→ it is distributed and secure.

20/10/22

Portable / Platform independency / WORA

① Platform :- Processor + OS → Called platform.

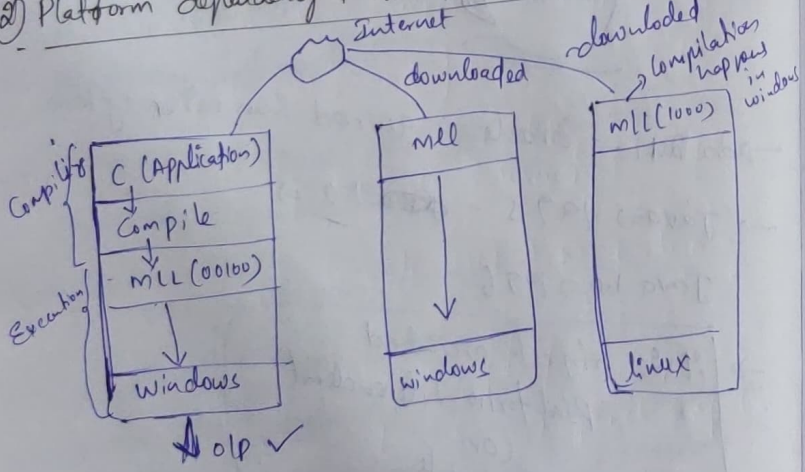
intel + windows

M1 + Mac

intel + linux

S/w ⇒ platform ⇒ OS → windows, mac, linux

② Platform dependency & non-portable :-



→ Program (or) App

Case ⓪

platform of Compilation	Execution	Result
windows	windows	✓ we get O/P

→ over network all Code never be transported.

→ only Compiled Code will be transported. (MLL)

Case ⓪	Compilation	Execution	Result
①	windows	windows	✓ (we get result)
②	windows	windows	✓
③	windows	linux	✗ (no output)

→ MLL is always platform dependent.

↳ if it is Compiled in windows then it Executes in windows system only.

⓪ (i) Platform dependency :- if a programming language is platform dependent such a P.L where platform of Compilation and platform of Execution has to be same.

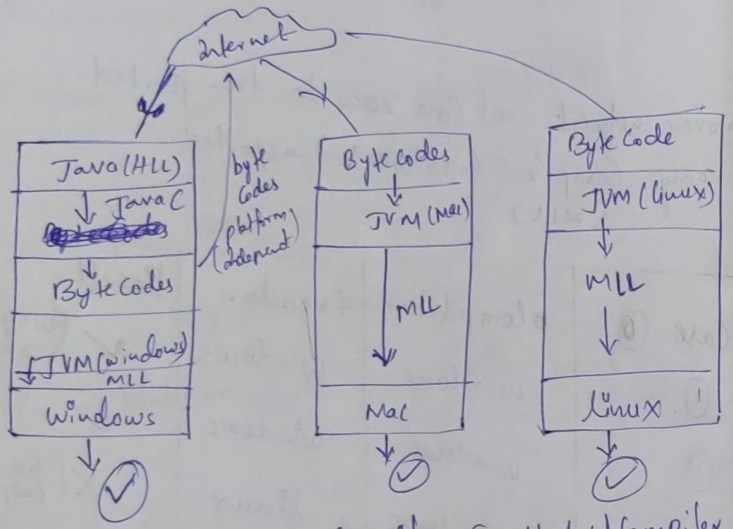
Eg: C & C++ platform dependent program.

→ we Cannot Port it and Execute it. hence it is called non-Portable.

Java

↳ ~~How java achived independency~~ ?

How Java achived independency?



→ Java used it own Compiler (or) Hybrid Compiler

→ Java Compiler (or) Javac

→ to make Java portable (or) platform independent

→ ordinary Compiler Converts HLL to MLL directly

→ to make Java portable (or) platform independency (or) WORA they are Java C (or) Special Compiler.

→ This Compiler will not Convert Java to M.L.L rather it Converts to Byte Code.

(i) Byte Code :-

→ Intermediate level code

→ they are neither in H.L nor in M.L

→ When Java Code is Compiler it is Converted to byte Code.

→ Byte Codes are platform Independent,
↳ it works on any platform.

→ JDK → JRE, Javac, JVM

→ JDK is different for all platforms

↳ JVM is platform dependent.

↳ different JVM's for all platform.

JDK ⇒ windows, linux, macintosh.
JVM

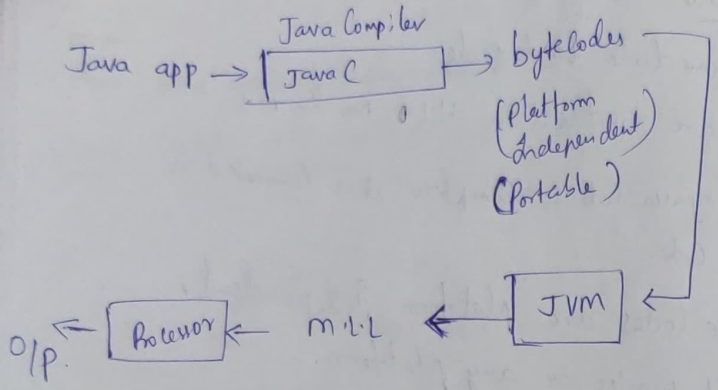
(ii) JVM :- designed using c language.

JVM is platform dependent.

→ JVM is different for different platform

→ JVM Converts will Convert byte Code into M.L.L (or) Machine understand language by Internally makes use of Interpreter & takes care further execution of Java.

→ Write once & run anywhere.

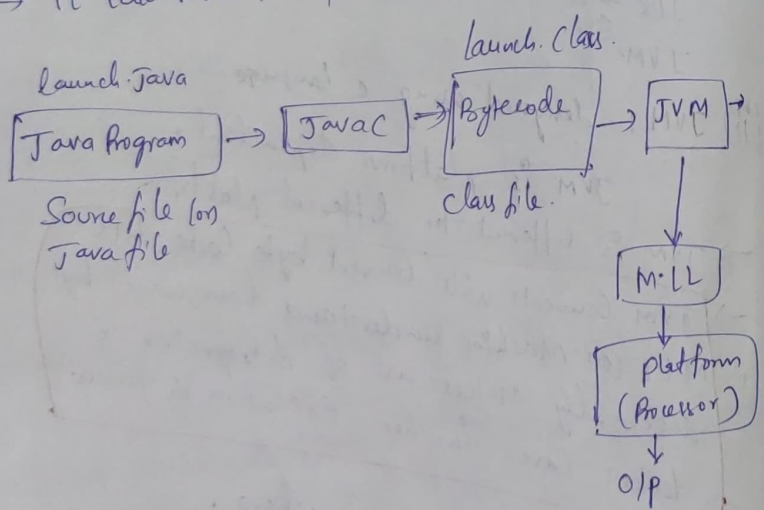


→ Java is architectural neutral.

→ internal programming language

→ WORA :- write once and Run Anywhere

→ it take more time for Execution than c++ & C.

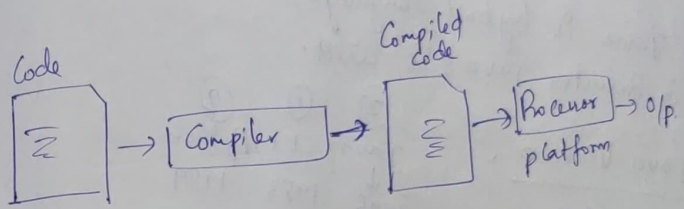


(iii) class file :- the place where byte codes are present is called class file.

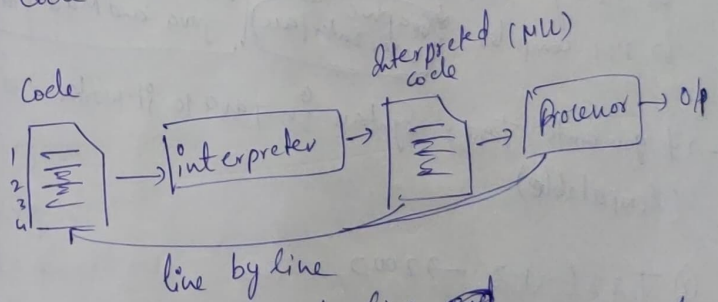
→ .class (Extension)

→ After Compiling a Java program it will generate Class file (byte codes).

(iv) Compiler vs Interpreter :-



→ Compiler will translate (or) Compiled Entire Code all at once.

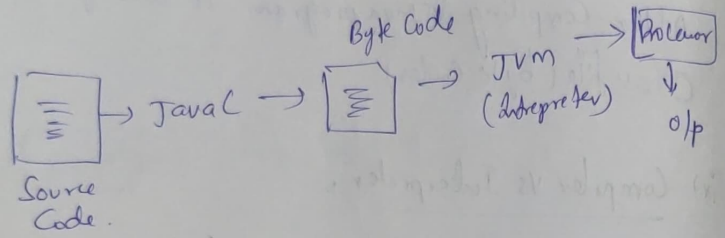


→ it interprets line by line.

→ if there mistake in 99th line Compiler throughs Error. In interpreter 98 line will work.

→ Eg: Python, Java script all are Interpreted Programming Language.

→ Java is Compiled and Interpreted language
→ Compiled at Early stage
→ Interpreted at Execution stage



→ Java is hybrid language.
→ In Industry Java 8 is used.

#Java Version :-

③ ① ②
Java, C++, Python
1996 1983 1989
1998 → J2SE 1.2.

① J2SE 1.2 – 1998
↳ Collection framework
↳ Swing → GUI (Graphic user Interface)
↳ JIT compiler

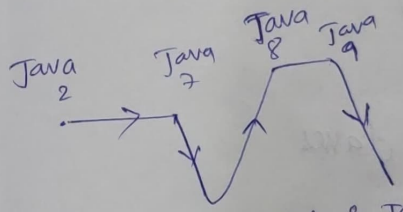
C++ → C#
Microsoft Copied from Java and Mad C#

→ if you write Java 2 Syntax in Java 10 it works (Compatable).

② J2SE 1.3 → 2000
③ J2SE 1.4 → 2002
④ J2SE 5.0 → 2004
↳ Annotations
↳ Auto boxing
↳ Enumeration
↳ Enhanced for Loop

⑤ Java SE6 – 2006 benchMark
⑥ Java SE 7 → 2011 → ~~bench~~Mark (oracle acquired)
↳ String in Switch
↳ try with resource
↳ <> diamond ~~operator~~ operator

⑦ Java SE 8 – 2014 bench Mark
↳ lambada Expression
↳ Support for JS Code
↳ Date and Time API
↳ Stream API



→ we Can write JS Code in Java.

⑧ Java 9 – 2017
↳ Modularity
↳ Reactive streams
↳ JShell

→ oracle Mad JDK paid. (for Commercial)
→ licence fee
→ Cloud started famous.

⑨ Java 10 – 2018.
↳ local variable Type Infe

⑩ Java 11 – Run source file
↳ in one step we can run Java code
→ Var for lambda.

⑪ Java 12 –

⑫ Java 13 – Switch Expression
Multiline - Strings.

⑬ Java 14 – Records
– Packaging tool

⑭ Java 15 –

⑮ Java 16 –

⑯ Java 17 – sealed classes

⑰ Java 18 –

⑱ Java 19 – Virtual threads
– vector API.

→ 70% of industry project are Java 8.

LTS Versions Long term Support.
↳ Provided by Company
↳ Java 7
↳ Java 8 ✓
↳ Java 11 ✓ focus] Company using this Version.
↳ Java 17 ✓

→ oracle JDK paid
↳ open source JDK (Alter).
↳ Amazon Corretto. ✓
→ oracle JDK free from Java ⑰

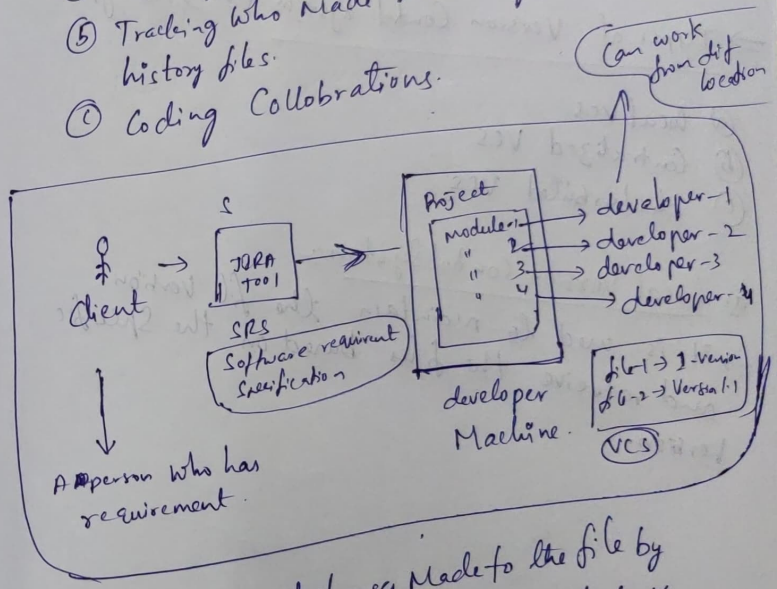
→ Jar → Zipping of .class files is called as "jar" file..
→ it stands for Java Archive file.

③ Git and Github Part-1

① Introduction
② What is Version Control System (VCS) & types of VCS
③ Git software Installation
④ Git Project Architecture
⑤ Git Commands for operations.
⑥ Git Command Execution
   ⓐ using Commandline
   ⓑ using IDE's like Eclipse / STS (Spring toolsuite) / intelij
⑦ git account Creation.
   ⓐ Public repository Creation
   ⓑ Private " "
⑧ Git folder structure
⑨ Git Braching Strategy
   ① developer branch
   ② master branch
   ③ release branch
   ④ hotfix branch
   ⑤ fork "
⑩ Project review process (PR process, Code Review) Code Merge.
⑪ Real time problems with git & how to fix them.

Introduction

→ Git is a popular Version Control System (VCS)
→ it was Created by linux Torvalds in 2005
→ it is Maintained by Junio Hamano
→ Git is used for
   ⓐ Tracking Code changes.
   ⓑ Tracking who made the changes like history files.
   ⓒ Coding Collobrations.



→ keeping track of changes Made to the file by the dev as per the changes Made by the Client requirements would be difficult in Machine.
→ To resolve this Problem we need to use "VCS".

→ It is a system that records changes Made to the file or set of files over the time, so that we can recall the specific version later.

→ for Every source Code Change in a file a new version will be Created.

⇒ Types of Version Control System (VCS): 3 types

ⓐ local VCS
ⓑ Centralized VCS
ⓒ distributed VCS

ⓐ local Version Control System :-

→ it is used to maintain the file version and retreive the files based on the specific version.