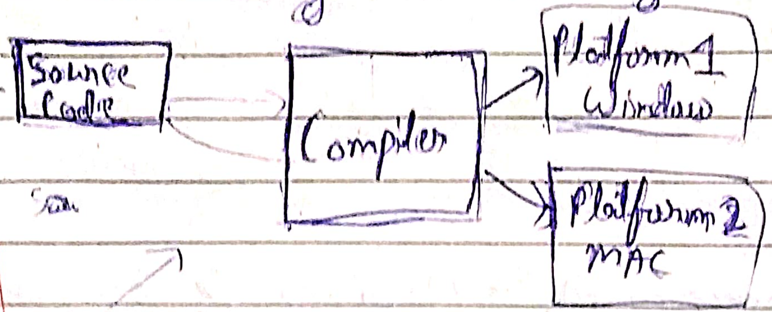
16/2/26 JAVA - Coder Army
DAY 1

Bytecode → provide Secured environment using Sandbox model (for security using JVM)

C/C++ Disadvantage / Shortcoming → JAVA Platform Independent (Portable) (Simple) (Secure)
- Platform Dependant — ✗ Independent
- Less simplicity (Pointers, multiple inheritance) (memory allocation/deallocation manually)
- Less Security

Platform
- Processor — follows ISA (Instruction Set Architecture) (bcoz of it's generate different Machine Code) for every platform
- OS — Win, MAC, Android
- System libraries



C++ → Compiler is a S/W that convert source code into machine code
JAVA → JVM (translator) S/W / program that converts/translate source code into intermediate code and the based on platform compiled into machine code

Source code (hello.java) → compiler / compilation → Byte Code (hello.class) → JVM (Translator) → machine code

(i) Interpreter - Interpret the bytecode instructions line by line into machine code
(ii) Just-in-Time (JIT) Compiler - Integral part of JVM that identifies frequently executed code sections & compiles them into highly optimized native machine code for faster execution

- Platform Independent
- WORA → Write once run anywhere
- Useful in Security → Backend → Servlets
  → Frontend (Client → Web browser) → (Applets (Light weight code) form, displays, calculator) (JDK 11 → discontinued)
- Solve Portable & Security issue using JVM

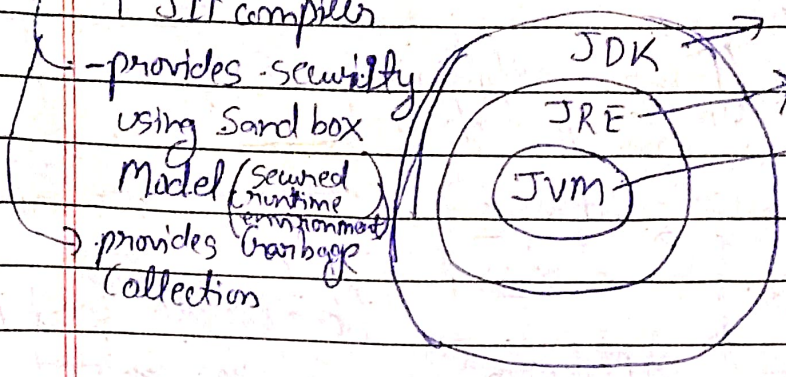
Can C/C++ be platform independent?
Yes, but not in the same way as Java. The reason is mostly about design philosophy and performance trade-offs, not technical limitations.
- But then: - It wouldn't be C++
  - It wouldn't dominate system programming
  - It would lose hardware-level power

JAVA chose safety and portability → portable at binary level
C++ chose control and performance
↳ portable at source level

17/2/26

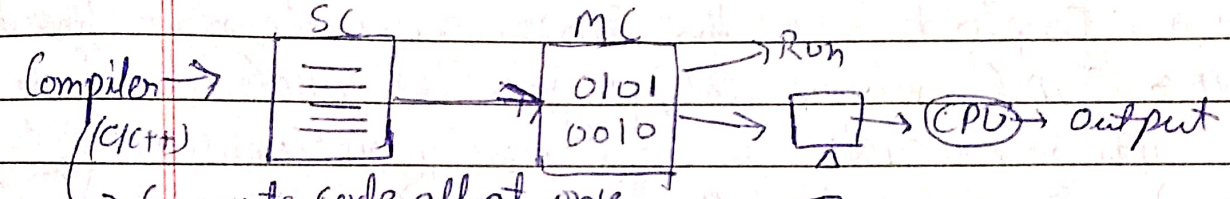
DAY-2

JVM → converts Bytecode into machine code using interpreter + JIT compiler
- provides security using Sand box Model (secured runtime environment)
→ provides Garbage Collection

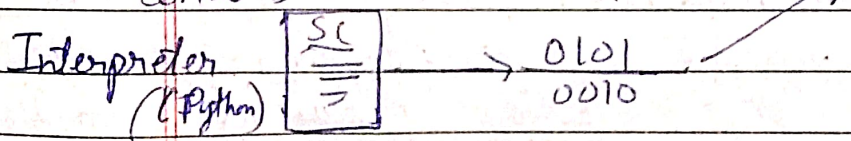


→ Java Development Kit
→ Java Runtime Environment
→ Java Virtual Machine

SC - Source code . MC - Machine code



→ Converts code all at once



→ Converts code line by line

JAVA → Compiled + Interpreted
SC - Bytecode | Bytecode - MC

Earlier problem in JVM
→ H/W slow
→ RAM limited
→ Disc slow

To solve → In JVM included :
Interpreter + JIT compiler
Less important code to MC
Just in time for frequent code to MC

JRE → JVM + Class libraries (consist of func^n)

S/W package that provides the necessary libraries, JVM and other components required to run Java applications on a computer.

- Acts as a layer on top of the operating system, ensuring JAVA programs can run across different platforms

JDK → JRE + Compiler + Debugger + Java Docs
↳ complete package to run JAVA program
↳ Comprehensive S/W package for building, compiling, debugging, and running applications written in the Java programming language.

JSE, JEE, JME

(Core JAVA) JSE (Java Standard Edition) - Can develop stand-alone applications. It provides the following packages -

- java.lang - provides language basics
- java.util - This package provides classes & interfaces (API's) related to collection framework, events, DS and other utility classes such as date.
- java.io - package provides classes & interfaces for file operations, and other I/P & O/P operations
- java.math - provides classes & interfaces for multi precision arithmetics
- java.nio - provides classes & interfaces the Non-blocking I/O framework for Java
- java.net - provides Classes & Interfaces related to networking
- java.security - " " " " such as Key generation, encryption & decryption which belongs to the security framework
- java.sql - provides classes & Interfaces for accessing / manipulating the data stored in DB and data sources
- java.awt - provides classes & interfaces to create GUI components in Java.

- java.text - provides classes & interfaces to handle text, dates, numbers, & messages.
- java.rmi - Provides the RMI package.
- java.time - main API for dates, times, instants, & durations.
- java.beans - package ~~consist~~ contains classes & interfaces related to JavaBeans components

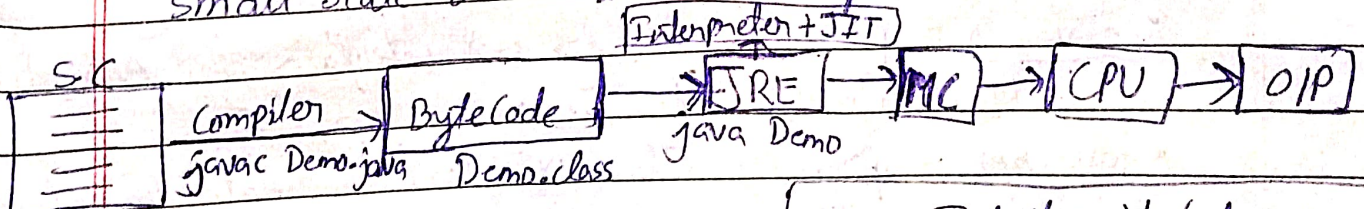
→ JakartaEE

(for Web/ Advanced) JEE (Java Enterprise Edition) - Can develop enterprise applications. This includes :- Transactionals

API's like Servlets, WebSocket, JavaServer Faces, Unified Expression Language.

- Web Service specification - API for Restful web services, API for JSON processing, API for JSON Binding, Architecture for XML binding, API for XML web service
- Enterprise specification - Dependency Injection, Enterprise JavaBean, Java Persistence API, Java Transaction API

JME (Java micro Edition) - Can develop applications that run on small scale devices like mobile phones.



IDE → Intellij, Vscode
Install JDK latest version
Set path variable/environment

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

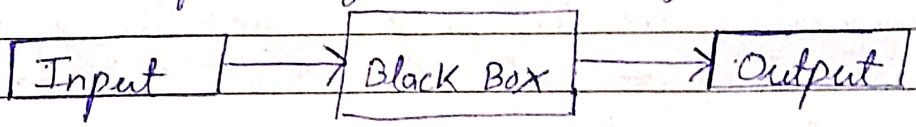
javac Demo.java
java Demo
Terminal → O/P → Hello World!

JAVA is statically typed

13/2/26

DAY-3

- Variable - Container that holds value in memory
  - Naming Convention - Case sensitive, cannot use Keyword or can't start with Number, Can only use underscore & no other special character, Can be any character - uppercase/lowercase, No space between two variable
- Identifiers - Name given to variables, classes, methods, packages, Interfaces Unique names used to identify programming elements. Every variable must be identified with a unique name
- Black box model of learning
  Understanding and using a class or program only through its inputs and outputs, without knowing its internal implementation.
  It comes from the general concept of Black Box Testing



You give input, observe output, Do not see internal logic.
Ex: You insert card, Enter PIN, Get money. You don't know Internal banking system, DB queries, Server communication.

- Data Type - Define the kind of data a variable can hold. It is of two type → primitive & Non-primitive
  - Integer → byte, short, int, long
  - Real no → Float, double
  - Character → char
  - Boolean → boolean

Name →	byte	short	int	long
Width →	8 bits	16 bits	32 bits	64 bits
Range →	-128 to 127	-32768 to 32767	-2147483648 to 2147483647	-9223372036854775808 to 9223372036854775807

** Every number in JAVA is signed (positive & negative)
** Bit - smallest unit of digital information, it can either be 0 or 1.
** Byte - Collection of 8 bits, it can represent values from 0 to 255.

↗ single precision

float - 32 bits — 1.4e-045 to 3.4e+038

double - 64 bits — 4.9e-324 to 1.8e+308

↳ double precision ↳ $4.9 * 10^{-324}$

→ 0-127 (8 bit)

Char → unicode —using→ ASCII (American Standard Code for Information Interchange)

↳ 16 bit in JAVA

A-65 Z-90 a-97 z-122 0-48 9-57

boolean → True / False 0X1X

Literals - Value of boolean, numeric, character, or string data.
Any constant value that can be assigned to the variable

prefix ← 0b (zero) 101 → binary of 5 0x (zero) 5 → hexadecimal of 5
↳ Rang - 0-15

05 → Octal of 5
↳ zero Rang - 0-7

** If there is large number, to make it readable use underscore (_). Ex: long l=34_12_56_789.
You cannot use (_) before & after decimal / exponent

Declaring & Defining at the same time int X=5;
↳ at different time int x;
System.in(x);

Keywords - Reserved word that cannot be used as variable, methods, classes, or any other identifiers..
There are 68 Keywords in Java.

Comments - Can be used to explain code & to make it more readable. Two kinds of comments :-

i) Implementation → delimited by /*...*/ and //

ii) Documentation → delimited by /**...*/

19/2/26

DAY 4

- Problem with floating numbers :- float - 32 bit (less precision 6-7 digit), double - 64 bit (more precision 15-16 digit) (binary format is IEEE 754) Precision problems, Comparision problem (Never compare floating numbers using ==, it may return false), Rounding Errors.

- How negative numbers are stored?

  Stored using a system called 2's complement

  -5 Binary → Write positive 5 in binary (00000101) —1's→ Invert all bits i.e., 0 to 1 and 1 to 0 (11111010) —2's→

  Add 1 to binary no 11111010 + 1 = 11111011 (-5)

  MSB ① 1 1 1 1 0 1 1 ← LSB

  MSB - most significant Bit → LSB - Least significant Bit

  MSB = 1 → -ve

  MSB = 0 → +ve

- How floating points number are stored?

  ← 32-bit →

1 bit	8 bits	23 bits
sign bit	exponent	Mantissa

  8.125

  1000 . 001

  0.125 * 2 = 0.25 → 0

  0.25 * 2 = 0.5 → 0

  0.5 * 2 = 1.0 → 1

  ① Find binary of no (8.125) = 1000.001

  ② Make it in the form of $(1.m) * 2^{exp}$ (m = mantissa) $= 1.000001 * 2^3$

  ③ Add Bias to the exponent → For float bias = 127

  $= 3 + 127 = 130$

  exp = 1000010

2	130	0
2	65	1
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
	1	

  ④ Place value in memory

0	1000010	00000	00000000000000000
		$2^{-1}+2^{-2}+2^{-3}+2^{-6}$	

  How System.out.println(f); will read above from memory & give output 8.125?

  $(-1)^{sign} * (1 + mantissa) * 2^{exp-Bias}$

  $= (-1)^0 * (1 + 2^{-6}) * 2^3$

  $= (1 + \frac{1}{64}) * 8 = (1 + 0.015625) * 8$

  $= 1.015625 * 8$

  $= 8.125$

  exp = 1000010 (2)

  $= 128 + 2 = 130$

  exp - bias = 130 - 127

  $= 3$

Edge case of floating number

float f = 0.7f;

(1) Express in Binary 0.10110 0110 ---

0.7 → 0.10110

$0.7 * 2 = 1.4 \rightarrow 1$
$0.4 * 2 = 0.8 \rightarrow 0$
$0.8 * 2 = 1.6 \rightarrow 1$
$0.6 * 2 = 1.2 \rightarrow 1$
$0.2 * 2 = 0.4 \rightarrow 0$
$0.4 * 2 = 0.8 \rightarrow 0$

(2) $(1.xx * 2^{exp})$

$(1.0110\ 0110 \ldots) * 2^{-1}$

(3) Add Bias $= -1 + 127 = 126 = 01111110$

(4) Express in memory

0	01111110	0110 0110 0110 0110 0110 011

(bit positions marked: -2 -3, -6 -7, -10 -11, -14 -15, -18 -19, -22 -23)

→ In memory we are storing truncated data/value that's why we are not getting correct output

System.out.println(f);

$(-1)^{sign} * (1+m) * 2^{e-B}$

$= (-1)^0 * (1 + \text{—}) * 2^{126-127}$

$= (1 + \text{—}) * 2^{-1}$

$= \frac{1}{2} * \left(1 + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^6} + \frac{1}{2^7} + \frac{1}{2^{10}} + \frac{1}{2^{11}} + \frac{1}{2^{14}} + \frac{1}{2^{15}} + \frac{1}{2^{18}} + \frac{1}{2^{19}} + \frac{1}{2^{22}} + \frac{1}{2^{23}}\right)$

$= 1.39999997615814208964875 * \frac{1}{2}$

$= 0.69999998807907104421875$

Bias → In floating point number it stores 32 bit that follows IEEE standard which states no negative number can be stored so it uses bias = 127 (float) to make positive

Exponent = 8 bit = 0 − 255 character representation
↳ center point = 127

$= 2^{8-1} - 1$

$= 2^7 - 1 = 128 - 1 = 127$

$\boxed{Bias = 2^{exp-1} - 1}$

- Expressing double in memory — double = 64 bit

← 64 bit →

1 bit	11 bit	52 bit
sign bit	exponent	mantisa

$Bias = 2^{11-1} - 1 = 2^{10} - 1$

Bias = 1023 (double)

Similar to float, double can also not store some value as it is (0.7). So Java came up with concept of BigDecimal → gives exact value not estimated.

20/2/26

DAY 5 Type Conversion

WEEK-1

Data Type (Integers, Real, Character, Boolean)

- Type Conversion means converting one data type into another.
  - Ex :- int → double, double → int
  - Two types
    - byte → short → int → long → float → double
    - i) Widening (Implicit) — Rule: Done automatically, smaller → Larger type, No data loss
    - ii) Narrowing (Explicit) — Rule: Done manually, Data may be lost, larger type → smaller type

Ex :-

Implicit

```
byte b = 24;
int i = b;
System.out.println(i);
```

O/P :- 24

Explicit

```
int i = 300;
byte b;
// b = i; X
b = (byte) i; ✓
System.out.println(b);
```

O/P :- 44

(Truncated data)

- If you don't want to convert to binary use :-

  b = 300 % (Range of byte) $2^8 = 256$

  b = 300 % 256

  b = 44

Type casting → done in narrowing/explicit using the type which is required in front

- Truncating Conversion — type of narrowing conversion where higher precision data is converted to lower precision, causing loss of info. double → float, double → int, long → int, int → short/byte

Truncation → Cuts off extra data // Rounding → Adjusts to nearest value

- Automatic Type promotion — mechanism where a smaller-sized data type is automatically converted to a larger-sized data type to prevent data loss and ensure expression consistency.

Ex :-

```
bool a = 50;
bool b = 40;
bool c = 100;
int i = (a*b)/c; // 200
```

Rules :- i) All byte, short, char operands are promoted to int (ii) If one of the operand is long, entire expression is promoted to long.
(iii) One operand float → entire exp float
(iv) One " double → " " double