android → Java based.

Programming Language :-
Giving instruction to the Computer

Algorithm :-
Step by Step Procedure for solving Problem or Task

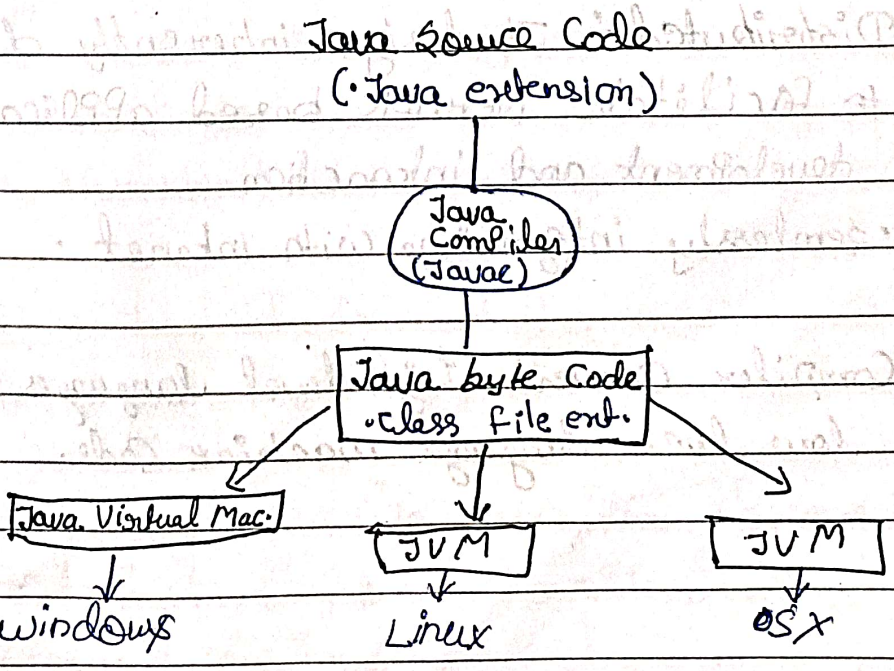
Syntax :-
Structure of word in sentences for Programming
exact syntax required.

Java :-
Developed by James Gosling (at Sun Microsyst
1995.

Principle :-
"Write once Run anywhere". with Platform
Compatibility.

Java Version :-

Java 1.02	Java 1.1	Java 2 (1.2-1.4) Versions	Java 5.0 (1.5 = w) Version
• 250 class	• 500 class	• 2300 class	• 3500 class
• Slow	• little faster	• Much faster	• More Powe easier to Deb



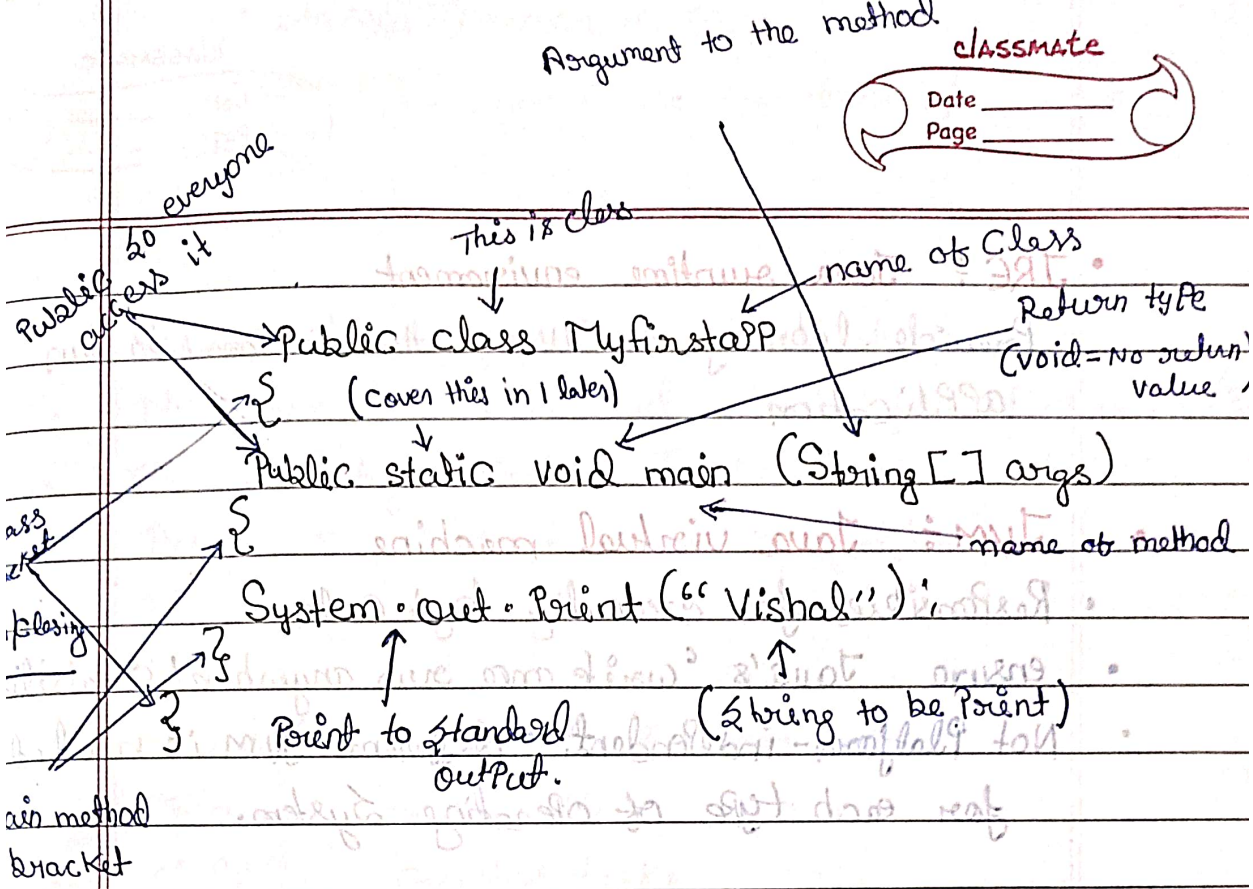
Benifit of Java :

- Potability - write once Run anywhere
- Security - own of virtual machine.

Java Buzzwords :

- Robust : due to strong memory management
  - exception handling
  - Types checking mechanism.
  - Preventing System Crashes and ensure Perform
- Multithreaded : ability of CPU to execute multiple thread Concurrently.
  - More efficient Processing and task management.
- Architectural Neutral : It is arch. Neutral because it Compile Code (bytecode) Can run on any devic with a Java virtual machine.
- Interpreted and high Performance :
  Its bytecode is interpreted by the Java VM which onfley Just in time (JIT) compilation for efficient and fast execution
- Distributed : It design inherently distributed to facilitate Network based application development and interaction
  - Seamlessly integrating with internet.

Compiler Convert high level language to low level language machine code.



File extension :-

- Java.
  - Contain Java Source Code
  - High level human Readable
  - Use for development
  - File editable
- Class → Collection of object
  - Contain Java bytecode
  - for Consumption of JVM
  - Used for execution
  - Not meant to be edited

- JDK = Java development kit
  - Software development kit require to develop Java Application.
- It include JRE (Java runtime Enviorement) :- interpreter (Java), Compile (Javac) document generator (Javadoc)
- JDK is super set of JRE

- JRE : Java runtime environment
  Provide library, the JVM, other component to run application

- JVM : Java virtual machine
- Responsible for executing byte code.
- ensure Java's 'write once run anywhere' capability
- Not Platform-independent : different JVM is needed for each type of operating system.

Println : next line
  ln used to Print the word in next line after its execution line.

→ Importance of main method :-

Access Specifier ↓ Public static void main (String args[])
Return type ↑ void
Array of String type ↑ String args[]
Keyword ↑ static
method name ↓ main

- Entry Point : It is entry Point of Java Program, where the execution start, without main method (JVM) does not know where to begin running the code
- Public and static : Main method must be Public & Static ensuring it's accessible to the JVM without need to Instance the Class
- Fixed Signature : This main method is fix Signature (Sequence), Deviating mean the JVM will not recognise the Entry Point.

- Storage ÷ Permanent Save.
- Memory ÷ Store in Ram (Not Permanent)

→ IDE ÷

Integrated development environment.

- It Consolidate basic tool require for Software dev
- Central hub for Coding, finding Problem, and testing
- Design to improve developer efficiency.

Need ÷

(1) Streamline development
(2) Increase Productivity
(3) Simplifies Complex tasks.
(4) Offer a unified workspace
(5) IDE features ÷
① Code AutoComplete ③ Version Control
② Syntax Highlighting ④ Error checking

→ Data types, Variables & Input ÷

(1) Variable ÷ Variable is like Containers used for storing data values

Ex ⇒ It may be string, number, boolean in a box.

Declaring a variable, assignment operator:

```
int a = 5
```

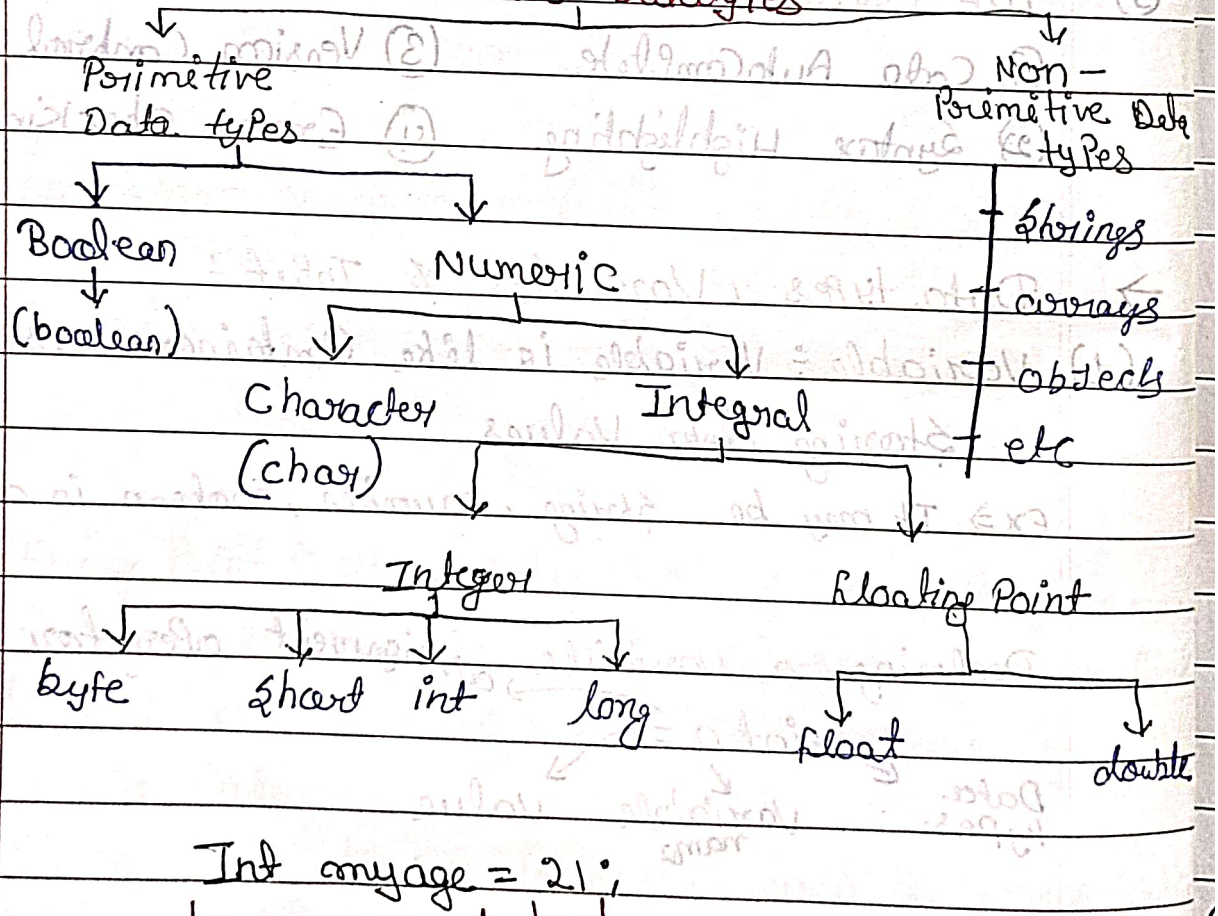
Data types: int — Variable name: a — value: 5

classmate
Date
Page

÷ DataTypes ÷

Name	Size	Default value	Type of value
Byte	1 byte	0	Integral
Short	2 byte	0	"
Int	4 byte	0	"
long	8 byte	0L	"
Char	2 byte	'\u0000'	Characters
float	4 byte	0.0f	Decimal
double	8 byte	0.0d	"
boolean	1 bit	false	True or false

Java DataTypes




```
Int myage = 21;
```

Declaration — initialization

Java identifier rule :-

(1) Only allowed characters are alphanumeric characters ([A-Z], [a-z], [0-9], '$' (Dollar sign) and '_' (underscore)

(2) Not use Keywords or reserved words

3) Identifier should not start with digit (0-9)

(4) Java identifier are Case sensitive (upper ≠ lower case)

5) No limit on length of identifier, altimum length use of 4-15 letter only.

Keywords in Java

Default	assert	enum	Package	extends
super	throw	goto	interface	final
switch	native	long	continue	try
strictfp	throws	const	Public	volatile
void	class	byte	finally	abstract
static	import	char	implement	synchronized.
break	catch	float	boolean	
Protected	transient	short	Private	
case	double	interface	instanceof	
implements	new	return	int	

Escape sequence :-

(1) \t → insert a tab in the text at this Point

ex:- Vishal Singh

2) \b → insert backspace in the text at this Point

ex:- Visha Singh

3) \n → insert new line in the text at this Point

Vishal
Singh

④ \' → insert single quote character in text at this Point.
Vishal 'singh'

⑤ \" insert double quote chercle in text at this Point.
Vishal "singh" vishal \"singh\"

⑥ \\ → insert a backslash chercle :
Vishal \singh\

User input :

```
import Java.util.Scanner;
Public class main
{
Public static void main (String [] args)
{
Scanner scanner = new Scanner (System.in);
System.Out.Print("Enter your name :");
String name = scanner.nextline();
System.out.println("welcome" +name);
}
}
```

Type Conversion and Casting

Implicit Conversion or Coercion

Explicit Conversion or Casting

```
// Implicit
Long big = 45;
float dec = 3;
double d = 3.4f;
```

```
// Explicit
float eDec = (float) 3.4;
long ebig = (Long) 3.4;
int e
```

classmate
Date
Page

Operators in Java

- Assignment Operator :
Assigns the right-hand operand's value to the left-hand operand.

ex int a = 6;

- Arithmetic Operators

+	Addition
−	Sub
*	Multiple
/	Division
%	Modulus

- Order of Operation :
BODMAS
PEMDAS
() $\sqrt{x}$ x^2 * ÷ + −
Parenthesis Exponent mul divide add Subtract

If same no. of operates use (→)

9 ÷ 3 x 2 ÷ 6
→ (Direction of solve)

- If solve any part of equation first then make it under the Parenthesis
(4 ÷ 2 + 7) x 4

→ Shorthand Operators :-

Operator Symbol	Name of the Operator	Example	Equivalent
+=	Addition assi.	x += 4;	x = x+4
-=	Subtraction	x -= 4;	x = x-4;
*=	Mult.	x *= 4;	x = x * 4;
/=	Div	x /= 4;	x = x/4;
%=	Remainder	x %= 4;	x = x%4;

→ Unary Operator :-

Operator	Description	Example
—	Convert Positive value into negative	x = -y
++x	Pre Increment	increasment Value by 1 and Decremen the use it for our Statement	x = ++y
--x	Pree decrement		x = --y
x++	Post increment	Use current values in Statement	x = y++
x--	Post decrement	then increase/decrese by 1.	x = y--

Important.

- For taking input from user Scanner must be u
  uppes the function :-

Scanner inPut = new Scanner (System.in);
(inPut name)

- for each inPut use after Sout inPut
  int name = inPut.nextInt();

To take Maths function :

- first Convert it into the double.
- Make another Parenthesis (( ) , Math) ;

ex⇒ For Compound interest Power math function used :

```
double Cominterest = P * Math.Pow((1+R1/100), T);
```

P → Principal, R → Rate, T → Time

If - else :

1. Syntax : Uses if (Condition) { } to check Conditic
2. if : execute block if cond. is true, skip if false.
3. else : execute a block when the if cond. if false.
4. Curly Braces - it can be omitted for single statemt but not recommended.
5. If-else ladder : multiple if and else if block ; only one executes.
6. Use Variable : Can store Conditions in variables for use in if statement.

ex⇒

```
boolean Best = True ;
boolean Bad = false ;
if (Best) {
    SOUT ("Product Good") ;
}
else if (Bad) {
    SOUT ("Product Bad") ;
}
else
{
    SOUT ("Very bad") ;
}
```

Relational Operator

Equality :

== check value equality.

Inequality :

!= check value inequality.

Relational :

> Greater than.

< Less than.

>= Greater than or equal to

<= Less than or equal to.

- Order of relational operators is less than arithmetic operators

Logical Operators :

1. Types : && (AND)
   || (OR)
   ! (NOT)

(i) AND : All condition must be true for result to be true

(ii) OR : only one condition must be true for result to be true.

(iii) NOT : Invert the Boolean value of a condition

= Lower Priority than Math and Comparison operator.

classmate
Date
Page

Operator Precedence :

Determine the evaluation order of operators in an expression based on their Priority level.

Associativity : Define the order of operation for operators with the same Precendence, usually left to right or right to left.

	Operator type	Category	Precedence	Associativity
i)	Unary	Postfix	a++, a--	R to L
		Prefix	++a, --a, +a, -a, ~, !	R to L
ii)	Arithmetic	Multiplication	*, /, %	L to R
		Addition	+, -	L to R
iii)	Shift	Shift	<<, >>, >>>	L to R
iv)	Relational	Comparison	<, >, <=, >=	L to R
		Equality	==, !=	L to R
v)	Bitwise	Bitwise AND	&	L to R
		Exclusive OR	^	L to R
		Inclusive OR	|	L to R
vi)	Logical	Logical AND	&&	L to R
		Logical OR	||	L to R
vii)	Ternary	Ternary	? :	R to L
viii)	Assignment	assignment	=, +=, -=, *=, /=	R to L
			%=, &=, ^=, |=, <<=	
			>>=, >>>=	

[Intro to Number system]

Binary number system :- only two Possible value 0 and 1.

Ex ⇒ $(11010.11)_2$ value in decimal

$1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 1\times2^{-2}$

$16 + 0 + 8 + 2 + 0 + 0.5 + 0.25 \Rightarrow [26.75]$

→ Bitwise Operators :-

(1) AND Operator (&) : both bits one (1) to result 1 other wise 0.

Ex :- 12 & 13

12 = 1100

13 = 1101

1100

12 & 13 = 12

Both one then (1)

(2) OR operator (|) : If both bit is zero (0) then result 0 ; Otherwise 1.

Ex ⇒ 12 | 13

12 = 1100

13 = 1101

1101

12 | 13 ⇒ 13

Both zero then (0)

(3) XOR Operator (^) : Both ~~Bits~~ Bit are different (0,1) result 1 otherwise same 0.

Ex :- 7 ^ 13

7 = 0111

13 = 1101

1010

7 ^ 13 = 10

• Different get 1 same get 0

Complement

4) Not operator (~): It invert the bit from (0 → 1) & (1 → 0).

Ex = 7 = 0111 → 1000

5) Left shift operator (<<): shift the left operand bit to the left.

Ex ⇒ 4 << 1 ⇒

0100 << 0001

1000

) Right shift operator (>>) shift the left operand bits to the right.

- if no. is (+ve) the filling digit will be 0
- if no is (-ve) then filling digit will be 1.

Ex ⇒ 1 >> 4

0001 >> 0100

0010

#Extra

Q: check even or odd no. by using bitwise operator

⇒ num & 1 == 1 odd

num & 1 == 0 even

Ex ⇒ num = 14 ⇒ 1110

0001

0000 → even no

- Binary number system use base 2
- Decimal number system use base 10

Java Comments :

- Used to add notes in Java Code.
- Not displayed on the web Page.
- Helpful for Code organization.

Syntax :

(i) Single line : //

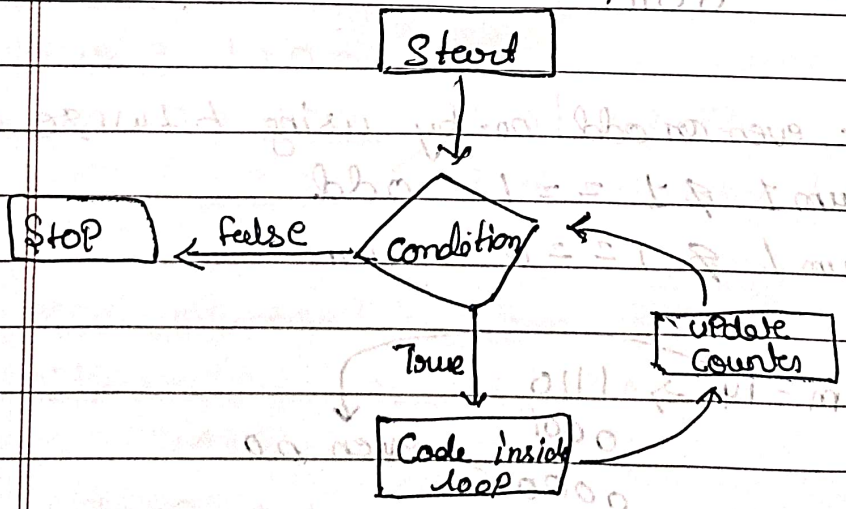
(ii) Multi line : /* ..... */

(iii) Java Docs : /** ..... */

LOOP

→ Loop = ICU → initialization
→ updation.
→ condition

- Code than run multiple times based on a condition.
- Repeated execution of Code.
- Loops automate repetitive tasks.
- Types of Loops : while
  : for
  : do-while
- Iterations : No. of times the loop runs



While Loop :

```
while (condition)
{
  // Body of the loop
}
```

- Used for non-standard condition
- Repeation a block of Code while a Condition is true.
- Always include an update to avoide infinite loops

Functions / Method :

- Block of reusable Code
- Dry Principle : "Don't Repeat Yourself" it encourage code reusability
- Usage : Organizes Code and Perform Specific task.

Syntax :

```
Modifiers  Return type  Method name        Parameter-list
    ↓          ↑            ↑                   ↓
  Public int addition (int operand 1, int operand 2)
  {
      int sum;
      sum = operand 1 + operand 2;        } → Body of the Method
      return sum;
  }
```

use (function) keyword to declare.
use () to contain Parameters.

classmate
Date
Page

Return Statement :

- Send a value back from a function

→ Return would be : Value, Variable, Calculation

- Return end the function immediately.
- Function Call make a Code Jump around
- Prefer returning value over using global Variable.

Arguments Vs Parameters :

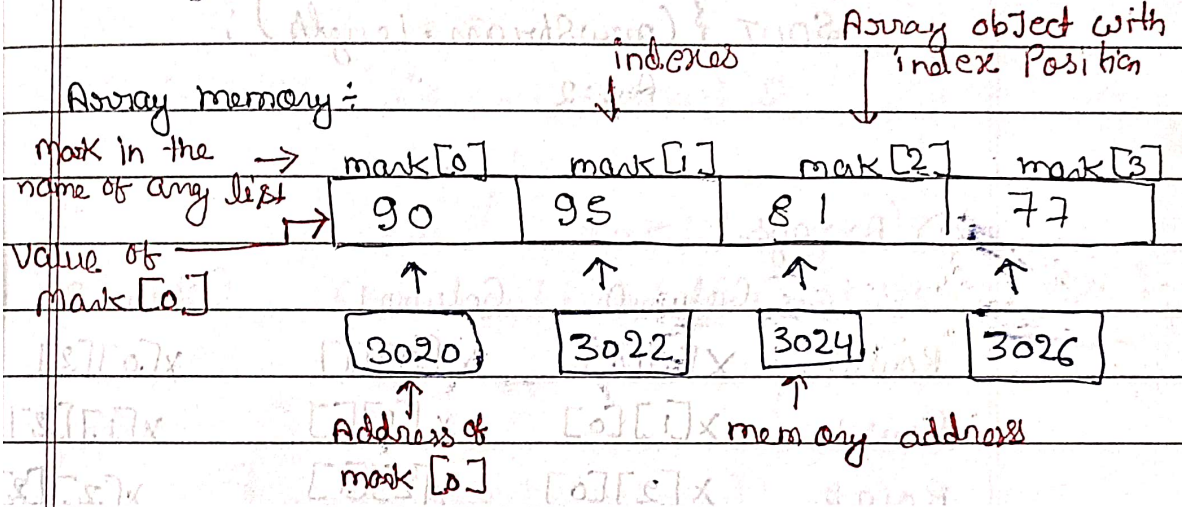
1. Input value that a function takes
2. Parameter Put value into function, while return gets value out.
3. Naming Convention Same as Variable name
4. ex ⇒ System.out.Print(); Scanner.nextInt() or function we have already used
5. Multiple Parameter : function can takes more than one.
6. Default Value : Can set a default value for a Parameter.

classmate
Date
Page

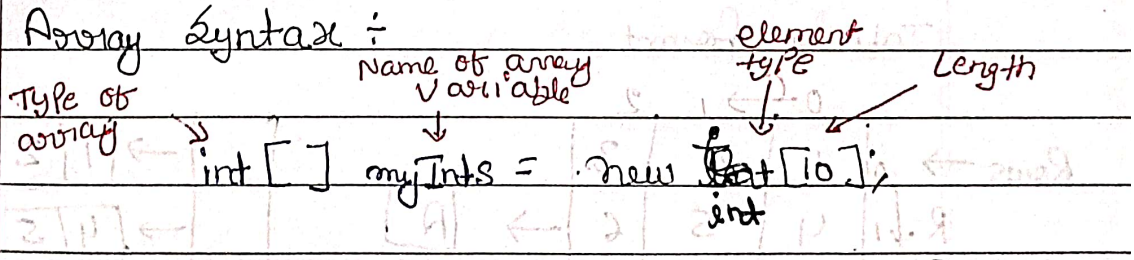
Array

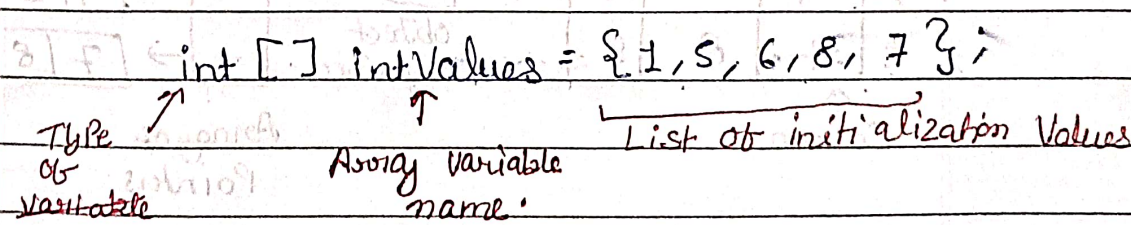
Array Length = Array last index + 1

- An array is just a list of values.
- Index : start with 0
- Array are used for storing multiple values in a single variable.



Array Syntax :





Use {} to create a new array, {} bracket enclose list of element.

Array can be saved to a variable

Accessing Values : Use [] with index.

- Bracket start and end the array
- Values separated by Commas
- Can span multiple lines.

Declare a String in Array :

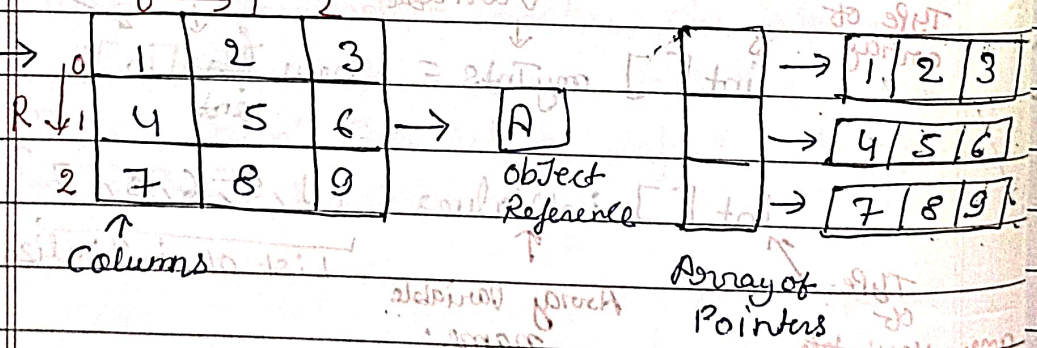
```
String[] strArr = new String[4];
StrArr[0] = "My Name";

String[] newStrArr = {"One", "Two"}
SOUT $(newStrArr.length);
```

Ans = 2

2D Arrays :

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Tabular format



2D array Declaration

```
int[][] numArr = new int[2][3];
int[][] inArray = {{1,2,5}, {8,9,4}};

numArr[0][0] = 5;
```

row Column

To Print an 2D array :

```
Public class TwoDimensionalArray {
  Public static void main (String[] args) {
  int [][] arr = {{1,2,3},{4,5,6},{7,8,9}
  System.out.Print
      int i = 0;
  while (i< arr.Length) {
          int J=0;
      while (J< arr[i].length){
        System.out.Println(arr[i][J] + " " );
          J++ i
      }
    System.out.println ();
    i++;
  }
}
}
```

Array Utility :

Array utility is a sub class used to take input from user.

- It takes input from user in array and modified according to sequence.

```
import java.util.Scanner;

Public class ArrayUtility {
  Public static int[] inputArray() {
    Scanner input = new Scanner(System.in);
   System.out.print("Enter the number of elements:");
     int size = input.nextInt();
      int[] num = new int[size];
        int i=0;
    while (i<size) {
     System.out.Print("Enter the element no"+(i+1)+":");
     num[i] = input.nextInt();
        i++;
    }

    return num;
  }
}
```

How to Call this sub-class in any Program:

```
int[] numArr = ArrayUtility.inputArray();
```

Display an array :-

```
Public static void DisplayArray(int [] numArr) {
    int i=0;
  while (i < numArr.length) {
      System.out.print(numArr[i] + " ");
        i++;
    }
      System.out.println();
  }
}
```

Simply / Traversal of Array. (1D)

```
Public class Array {
    Public static void main (String[] args) {
    int[] values = {1,2,3,4,5};
      int index=0
      while (index < values.length) {
          System.out.println(values[index]);
          index ++;
      }
  }
}
```

Logic about Pattern :

```
*
* *
* * *
* * * *
* * * * *
```

```
int rows = 0;  //Row start with 0-8(5)
while (rows < 5) {
   SOUT("* ");  //Print * in Line
   int i = 0;   // for 2nd * in 2nd line
  while (i<rows) { // Loop inside loop
    SOUT (" *");   // 2nd step & next
     i++;
  }
   SOUT( );
   rows++;
  }
}
```

If we use array to not known size
So we input same value take array size of same length

```
int size = input.nextInt();
int[] num = new int[size];
```
↑ It take the no. of I/P you take as its length

2D array Traversal :-

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

int i=0;
while(i<arr.length){          {for i (0<2) ⇒ 0,1,2}
    int j=0;
    while(j<arr[i].length){   {0,1,2}
        System.out(arr[i][j] + " ");   Print element
                                       → Make Space
        j++;
    }
    System.out();   → for each new column
    i++;
}
}
}
```