INTRODUCTION TO JAVA

.java file	Compiler	.class file	Interpreter	Machine code
(Human Readable)	(Entire File)	(Byte Code)	Line by Line	(0's & 1's)

- **Source Code**
  - The code written in java in human readable and it is saved using extension .java
  - The code is known as source code

- **Java compiler**
  - Java compiler converts the source code into byte code into byte code which have extension .class
  - The byte code does not directly run on system we need JVM for this.
  - Reason why java is platform independent.

- **Java interpreter**
  - Converts byte code to machine code i.e 0's and 1's
  - It translate the byte code line by line to machine code.

JDK = JRE + Development Kit
(Java Development Kit

JRE = JVM + Library classes
(Java RunTime Environment)

JVM
(Java Virtual Machine)

JIT
(Just-in Time)

- JDK

- Provide Environment to develop and run the java program.
- It is package that includes:

Development tools - To provide an environment to run your program.

JRE - To execute your programs

compiler, Archiver - javac, Jar

Docs generator - Javadoc

Interpreter/Loader

- JRE

It is installation package that provides environment to only run the program.

It consist of -

Development technology

User interface toolkit

Integration libraries

Base libraries

JVM - Java Virtual Machine

Compile Time → .java file → .class file

Class Loader — Step 1: Classloader loads all classes needed to execute the program.

Bytecode Verifier — Step 2: JVM sends code to bytecode Verifier to check the format of code

Interpreter

Runtime

Hardware

M T W T F S S

Page No.:

Date:

YOUVA

- (How JVM Works) Class Loader
- Loading

→ Read .class file and generate binary data

→ an Object of this class is created in heap

- Linking

→ JVM verifies the .class file.

→ allocates memory for class variables and default values.

→ replace symbolic reference from the type with direct references.

- Initialization

→ All static variables are assigned with their values defined in the code and static block.

→ JVM contains the stack and heap memory locations.

- JVM Execution

- Interpreter

line by line execution

when one method is called many times it will interpret again and again

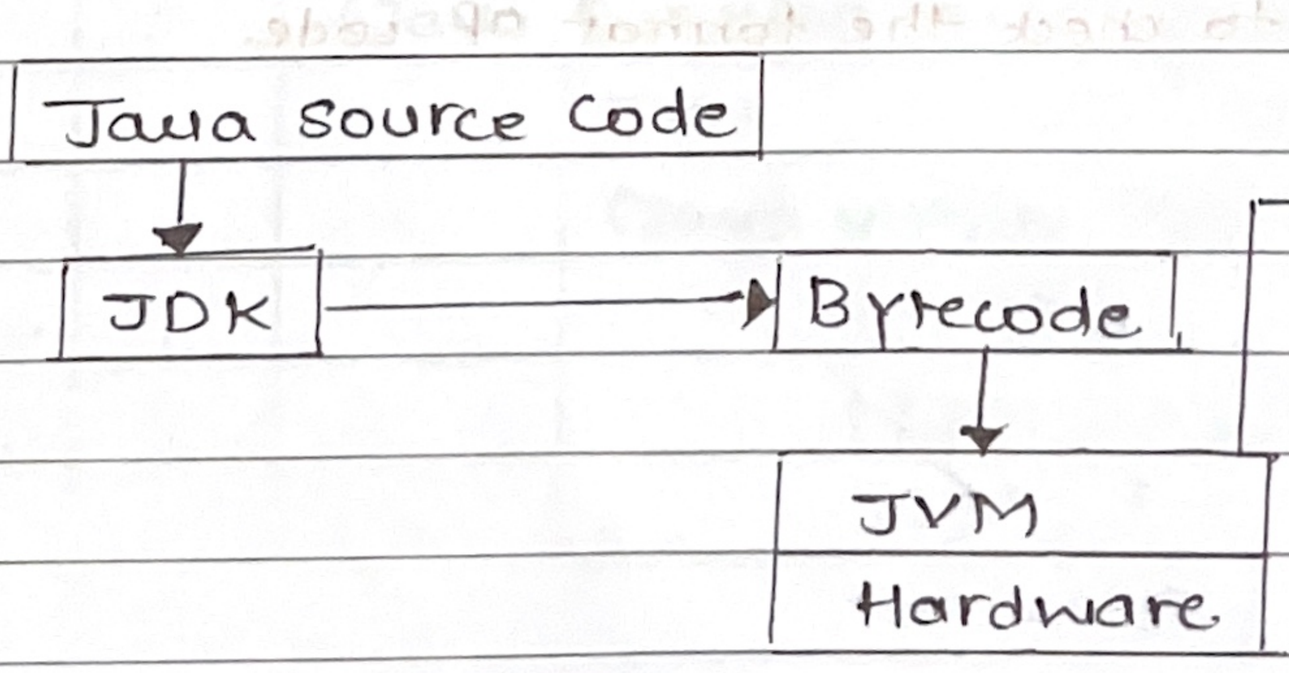
- JIT

Those method that are repeated JIT provides direct machine code so the interpretation is not required.

Makes execution faster.

Garbage collector

- Working of Java Architecture-



- Structure of Java File
  - A class with same name as file name must be present in .java file.
  - class which is having same name as file must be public class.
  - A main function/method must be present in the public class, main is a function from where the program start.

Recommended to use capital Alphabet of first letter of class not compulsory.

- javac file_name - Command to convert .java to .class
  For eg:
  javac Main.java.

- java filename - command to run program
  For eg:
  java Main

- Hello World Program.

```
public class Main {
```

Name of class same as file.

access modifier

name group of properties and function

```
    public static void main (String [] args) {
        System.out.println("Hello World");
    }
}
```

Output:

Hello World.

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

public → access modifier
static →
void → return type
main →
String[] args → CL argumen of string

- public - public is an access modifier which allows function to access the ~~class~~ from anywhere.
- static - It is a keyword which helps the main method to run without objects.
- void - It is return type that does not return value.
- main - name of method.

```
                var
System.out.println("Hello World")
```

System → class
out →
println → fn/method
"Hello World" → arguments

print - does not add new line
println - add new line

- System - It is final class defined in java.lang package.
- out - It is variable of PrintStream type which is public and static member field of System class
- println/print - They are method of PrintStream class it prints the arguments passed to it.

- Taking inputs
- The Scanner class is used to get user input, and is found in the java.util package.
- To use Scanner class we have to create an object of the class and then we can use any available method found in Scanner class.
- use the object to take input from user

Methods -

nextInt()	Reads a int value/me from the user
nextLong()	Reads a long value from the user
nextFloat()	Reads a Float value from the user
nextDouble()	Reads a double value from the user
nextByte()	Reads a byte value from the user
nextShort()	Reads a short value from the user.
nexBoolean()	Reads a boolean value from the user
nextLine()	Reads a String value from the user
next()	It will take one word input until a space occurs

Syntax -

```
import java.util.Scanner;
public class Main {
    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);
    }
}
```

class req. to take input

object - keyword used to create object

System is a class and in is a variable that denotes we are taking input from standard input stream (i.e keyboard)

Int Input -

```
int rollno = input.nextInt();
```

float Input -

```
float marks = input.nextFloat();
```

String Input -

```
String s = input.nextline();
```

Primitive Data Types

Primitive Data Types are those data types which is not breakable

4 bytes	int	- -2,147,483,648 to 2,147,483,647
8 bytes	long	- -9223372036 854 775808 to 807
2 bytes	short	- -32768 to ...767
1 byte	byte	- -128 to 127
8 bytes	double	- sufficient for storing 15 decimal digit
4 bytes	floagt	- sufficant for storing 6 to 7 decimal digit
1 bit	boolean	- true/false values
2 bytes	char	- single character/letter/ASCII values

default decimal value stores in ~~float~~ double.
default whole value store's in int
∴ We use "f" and 'l' at end of float and doullong

data type ← int a = 10 → liberal (value)
↓
identifier (variable name)

- Identifier - Identifiers are the name of variables, method, classes, package and interfaces
- Liberal - Java literals are syntatic representations of boolean, character, numeric or string data.

int a = 234_000_000;
↳ value will be 234000000, underscore will be ignored.

- Comments
  - Single line comment //
  - Multi line comment /* */

- Type casting and Type conversion

Widening	Narrowing
Automatic	Explicit

- Widening or Automatic Type Conversion
  - Two datatypes are automatically converted.
  - This happens when we assign value of smaller datatype to bigger datatype and two datatype must be compatible.

byte → short → int → long → float → double

Eg- int i = 100; → 100
long L = i; → 100
float F = L; → 100.0

- Narrowing or Explicit Conversion
  - This happens we want to assign a value of larger data type to smaller datatype

double → float → long → int → short → byte

Eg- double d = 100.04; → 100.04
long L = (long) d; → 100
int i = (int) L; → 100

- Automatic Type Promotion in Expressions:
  - While evaluating expression, the intermediate value may exceed the range of operand & hence the expression value will be promoted.
  - Conditions:
    Java automatically promotes each byte, short, char to int when evaluating an expression.

b = 42; c = 'a'; s = 1024; i = 50000; f = 5.67F, d = .1234

eg: After solving expression:

(f * b) + (i/c) - (d * s)

↓ ↓ ↓

float + int - double = double

convert to biggest one

Explicit type casting in expression -

Eg : byte b = 50;

b = (byte) (50 * 2); → type casting into byte