemo2
extends Applet
{
 /* <applet code = "AppDemo2.
 class" width = "....." height = ".....">

```

```
> <applet>
* /
; P v paint(G g)
; {
; g.drawString("ram", 100, 100);
; setBackground(Color.red);
; }
; Appletviewer AppDemo2.java
;
Initialization के purpose
; ये applet ने ही init
method provide की है। It
works like constructor.
Example :
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
P c AppDemo2 extends
Applet implements AL
}
/*
TextField tx1;
Button b1;
public void init()
{
tx1 = new TF(10);
add(tx1);
b1 = new Button("click");
add(b1);
b1.addActionListener(this);
}
public void actionPerformed
(ActionEvent e)
{
tx1.setText("Softwaves");
}
P v paint(G g)
}
```

## Lifecyle of applet

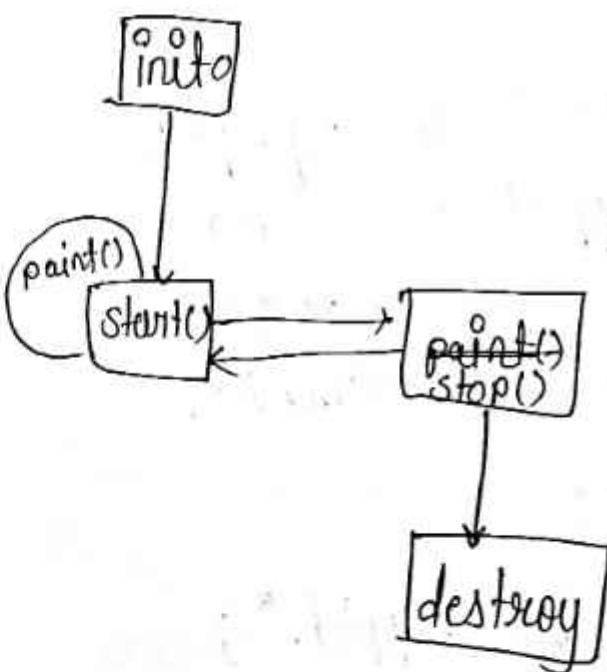
Lifecycle of applet has  
4 methods :

- ① public void init()
- ② public void start()
- ③ public void stop()
- ④ public void destroy()

Lifecycle of applet is  
basically used for anim-  
ation.

# जैसे ही applet load होता  
है तो से ही init method  
call हो जाती है। init  
method के just बाद  
start call होती है।  
जिसे भव अप्पलेट में  
visible होती है तो  
ही start call हो जाती  
है भव - भव start  
method call होगी उसके  
साथ ही paint method  
call होगी। भव हम

applet को minimize करना  
है। stop method call  
द्वारा cmd वरे maximize  
होते हैं तो start method  
call होगी जब एक applet  
को close करते हैं तो  
stop method call होती  
है तभी start method  
destroy call होती है। और applet  
destroy हो जाती है।



```
public void init()
{
 S.o.println("init");
}

public void start()
{
 S.o.println("start");
}

public void stop()
{
 S.o.println("stop");
}

public void destroy()
{
 S.o.println("destroy");
}

protected void paint(Graphics g)
{
 S.o.println("paint");
}
```

## Q) Matrix Multiplication

```
int i, j, k;
int x[2][2] = {{2, 3, 4, 5},
 {6, 7, 8, 9}};
int y[2][2] = {{1, 2, 3, 4},
 {5, 6, 7, 8}};
int c[2][2] = new int [2][2];

for (i=0; i<2; i++)
{
 for (j=0; j<2; j++)
 {
 c[i][j] = 0;
 for (k=0; k<2; k++)
 {
 c[i][j] += x[i][k] * y[k][j];
 }
 cout << c[i][j];
 }
}
```

5.0.0();

}

Output:  
27 40  
35 52.

# Difference b/w for and for - each loop.

: for loop

for - each loop  
(enhanced loop)

- ① It is one of the original ways of iterating over an array.  
It is a newer way with lesser code to iterate over an array.
- ② It is faster in performance.  
It is slower than the traditional loop in performance.
- ③ The break statement can be used to come out from the loop.  
The break statement can not be used bcz of the callback function.
- ④ The parameters are the iterator, counter and incrementor.  
The parameters are the iterator, index of item, and array to iterate.
- ⑤ It works with the await keyword.  
The await keyword cannot be used due to the callback function.

F await/async keyword :— The await keyword means to wait until this value (an error) has been finalized. (in java script)

java does not have any await keyword.

for loop is introduced in java 1.

# If a null value is passed to a switch statement, it results in a ~~ANALYSIS~~ NullPointerException.

eg → String s = null;  
switch (s)  
{  
 case "null":  
 System.out.println("room");  
 default:  
 System.out.println("default");  
}

eg → String s = "null";  
switch (s)  
{  
 case "null":  
 System.out.println("room");  
 default:  
 System.out.println("default");  
}

Exception in thread  
"main" java.lang.  
NullPointerException.

eg → String s = 1;  
error: incompatible  
Types : int can not be  
converted to String

# default value of classes and object  
is null.

Do Movie You can win If you want 😊

