||Shreell

Java.

- JRE - java runtime environment
- JVM :- Java virtual machine

} JDK

First.Java → (First.class) call → Java first

(byte code)

JVM

↑ File

↑ compiled file

* CMD

- To create file md - main directory
- To enter file cd
- To open directory - dir
- For complie use javac
- For Run type (Java File name) → Java myfirst.

Skeleton of Java program

First.java

```
import java.lang.*;
Class First
{
    Public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```

either use ~~arg~~ or ~~args~~

Reading from keyboard

```
import java.lang.*;
import java.util.*;
```
} Not imp to write

```
class Sum
{
  public static void main(String[arg[]])
  {
    Scanner S = new Scanner(System.in);
    int a,b,c;
    System.out.println("Enter 2 No");
    a = S.nextInt();
    b = S.nextInt();
    c = a+b;
    System.out.println("sum is -" +c);
```

* Class Scanner -- { Data type }.

- nextInt() → 1,2
- nextFloat() → 1.32
- nextDouble()
- next()
- nextline()
- nextByte()
- nextShort()
- nextlong()
- nextboolean()

*To conform data type

hasNextInt()
hasNext Float()
} True/false

String x = sc.nextLine();

* Program for welcoming a person

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

class Name
{
  Public static void main (String arg[])
  {
    String name;
    Scanner S= new Scanner(System.in);
    System.out.println("We May I know your Name");
    S name = S.nextLine();
    System.out.println("Welcome"+name);
    }
}
```

★ Chapter 2 - Data types

* Primitive- Data types

- Integral
  - byte
  - Short
  - int
  - long
- Floating Point
  - Float
  - double.
- Char
- boolean

- enum is not a primitive data type

Type	Size	Range	Default
byte	1 byte	-128 to 127	0
Short	2	-32768 to 32767	0
int	4	-2147483648 to 2147483647	0
long	8	- -	0
float	4	± 1.4E -45 to ± 3.4E +38	0.0f
double	8	± 439E -324 to ± 1.7E+ 308	0.0d
Char	2	0 to 65535	\40000
boolean	?	true/ false	false.

Integral

Floating

1 byte = 8 bit

→ Double value can't be assign to float data type

* Code for checking Range of Datatypes

```
import java.lang.*;
class DataRangeSize
{
    Public static void main (Strings arg[]) {
    {
        S.O.Pln ("Int Min-" + Integer.MIN_VALUE);
        S.O.Pln ("Int Max-" + Integer.MAX_VALUE);
        S.O.Pln ("Int Bytes-" + Integer.BYTES);
    }
}
```

-- We can also check for Bytes, Float, Boolean etc using same syntax.

variables (a, b, c)

- It is used to store data
- It must have data type

* Rules

1. case Sensitive
2. Contains Alphabets, Numbers, _ or $ → no special char but this allowed
3. Starts with Alphabet, _ or $
4. Should not be Keyword ex:- int, if ...
5. Should not be a class name, if class is also in
6. No limit on length of name
7. follow camel cases

ex:- rollNumber ✓
averageMarksOfClass;

literal

→ The value of variable known as literal

→ it is also has it's data type

→ Types of literals

1) byte, short, int ⇒ int literal

2) long → L or l

3) float → F or f

4) double → D or d

5) char →

6) boolean → true or false

for byte b = 10;
byte b = 0b10 binary
byte b = 0 octal
byte b = 0xA hexa
decimal

Operators

A. Arithmetic operator

- +
- −
- *
- /

} done by all data type except boolean ⇒ int + int, float + float, double, char, boolean ×

- % → mod operator → gives remainder.

$x = 12.7 \% 3.1$ → 3.1) 12.7 (4
12.4
0.3

mod operator can also float & double

* Expression

- *, /, % → have more precidence → ie priority
- +, − → have low precidence compared

* if we want to change precidence / priority then use ( ) → paranthesis

ex. $\frac{a+b}{2}$ → $x = (a+b)/2$

- $\frac{a+b}{2c}$ → $x = (a+b)/(2*c)$
- $a^2 - b^2$ → $a*a - b*b$

---

operator with different datatypes

- byte b=10;
- short s=15;
- int i=7;
- long l=50L;
- float f=12.5f;
- double d=17.5d;
- char c=65;

if byte+byte
short + short
byte + short
int + int
short + int
↓
main data type will (int)

* The process of changing datatype while performing operations known as <u>coercion</u>

- x = b+s → int
- x = s+i → int
- x = i+f → float.
- x = l+f → float

x = c+s; → int
x = c+i; → int
x = f+d; → double.
x = l+d. → double.

<u>Common programm</u>

1) <u>Area of triangle</u> → $\frac{1}{2} \times b \times h$.

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

class Expression
{
  Public static void main (String arg[])
  {
    int height, width;
    float area;
    System.out.println(" Enter the value of height width ");
    Scanner sc = new scanner (System.in);
    height = sc.nextInt();
    width = sc.nextInt();
    area = (width * height * 0.5);
    System.out.println(" Area of triangle :- " + area);
  }
}
```

2. Area of triangle → $S = \frac{(a+b+c)}{2}$

$Area = \sqrt{(S*(S-a)(S-b)(S-c)}$

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

Class SidesOfTriangle
{
Public static void main(String arg[])
{
    float a, b, c, s;
    double Area;
System.out.println("Enter the value of side A -");
Scanner sc=new scanner (System.in);
a = sc.nextFloat();
System.out.println("Enter the value of side B -");
Scanner sb=new Scanner (System.in);
b = Sb.nextFloat();

System.out.Print ln("Enter the value of side c -");
Scanner sc = new Scanners (System.in);
c = Sc.nextFloat();

S = (a+b+c)/2;
area = Math.Sqrt(S*(S-a)*(S-b)*(S-c));
System.out.Print ln("Area of Triangle is- " + area);
}
}
```

* Program for Roots of Quadratic equn

$$r1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$r2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

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

class QuadraticEqun
{
public static void main(String arg[])
{ int a,b,c;
  float r1,r2;
System.out.println("Enter the value of a,b,c from
                    Quadratic equn");

Scanner sc=new Scanner(System.in);
  a=sc.nextInt();
  b=sc.nextInt();
  c=sc.nextInt();

r1=(-b+Math.sqrt((b*b)-(4*a*c)))/2*a;
r2=(-b-Math.sqrt((b*b)-(4*a*c)))/2*a;
System.out.println("Roots of QE are R1="+r1);
System.out.println("Roots of QE are R2="+r2);
}
}.
```

---

* Program for Area & volume of cuboid

```
import java.lang.*;
!

{ int
  float
  --
  --
Area= 2*(---)
vol ____ )
```

Area = (length × height) + (breadth × hight) + (length × breadth)

volume = length × breadth × height

Operators & Expression

* Increment (++) & decrement (--) → only on boolean we cannot perform increment or decrement

- post ++, post --
- ++ pre, -- pre.

imp

ex int x = 5;
x++; = 6
++x; = 6

ex int x = 5, y;
y = x++;
y = 5, x = 6

ex int x = 5, y;
y = ++x;
x = 6 y = 6

ex float x = 5.3;
x++;
x = 6.3

ex byte x = 4;
x++;
x = 5

ex char x = "A";
65
x++;
so.p(x);
B 66

Bitwise operator

- AND ∧ & &
- OR ∨ | |
- NOT ¬ ~
- XOR ⊕ ^
- Right shift >>
- left shift <<
- Unsigned Right shift >>>

&

A	B	&
0	0	0
0	1	0
1	0	0
1	1	1

∧

A	B	∧
0	0	0
0	1	1
1	0	1
1	1	0

|

A	B	|
0	0	0
0	1	1
1	0	1
1	1	1

* left shift <<

int x = 10;

x → 00001010 extra 0
00010100
16 8 4 2 1
= 20

>> -- ×2
>>> -- ×4 ie $(n \times 2^k)$

* Right Shift >>

int x = 10;

x → 00001010
00001010

5
<< -- ÷2
<<< -- ÷4
i.e $\left(\frac{n}{2^k}\right)$

Merging & Masking

* program for swaping two number

```
import java.long.*;
import java.util.*;
class swap;
{
  public static void main(String arg[])
  {
    int a,b;
    a= 5;
    b= 10;
    a= a^b;
    b= a^b;
    a= a^b;
    System.out.println(a+"_"+b);
  }
}
```

Merging → OR (|)

Masking → AND (&)

* Merging — b inside a

a = (0) → 0000 0000

b = (5) → 0000 0101

a=a|b → 0000 0101

ex ② → a= (5) → 0000 0101 ✓

b= (9) → 0000 1001

b= b<<4 — b = (9) → 1001 0000 ✓

a= a|b — 1001 0101

Merging.

* Code for Merging of 2 No.

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

class Merging
{
    public static void main (String arg[])
    {
        byte a, b, c;
        a= 10
        b= 5
        c= (byte) (a<<4);
        c= (byte) (a|b);
        System.out.println (c);
    }
}
```

Widening & Narrowing

Widening → upcasting

Narrowing → downcasting

widening :- Storing the data of a smaller data ~~size~~ type into a larger size data type.

```
byte b=10;
short s;
int i;

s=b;   // ∴ s=10
i=b;   // ∴ i=10
```

-- in narrowing we loose data.

* **Narrowing** :- when widening is not possible it or due to size compatability at that time we have to typecast it.

Ex :-

```
byte b=5
short s=15;        b=(byte) s;

int i=10;          f=i;
float f=12.1;      i=~~int~~ (int) f;
```

← in this 0.1 will

* **Destination** →

Source ↓	byte	short	int	long	float	double	char	boolean
byte.	✓	✓	✓	✓	✓	✓	✗	✗
short	✗	✓	✓	✓	✓	✓	✗	?
int	✗	✗	✓	✓	✓	✓	✓	✗
long	✗	✗	✗	✓	✓	✓	✗	✗
float	✗	✗	✗	✗	✓	✓	✗	✗
double	✗	✗	✗	✗	✗	✓	✗	✗
char	✗	✗	✓	✓	✓	✓	✓	✗
boolean	✗	✗	✗	✗	✗	✗	✗	✓

OR

byte → short → int → long → float → double
char ↗

Format specifier

char → c
int → d, o, x → decimal, octal, hexadecimal
float → f, e, g
string → s

argument index - %1$s, %2$f, %3$d

ex int x = 10;
float y = 0.0012f;
char z = 'A';
String str = "Java program";

(if "%20s", str)
↓ then it is width
ie value of str will
store in 20 place

S.o.pln("%3$s, %2$f, %1$d", x, y, str);

output:- Java program 0.0012 10

String.

- It is a collection of character that are forming words or sentences

String str1 = "Java program";

↑ Reference — string object / string literals

① String (char[]) :-

char c[] = { 'A', 'B', 'C', 'D' };
String str1 = new String (c);

② String (byte[])

byte b[] = { 65, 66, 67, 68 };
String str 2 = new string (b);

③ String (String)

String str1 = new String ("Java");

String Methods

ex String str = "Java"; → stored in pool

1. str.length → int
   int l = str.length(); ⇒ 4
2. str.toLowerCase(); → string
   str = str.toLowerCase(); ⇒ java → stored in heap
3. str.toUpperCase(); → string
   str = str.toUpperCase(); ⇒ JAVA
4. str.trim(); ex String str = " welcome "
   str = str.trim(); ⇒ welcome
   ↑ String

5. Str.substring(); → string

ex str = "welcome";
0 1 2 3 4 5 6.

string sub = str.substring(3) → come

string sub = str.substring(3, 6) → com

6. Str.replace(); → string

str = str.replace('e', 'k');
→ wklcomlc

7. Str.StartsWith(); → boolean.

ex :- string str = "www.abcd.org";

ex str = str.Starts with ("www");

8. Str.endswith(); → boolean

ex → str = str.endsWith(".org");

9. Str.CharAt(int index); → Char

→ verify the char at given index.

10. int indexOf(String S) → Int.

→ str.indexOf(".", 4);
it will give index of "." after 4

→ If it will not found it write -1 false

11. int lastIndexOf(String S); → Int

str.lastIndexOf(".") → it will start finding from last
or
(".", 7);

12. Str.equals(String S) → boolean

String str1 = "JAVA";
String str2 = "java";
String str3 = "Python";
String str4 = "python";

→ str3.equals(str4) → True.

str1.equals(str2) → False.

or if (str1 == str2) → true.

13. equalsIgnoreCase(String s) → boolean

→ str1.equalsIgnoreCase(str2); → True

↓

It ignore cases ie upper case & lower case

14. compareTo(String s) → int

→ str2.compareTo(str3);

→ It will check who will come first in dictonary

→ java is smaller than pyton → -1

→ str3.compareTo(str2); → +1

→ If it is greater i.e come next in dictionary then it will write +1

otherwise -1

* Note for ==

str1 → java ← str2

It is in pool

string = new.string → in heap

↓

str3 → java.

str3.equals(str1) → True

if (str3 == str1);

→ false

str1.equals(str2) → True

if (str1 == str2) → True.

Regular Expression

1) . → It must be any character but it should be single character

2) [abc] → exactly given letters → but only one

3) [abc] [vz] → Either first or second set

4) [^ abc] → Except abc.

5) [a-z1-7] → a-z or 1-7 → defing range

6) A|B → A or → str.matches(" ");

7) xz → exactly xz

* Meta characters

1. \d → Digits
2. \D → not digit
3. \s → Space
4. \S → not Space
5. \w → Alphabet or digit
6. \W → Neither alphabet or digit

* Quantifiers

1) * → 0 or more

[abc]* → abcccaaab

2.) + → one or more

3) ? → 0 ~~or~~ or 1 time

4) {x} → x times

ex [a-z]{5} → abcde

5. {x, y} → Between x & y times

Conditional Statement

Relational operator:

1) < 2) <= 3. > 4. >=

5) == 6) != → not equal to.

ex: int a=5, b=10, c=15;

a<b or a<=b → True

c>a or c>=a → True

a==b → False

a!=b → True

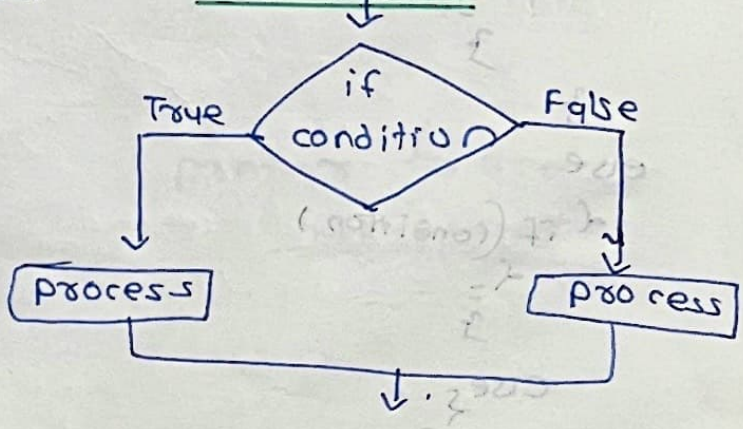
* logical operator → used to combine two conditional statement

- &&
- ||
- !

A	B	A && B	A || B	!A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

* conditional statement :→ ① If-else.

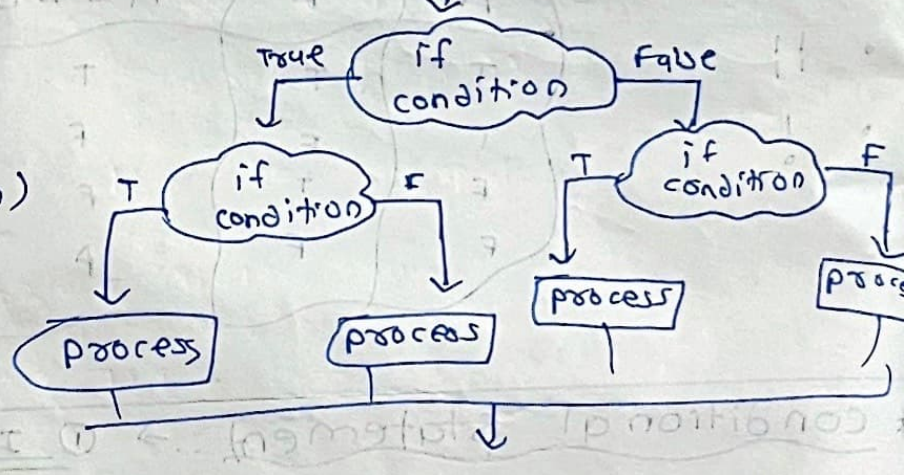
```
if (condition)
{
  =
}
else
{
  =
}
```



ex:-

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

class Number
{
  public static void main (String arg[])
  {
    int a=5
    if (a>=0)
    {
      System.out.println("The no is +ve");
    }
    else
    {
      System.out.println("The no is -ve");
    }
  }
}
```

2) Nested if




```
if (condition)
{
  if (condition)
  {
    =
  }
  else
  {
    =
  }
}
else
{ if (condition)
  {
    =
  }
  else
  {
    =
  }
}
```

Code for max of 3 NO:

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

class MaxNO.
{
  public static void main(String arg[])
  {
    int a=5, b=6, c=10;
    if ( a>b && a>c)
    {
      System.out.println(" a is max NO");
    }
    else
    {
      if (b>c)
      {
        System.out.println(" b is max NO");
      }
      else
      {
        System.out.println(" c is max NO");
      }
    }
  }
}.
```

3) else if ladder / Nested if

```
if (condition)
{
  =
}
else if (condition)
{
  =
}
else if (condition)
{
  =
}
else { = }
```

prefer to use.
Nested If

* code to find type of website & the protocol used

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

class website
  {
    public static void main (String arg[])
    {
      System.out.println(" Enter the url of website
      Scanner sc=new Scanner(System.in);
        String url;
        url= sc.nextLine();
      String protocol;
      protocol=url.substring(0, url.indexOf(":"
      if ( protocol.equals (" http"))
        {
          S.O.P (" Hypertext transfer protoco");
        }
      else if ( protocol.equals (" ftp"))
        {
          S.O.P ("ft file transfer protocol");
        }

      String ext;
      ext= url.substring (url.lastIndexOf(".")+1
        if (ext.equals ("com"))
          {S.O.P (" It is commercial website");
          }
        else if (ext.equals ("org"))
          {S.O.P (" It is organizational website");
          }
  }
```

Switch case.

ex:- To print day using no of Day.

```
{
  P.S.V.m
  {
    int day;
    Scanner - -
    - -
    -

    Switch (day)
    {
      case 1 : S.O.P ("Mon");
               break;
      case 2 : S.O.P.("Tue");
               break;
               |
               |
      case 7 ; S.O.P.("Sun");
               break;
      default : S.O.P ("Invalid Day");
               break;
    }
  }
}
```

---

* It Supports:-

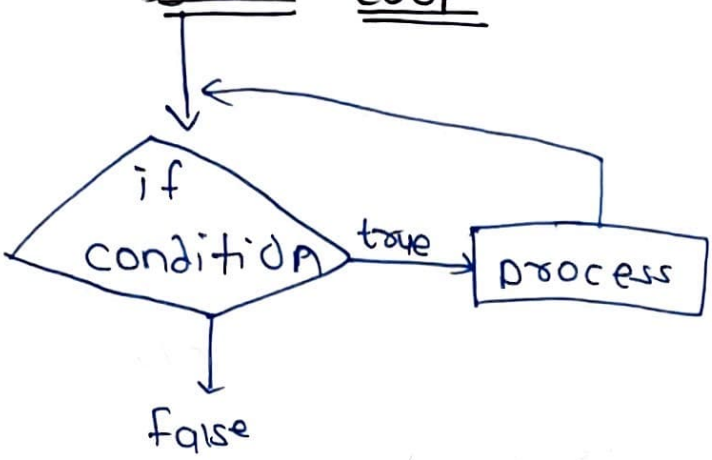
byte, short, int } case 1:

char } case 'a':

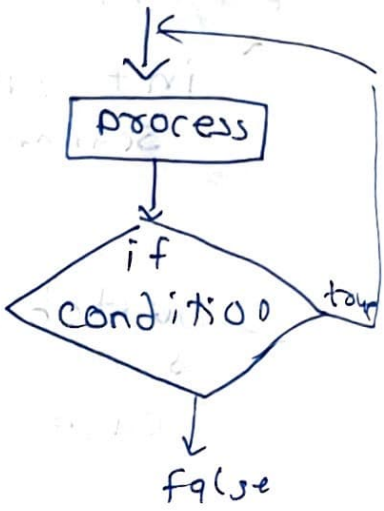
String } case "file":

★ Loops ★

While Loop



* Do-while loop



ex

```
int i=1, n=100;
while (i< n).
 {
 System.out.pln(i);
     i= i * 2;
 }
```

```
int i=1, n=100
 do
 {
   S.O.pln(i);
     i = i * 2;
 } while(i<n);
```

→ In this case both result will be the same but in such / very few cases result will be differe

ex

```
int i=100,;
 while (i<100)
 {
  S.O.pln( i);
     i= i*2;
 }
```

result :- —

```
int i = 100;
 do
 { S.O.p (i);
     i=i*2;
 } while (i<100);
```

result :- 100

* Infinite loop

①
```
int i;
while (true)
{
    S.O.Pln(i);
    i++;
}
```

②
```
byte i;
while (true)
{
    S.O.Pln(i);
    i++;
}
```

result :- from -128 to +127
but it will repeate in cyclic form

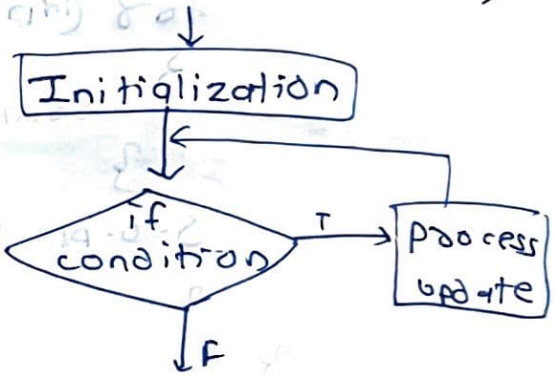
---

* For Loop

Statement :-
```
for (initialization; condition; updation)
{
  =
  =
}
```

counter ↓ ① ② ⑤ ④ ⑦ ③ ⑥

↑ done only once



ex
```
for (int i=1; i<=10; i++)
{
S.O.Pln(i);
}.
```

---

* codes

1) code for display multiplication table.
```
import java.lang.*;
import java.util.*;
class Mul
{
  public static ....
  {System.O.Pln("Enter the no");
   Scanner sc= new Scanner(System.in)
   int i = sc.nextInt();
```

```
for ( int n=1; n<=10; n++)
  {
    S.O.pln(" i+"*"+n+"="+(i*n));
  }
}
}
```

2) code to find sum of n numbers

```
public sta - -
  {
    S.O.pln (" Enter the no ");
    Scanner sc= new Scanner(System.in);
    int i = sc.nextInt();
    int sum = 0;
    for (int n=1; n<=i; n++)
    {
      sum = sum +n;
    }
    S.O.pln ("Sum of "+i+" Natural Numb" +sum);
  }
}
```

3) Factorial

```
class
  { p.s.v.m (S...)
    {
      S.O.pln(" Enter a Number ");
      Scanner sc= new Scanner (System.in);
      int i= sc.nextInt();
      long fact=1;
      for(int n=1; n<=i, n++)
        {
          fact= fact *n;
        }
      S.O.pln ( fact);
```

29602

4. Display Digit of Number

```
n = 2578
int r, n;
while (n>0)
{
r = n % 10.
Sy.o.p.ln (r);
n = n/10.;
}.
```

5 Count Digit on Number

```
class { p.s --
{
input n=;
~~int r;~~
while ( n > 0);
int count = 0;
{
~~r = n % 10~~;
n = n / 10;
count ++;
}
S.O.Pln ( count);
```

6. Amstrong No

```
int r;
int sum=0;
int m=n;
input n=;
while (n>0)
{
r = n % 10;
n = n / 10;
sum = sum + (r*r*r);
}
S.O.Pln( sum);
if ( sum = = m)
{
S.O.Pln (" It is amstrong No");
else }
{S.O.Pln (" It is not Amstrong No");
```

7. Reverse the number

```
~~int i=~~ input i=;
int n;
int rev=0;

while (i>0);
 {
    n = i % 10;
    i = i / 10;
    rev = (rev * 10) + (n);
 }
S.O.pln (rev);
}
}
```

8 Palindrome

```
input i = ?
int r;
string ~~str;~~ rev;
while ( i>0); ,int m=n;
 {
    r = i % 10;
    i = i / 10;
    rev= rev+r.
    }.
 if S.O.pln (rev);
 if (m==rev)
 {
    S.O.pln ("It is an palindrome")
 }
 else
 {
 S.O.pln("It is not an palindrome");
 }
}
}.
```

9. • Display a Number in words even with tail

```
class
p.s.v.m (String arg[])
{
    s.o.p.ln ("Enter the no");
    scanner sc = new scanner (System.in);
    int n = sc.nextInt ();
    int m = n;
    String rev = " ";
      int r;
    while (i>0)
    {
        r = n % 10;
        m = n / 10;
        rev = rev + r;
    }.
    char c;
    for (int i = rev.length () - 1; i>=0; i--)
    {
     c = rev.charAt (i);
       switch ('c')
       {
     case '0':- System.out.println ("zero");
      break;
     case '1':- s.o.p.ln ("one");
      break;
     case '2':- s.o.p.ln ("Two");
      break;
        |
        |
        |
        |
     case "9": s.o.p.ln ("Nine");
      break;
     default: s.o.p.ln ("Enter the valid No");
      break;
    }
  }
}
}
```

10. Display A.P series

```
class
{
    p.s.v.m(---)
    {
        S.o.p.ln ("Program for A.P");
        Scanner sc = new scanner (System.in);
        S.o.p.ln ("Enter the value of a, d, n);
        int a = sc.next Int();
        int d = sc.next Int();
        int n = sc.next Int();
        int term = a;
        for (int i=0; i<n; i++)
        {
        term = term + d;
        S.o.p.ln (term + ", ");
        }
    }
}
```

11. Display G.P series

```
class
{ p.s.v.m (---)
  { S.o.p.ln ("Program for G.P");
    Scanner sc = new Scanner (System.in);
    S.o.p.ln ("Enter the value of a, r, n);
    int a = sc.next Int();
    int r = sc.next Int();
    int n = sc.next Int();
    int term = a;
    for (int i=0, i<n; i++)
    {
      term = term * r;
    S.o.p.ln (term + ", ");
    }
  }
}
```

12. Fibonicass series

```
class
{ p.s.v.m (--)
  { S.o.pln ("program for G.P");
    Scanner sc = new Scanner (System.in);
    S.o.pln ("Enter the no of terms n=");
    int n = sc.nextInt();

    int a=0, b=1, c;
    S.o.pln (a+", "+ b +", ");

    for (int i=0; i<n-2; i++)
    {
      c=a+b
      S.o.pln (c+", ");
      a=b
      b=c;
    }
  }
```

Nested Loop

→ loop inside a loop known as nested loop

Ex ①

```
for (int i=0; i<5; i++)
{
  for (int j=0; j<5; j++)
  {
    =
    =
  }
}
```

Outer loop

inner loop

---

* Break

→ The break statement can also be used to jump out of loop

```
for (int i=0; i<10; i++){
  if (i==4){
    break;
  }
  S.o.p (i);
}
```

0, 1, 2, 3

* continue

→ It skips the mention value

```
for (int i=0; i<5; i++)
  if (i==4) {
    continue;
  }
  S.o.p (i);
```

0, 1, 2, 3, 5

Array.

→ It is a collection of similar data elements known as Array.

ex:. $X = [8, 3, 9, 5, 4, 6]$
$x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6$

Creating Array

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

Reference — object → created in Heap

A ⟶ [| | | |]
0 1 2 3 4

size = 5 × 4 = 20 byte

* property :- System.out.println (A.length);

initializing value in Array

```
int A[] = new int [5];
int A[] = { 2, 4, 6, 8, 10 }
```

← size of Array.

* For change

```
A[0] = 15;
```

* For printing

```
S.o.pln (A[1]);
S.o.pln (A[2]);
```

& much more

* For All value in Array

```
for (int i=0; i< A.length; i++)
{
    S.o.pln ( A[i] );
}
```

* For reverse order: (Traversing)

```
for (int i = A.length-1 ; i >=0 ; i--)
{
  S.o.pln (A[i]);
}
```

* For each loop

→ It is use to access all elements in Array.

```
for (int x : A)
{
  S.o.pln (x);
}
```

∴ It uses only in forward direction not in backward.

* Methods for initializing Array

- int A[] = new int[10]; → int A[] = {...}
- int B[] = {1, 2, 3, 4, 5};
- int C[];
  C = new int[10];

* Code for max element in Array.

```
int A[] = {5, 12, 8, 15, 16, 25, 120};
int max = A[0];
for (int i = 0; i < A.length; i++)
{
  if (A[i] > max)
  {
    max = A[i];
  }
}
System.out.println("Max element is " + A[i]);
}
}
```

* For 2nd largest element.

```
int A[] = { 12, 18, 25, 26, 102, 109 };

int max1 = A[0], max2 = A[0];

for ( int i = 0; i < A.length; i++)
{
    if( A[i] > max1)
    {
        max2 = max1;
        max1 = A[i];
    }
    else if ( A[i] > max2)
    {
        max2 = A[i];
    }
}
System.out.println ("2nd largest element is =" + max
}
}
```

* Rotating of Array

① ~~Right~~ left shift

```
int A[] = { 1, 3, 9, 10, 18, 40 };

int temp = A[0];

for (int x : A)
{
    s.o.p(x + ",");
}
s.o.p(" ");

for( int i = 1; i < A.length; i++)
{
    A[i+1] = A[i]
}

A[A.length - 1] = temp;

for (int x : A)
{ s.o.p (x+",");
```

② ~~Right shift~~

```
int A[] = { 1, 3, 9, 10, 18, 40 };
int temp = A[A.length - 1];
for (int x : A);
{
    s.o.p ( x+",");
}
s.o.p(" ");
int i = A.length-2; i>0; i--
for( int i = 0; i < A.length - 2; i++)
{
    A[i] = A[i-1];
}
A[0] = temp;
for (int x : A)
{
    s.o.p (x + ",");
}
```

③ Inserting element in array

```
int A[]= new int[10];
    A[0] = 1;
    A[1] = 2;
    A[2] = 3;
    A[3] = 4;
    A[4] = 5;
    A[5] = 6;

for (int x: A){
    S.op(x+" ,");
    }

for (int i= A.length-2; i>2 ; i--)
    {
      A[i] = A[i-1];
    }
    A[2] = 15;
for (int x: A)
{S.o.p(x+" ,");
    }.
```

④ Increasing size of Array

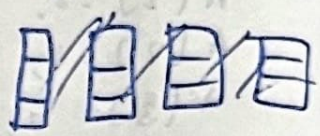
```
int A[]= { 1,2,3,4,5};
int B[]= new int[2* A.length];

for ( int i=0; i< A.length; i++){
    {
      B[i] = A[i];
    }
    A= B;
    B = null;
for (int x: A)
    {
    S.op(x+" ,");
    }
```

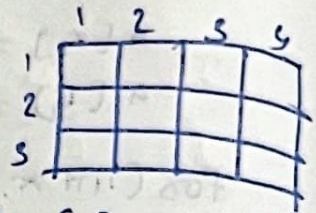
Two dimensional Array.

* **Syntax** :- A[][] = new int [3][4];

row column

Array of reference — Array of element





initalizing :

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

Access :-

```
for (int x[]: A)
{
  for (int y: x)
  {
    S.o.pln (y);
  }
  S.o.pln (" \n");
}
```

OR

```
for (int i=0; i<A.length ; i++)
{
  for (int j=0; j<A[0].length; j++)
  {
    System.out.println (A[i][j]);
  }
```

* **Matrix Multiplication**

```
int A[][] = {{1,2,3} {3,4,5} , {1,5,8}};
int B[][] = {{1,1,1}, {2,2,2}, {3,3,3}};
int C[][] = new int[3][3];
```

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 3 & 4 & 5 \\ 2 & 1 & 5 & 8 \end{array} \qquad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 2 & 3 & 3 & 3 \end{array} \qquad \begin{bmatrix} 00 & 01 & 02 \\ 10 & 11 & 12 \\ 20 & 21 & 22 \end{bmatrix}$$

$$C[i][j] = \sum A[i][k] * B[k][j];$$

```
for (int i=0; i<3; i++)
  {
    for (int j=0; j<3; j++)
      {
        c[i][j]=0;
        for (int k=0; k<3; k++)
          {
            c[i][j] = c[i][j] + A[i][k] * B[k][j];
          }
      }
  }

for (int x[] : c)
  {
    for (int y : x)
      {
        s.o.p(y+"\t");
      }
    s.o.pln(" ");
  }
```

* for sorting in Java

```
String arr[] = {"Java", "python", "c++", "HTML"};
java.util.Arrays.sort(arr);
for (String x : arr)
  {
    s.o.p(x+" " + ", ");
  }
```

Methods

→ It is members of class which provides functionality to class

* **Body of Method**

signature

```
returnType methodName(parameter List)
{
  ==
}
```

ex ① **Max method**

```
class Test
{
  static int max(int x, int y)   ← def of method
  {
    if (x>y)
      return x;
    else
      return y;
  }
  public static void main(String arg[])
  {
    int a=10, b=15, c;
    c= max(a,b);   ← call of method
    S.o.pln(c);
  }
}.
```

* If we don want to write static int.

then:-

```
p S Vm (Str.. arg)
{
  int a= 10; b= 15;
  MethodPractice mp = new MethodPractice();
  S.o.pln(mp.max(a,b));
```

* Passing array as parameter

```
class
{
    Static void update (int A[])
    {
        A[0] = 25;
    }
```

- if method is modifying the object then the actual object will be modified
- if any method is not mentioned then we can write **void**

* Examples

1)
```
int main (int x, int y)
{
    return x>y ? x:y;
}
```

2)
```
String UserName(String email)
{
    int a = email.indexOf('@');
    String name = email.substring(0,
    return name;
}
```

* Parameter Passing → only 1 method

ex-①:

```
Static int add (int x, int y)
{
    int z;
    z = x+y;
    return z;
}
p.s.v.m (Str-- arg[])
{
    int a=10, b=15, c;
    → c = add(a,b);
    S.o.pln (c);
}
```

Formal Parameter

method call

actual Parameter

→ content of actual parameter is copied in formal parameter

Primitive

int, char, float, double

Pass by value

It was value/ data which was copied

ex-②

```
Static void Welcome (String n)
{
    S.o.pln ("Welcome Mr./Miss"+n);
}
p.s.v.m (---)
{
    String name = " victor";
    Welcome(name);
}
```

→ content of actual parameter is copied in formal parameter

Object

Pass by reference

string.

It is reference / address which is copied

* Method overloading

→ It allow to write more than one method with same name having different parameter.

ex ①
```
int max(int x, int y)
{
return x>y ? x : y;
}
```

②
```
float max(float x, float y)
{
return x>y ? x : y;
}
```

③
```
int max(int x, int y, int z)
{
return x>y && x>z
x : (y>z) ? y : z
}
```

→ All is consider different known as method overloading

* variable arguments

ex:
```
show();
show(10);
show(10,20,30);
show(10, 20, 30, 40);
show(10, 20, 30, 40, 50, ---);
```

∴ it will act as array

```
void show(int...x)
{
for (int a : x)
{
s.o.p(a);
}
}
```

→ one method act with different arguments known as variable argument

→ variable argument must be at last → void show(int x)

→ only one variable argument is allowed ✓

* Recursion method

→ It will call itself known as Recursion method.

ex:
```
void fun(int n)
{
if (n>0)
{
s.o.p(n)
fun(n-1);
}
}
```

```
void fun(int n)
{
if (n>0)
{
fun(n-1);
s.o.p(n);
}
}
```