Java.

(i). Platform independent (Byte code is independent of any platform)

* Byte code - after the compiler compiles the java file
(.java file) → (javac compiler).

* Native Code -

Java Compiler - Compiles text file into platform independent java file.

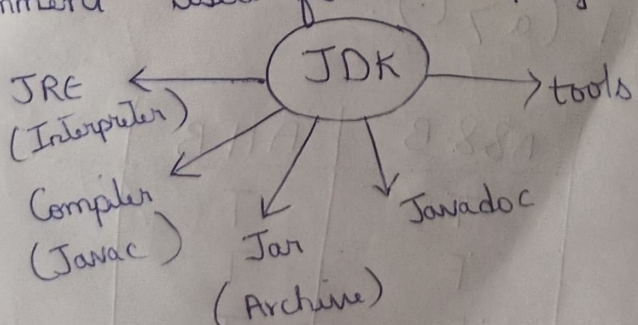
What Compiler do?
- Checkes Syntax error.
- Converts Source code into byte code with JVM.

Java Interpreter - Reads byte code & execute it.
What Interpreter do?
- Convert byte code → native code.
- Execute native code.

JIT (Just-IN-Time) → Same as Interpreter.

JVM (Java Virtual Machine) - Provides environment to run Java application.

JDK (Java Development Kit) - Software development environment used for developing Java appl$^n$ & run them



* JRE (Java Runtime Environment)

Download JRE only to run appl^n.

Source Code → (Text file saved as .java)

## Character Set

Digit (0-9), Alphabet (A-Z, a-z).

## Keywords

* Predefined words with specific meanings
* Eg- byte, short, int, long etc
* Cannot be changed (meaning).

## User defined words

'_' & $ or alphabet

## Variable

Data type Var_name = Value

```
int a = 5;
```

## Operators

(1). Arithmetic operator → (+, -, *, /, %)

(2). Relational (>, >=, <, <=) → (Boolean)

(3). Logical (&&, ||, !) etc

(a>b) && (a>c)

(a>b) || (a>c)

| A | B | A&&B | A\|\|B |
|---|---|---|---|
| T | T | T | T |
| T | F | F | T |
| F | T | F | T |
| F | F | F | F |

# Java

Data types

Primitive

byte - 1 [-128,127]
Short - 2
int - 4
long - 8 (L).
float - 4 (3.14F)
double - 8.
Char - 2 a,b,c
boolean - 1 (T/F)

Non-Primitive (Immutable).
String
CharAt. (User defined).

Functions → Concat → Strings
Length,
CharAt
Replace.

*. Strings are immutable.

Array - It's one variable that holds many values.

Arrays. (Non-primitive). (1-Dimension).

1. Datatype arrayname [size] = new datatype [ ] (without value)
   arrayname [0] = Value
   arrayname [1] = Value.

Sorting array → Array.Sort (marks) (Ascending).
S.O.P (variable [ ]);

2. Datatype [ ] marks = { value, value, value } (with value).

2-Dimension
datatype[ ] [ ] variablename = { { }, { } }.
S.O.P. (variablename [index] [index]);

Increment / Decrement operator (++, --).

(i) Prefix (++ variable) Value increase before it is used in expression

(ii). postfix (variable ++) Value is used then it is incremented

## Control Statements

(i). Conditional Control Statement (If & Switch Stat").

(ii). Looping Control Statement

(iii). Unconditional Control Statement.

Program.

* Pass 3 ~~lines~~ numbers and find minimum & max.

```
public class Lab1 {
public static void main (String [] args){
int a = 5; int b = 2; int c = 1;
if ((a > b) && (a > c))
{
S.O.P ("Max" + a);
} else if (b > c) {
S.O.P ("Max" + b);
} else {
S.O.P ("Max" + c);
}
if (a ~~==~~ a < b) {
S.O.P
```

## Scanner Class

It is a class that obtain input from various sources such as files, keyboard & strings.

Steps:-

Import - import java.util.Scanner;

Create object - Scanner input = new Scanner (System.in);
(for keyboard input)

Read data - nextInt(), nextDouble(), nextLine()
next() etc.

Close: input.close().

## Key Methods

nextInt(): Reads an int value
nextDouble(): Reads a double value.
nextLine(): Reads an entire line of text.
next(): Reads next token as string.
Close(): Close the Scanner.

## Types of Loops.

Looping statements execute over & over again, in a loop. It continues until the condition result is true.

(i). For Statement - It execute in a loop cont.

Usage - We use for loop when we know the no. of times the loops is gonna execute.

Syntax:- For (init"; Condition ; inc/dec)
{
S1; S2;
}

(ii). While Statement

*. Executes a single line continuously until condt^n is met.

Syntax: init^n;
while (condit^n)
{
S1;
S2;
inc / dec
}.

(iii) Do while Statement

Executes a particular code until the exp^n is true.

Syntax: init^n;
do { S1;
inc / dec
} while (condit^n);

Unconditional Control Stat^n

Break - It transfers the control to end of block.
Continue - It transfers to the beg^n of block.

# Functions

* Instead of writing the same code again & again, we create function & use it whenever needed.
* Using funct", Store multiple Values using arrays.
* Small piece of code that does one job & can reuse it.

Syntax :-
```
int add (int a, int b){
    return a+b;
}
```

Predefined

Eg of method → S.O.P.

Str.equals(Str2) → Checks if two Strings are equal.

Str.toLowerCase() → Convert String to small letters.

* Use static to use function directly without creating object.
* Methods can be overloaded (Same name diff Para).

Components - * Access modifier

* Return type: Specifies the data type of the value, the method returns. if method doesn't require a value use void.
* Method name - Should be unique.
* Parameters - Input value to pass in method.
* Method Body - Code block encloses in { } that performs method task.

Syntax :-
```
accessModifier returnType methodName (par1, par2)
{
    return value;
}
```

PascalNamingConvention → Classes

CamelNaming Convention → Methods.

## Constructors

* It is a special method whose name is same as that of class name.
* It does not have any return type.
* Constructs values.

Syntax.

```
Class ClassName {
        // Const^n decl^n
    ClassName (parameter1, par2, ....)
        //init^n Code
}
```

* Constructor body contains the code that initialize the object state.
* Calling a Constructor -
* Constructor can be overloaded i.e., you can write other Constructors by changing arguments.

## Super This

It is used to refer to immediate superclass obj or instance variable. It comes into usage because of concept of inheritance.

## This Keyword

It is used inside method or Constructor of a Class.
It is used to refer member of a current obj within the instance of a method or Constructor.

* It will represent current class obj.
* Can be applied to Var, Constructor & method.

Syntax:

```
this.a;   // Var Case
this.m1(); // method
```

## Access Modifiers

* It is a keywords that determine access of class, methods in a program.
* It determine where a field in a class can be used by another method in another class / Sub-class.
* It can also be used to restrict access.

## Types

(i). Private - It can be applied to global var, method, Constructor.

- Can be accessed only within enclosed brackets.

(ii). Public - It can be applied to Const, global Var, Static Var, methods, inner class / outer Class.

- It can be access everywhere.

(iii) Protected

* It cannot be applied to outer class, but can be applied to inner class.
* Cannot apply to local variable.
* It can be accessible within same package and another package of its subclass only.
* If any method is overridden from super class to sub class, then access specifier of the overridden method must be protected or public.

(iv). Default.

* When an access specifier is not specified to any member of a class, then it is default.
* It can be access only within same package/folder.
* It is a keyword in java. We may mistakenly think that is an access specifier, but it actually is used for switch statement.