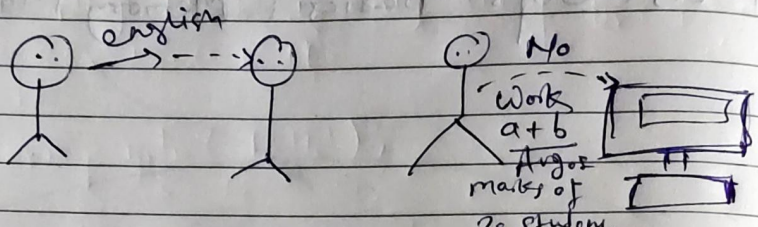
6. Beginner_Logical Thinking-2

27/08/24

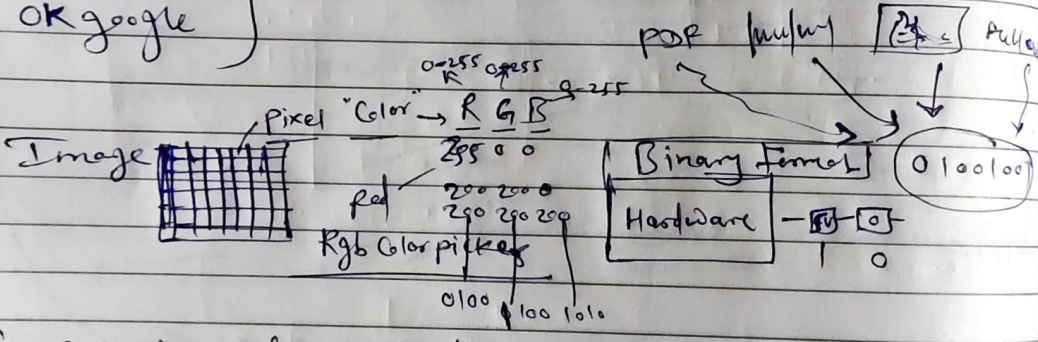
- Programming Languages
- Java
  - Input / Output
  - Variables / Datatypes
  - Operators
  - Type casting.

- Programming Language.



(It's not understand what to do)

Alexa
Siri
OK google
} Another layer added to understand english

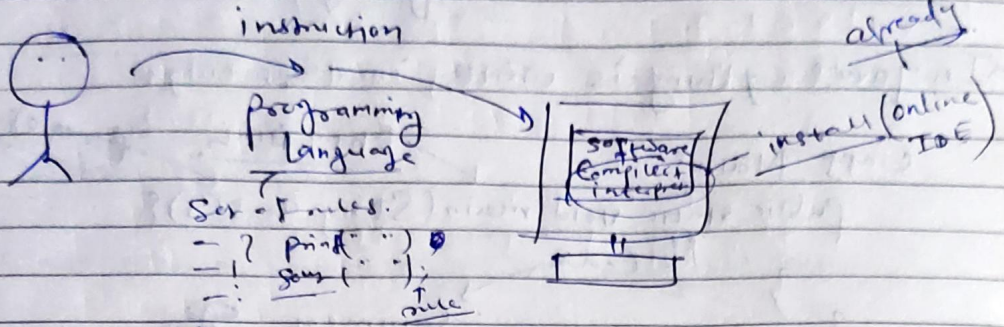


- Computer is made up of Hardware & it's made up of small small Charge registers.
- Inside memory the state is binary (0 / 1).
- Hardware can only store 0 & 1.

ASCII Table. Hello World

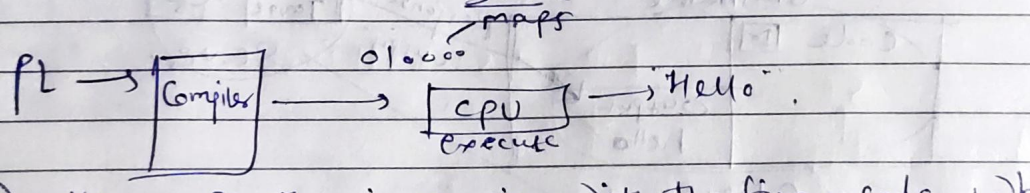
A → (65)

- Everything is stored inside Computer is only in 0's & 1's format (binary).



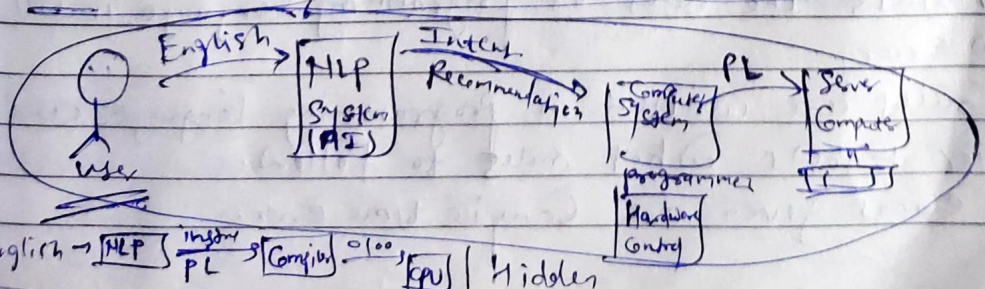
- To understand the instructions to the Compiler there is one Software called Compiler.
- Compiler is specific to every programming language.
- Some have Compiler and interpreter.
- These Software knows how to Convert ~~Code~~ Program/am into binary code.

- There is a grammer or rules associated with every programming language.
- Every instruction is terminate with the semicolon.



- Compiler maps the instruction with the binary code. Which Computer hardware can execute.

- Some PL need only Compiler or Some needs only interpreter or some needs both.
- Computer has the Software called Compiler which knows how to interpret rules of the programming language and Converted into the binary code that maching code.



→ interviewbit ide → ide.new / java / ide.new

⇒ In Java, everything is written in class only.

```
class Main{
public static void main(String args[]){
    // Code
}
}
```

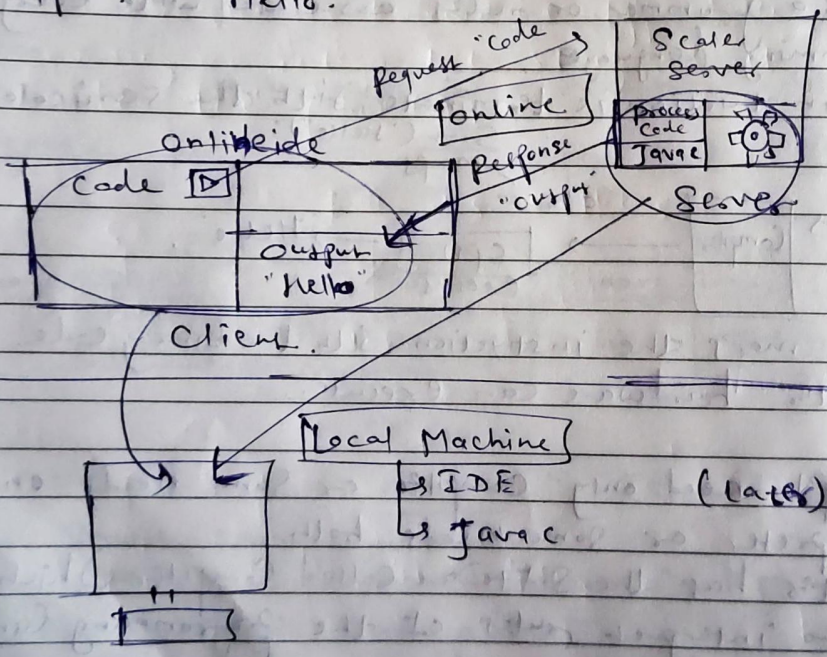
Entry point. Array of Strings

Comment

- All program, start executing from main().
- Comment is Any that is only for programmer's refer and Compiler ignores it.

System.out.println("Hello");

o/p :- Hello.



- If violates the rules of programming language, it will throw error.
- This rules are written in programming language, and Compiler knows what rules to follow.
- Such errors given are Compile time error.

Compile Time Error

↳ Violates the rule of programming language.

System → class.

- Class Names should begin with a capital letter.

→ 3 classes are present, all begin with Capital letter.

line break.

# System.out.println("Hellow");
System.out.print("Hellow");

print() vs println()

Hello "Hello"

Inside class

methodn() ⇒ Method / Function (outside class) ("Data") → same

System.out.print("Hellow");

↓

Some code behind the print functionality.

* String :- Sequence of characters.

"Hello";
"123";

* →

System.out.print(156) → Numbe (Integer);
System.out.println(156.25) → Numbe (Double);
System.out.println(156.25f) → Numbe (Float);

System.out.println(156+7); // Integer. → 163
System.out.println("156"+7); // String → 1567

156+7 → Addition

"156"+"7" → Join two Strings (Concatenation).

156 + 7 → Addition
"156" + "7" → Join two Strings (Concatenation)

Operator is same but type of data is different.

System.out.println("156" + 7) String + Integer →

"156" + 7 → "156" + "7" → "1567".
logic behind Print Funct.

Automatically → Implicit Typecasting

- This conversion is known as typecasting, It Converting an integer into String & it happens behinds the print().

"Hello" + 5 → Hello5

"56"
↑
56

← not work
"int"

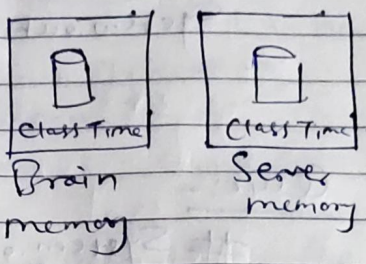
int   int
5 (+) 3
Operand1   Operator   Operand2.

Note: String is not primitive
Not Basic it's Advance Data

- Variables & Data Types :-

→ Store Data

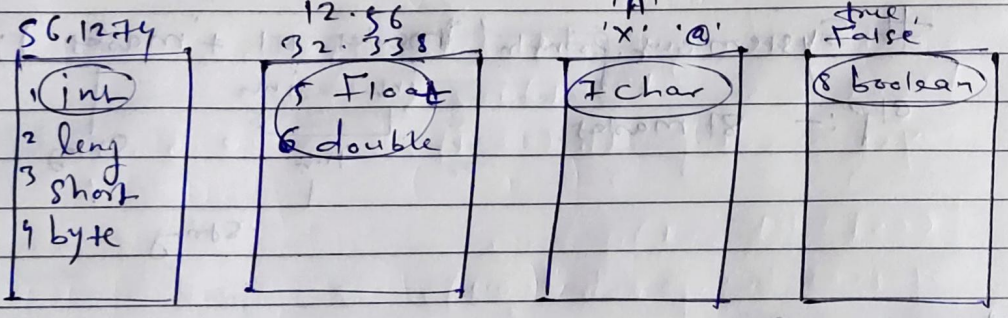
→ Bucket in the memory (RAM)
which store data of certain type.



→ 8 Basic Type of data (primitive)
(Simplest type of data) → Can't be broken down Further

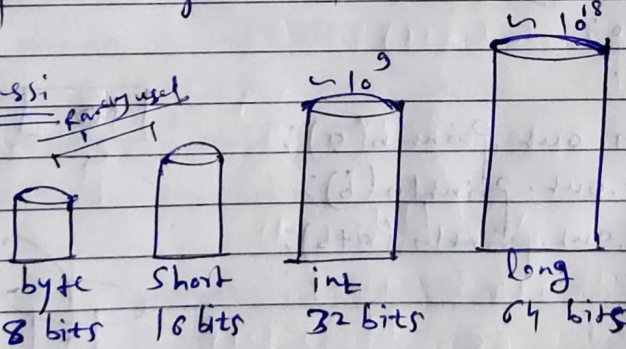
~~Int~~ ~~Float~~ ~~double~~ ~~long~~ ~~Char~~

~~56,127~~ ~~12.56f~~

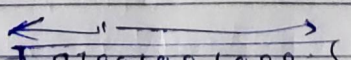


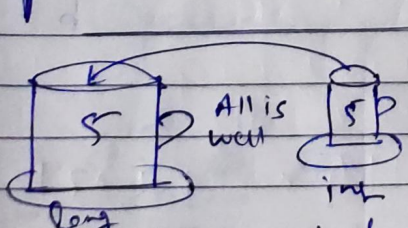
- We require storage based on the data.

- e.g.. lassi

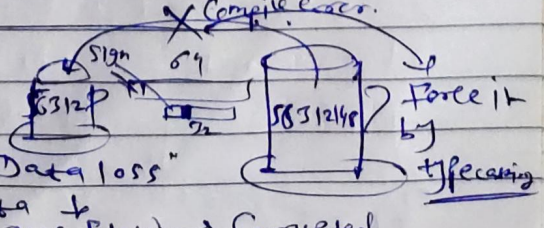


It will save the space





More space than required.



"Data loss"

Data + Overflow → Corrupted

int Datatype — bucket.(variable)

int marks = 80;

System.out.println("marks" + marks);

"marks" → String

No "quotes" here → int

String + int → String

Quiz:-

System.out.println("marks" + marks + 1);

o/p :- ? Ans ⟹ marks 801

String + int → String; String + int → String

BODMAS

System.out.println("marks" + (marks + 1));

o/p :- marks 81

String + (int + int)

String + int → String

System.out.println(marks + 1 + "marks");

o/p :- 81 marks

int + int → int

int + String → String

String + String → String

~~# Datatype~~ — Assignment operator

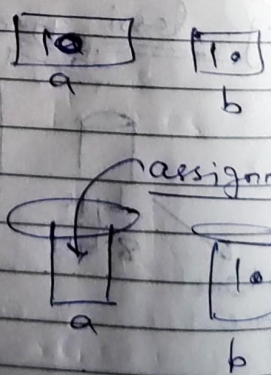
int a = 10;

int b = a;

System.out.println(a);

System.out.println(b);

System.out.println(a+b);

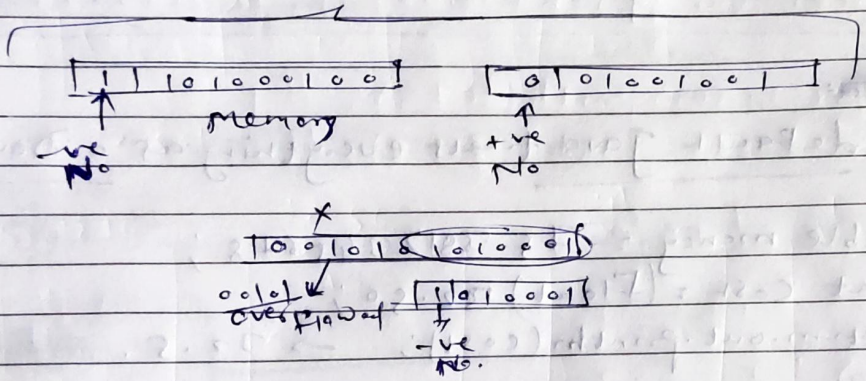


* Java by default treats the all no's without L as integers.

Long Distance → Should not Convert long to int to avoid overflow

```
long distance = 12345678912345678912L; // L at last.
int dist = (int) distance; // Explicit Typecasting.
System.out.println(distance); → as it is.
System.out.println(dist); → -10--- Corrupted
```

- Long numbers should terminate with the L as per the rule of Java.

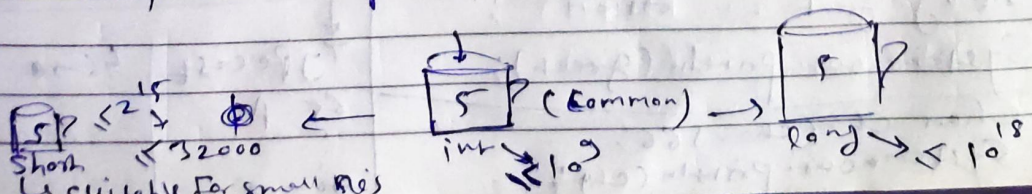


Short

```
Short red_pixel = 200;
Short age = 29;
int money = 1084378;
System.out.println(red_pixel);
System.out.println(age);
System.out.println(money);
```

byte → Ideally we not used it frequently.

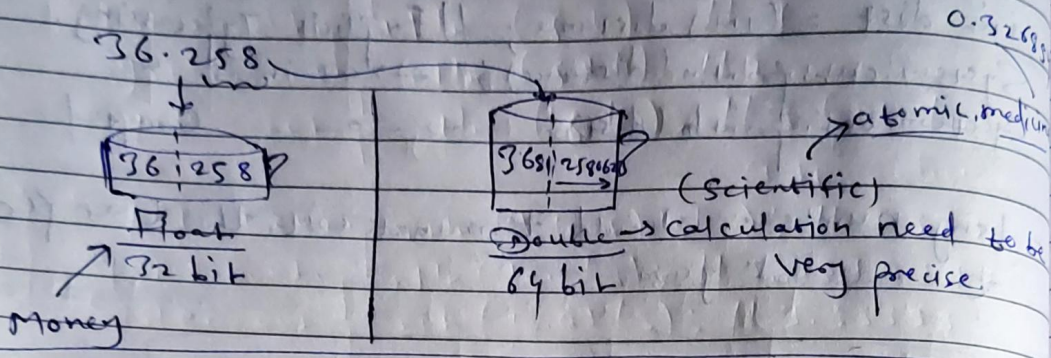
```
byte age = 29;
System.out.println(age);
```



Floats & Doubles

↓

More precision



// By default Java treats everything as a Double.

```
double money = 75.38123456 78;
Float cost = (Float) 32.50;
System.out.println(cost);  → 32.5.

cost = (Float) price;
System.out.println(cost);

Float cost = 32.5F; // Should be end with F.
System.out.println(cost);
```

Char

// 8 bits in c++

datatype

```
char letter = 'A'; // 16 bits
System.out.println(letter);
int number = 65; // 32 bits
System.out.println(number);
System.out.println((int) letter);
```

Any letter

Char

A → 65

→ Character to int.

```
String greet = "नमस्ते";
System.out.println(greet);
Char ch2=(char) 66;
System.out.println(ch2);
```

Typecast

65 - A

46 → .

```
ch = (char) 97 ; // a, b, c ...
System.out.println(ch);
System.out.println((int) 'A');
```

→ Char letter = 'A'; // char are written inside single quotes
// Char l = "A"; ??? Compile time error.

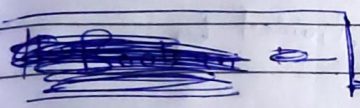
// Add char :- Integer addition.

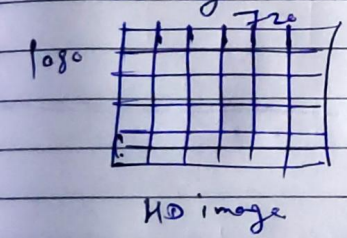
System.out.println('A' + 'B'); :- 131.
int addition

'+' → [Join String].

// Add string

System.out.println("A" + "B"); :- AB.





10,000 images.

Show image = 4 MB; → 4GB (16 bit per 100GB)

Long image = 16 MB; → 16GB 64 bit value

16.96 overbill

Boolean true or false :—

```
boolean x; // creating variable
x = true; // Assign a value of later.
boolean y = false; // Initialization (creating + Assign value)
Sout (x);
Sout (y);
```

* operators *

Arithmetic operators (+, -, *, /, %) :-

```
System.out.println(5+4); //9
System.out.println(5-4); //1
System.out.println(5*4); //20
System.out.println(10/3); //3 (int/int => Integer division)
```

```
System.out.println((float)10/3); //3.33333
System.out.println(10.0/3); //3.3333335
System.out.println(10.0f/3); //3.33333
```

Same

```
System.out.println(10%3); //1
System.out.println(17%3); //2.
```

% modulo

a % b Remainder when a is divided by

5 % 3 ⟶ 2
17 % 2 ⟶ 1
17 % 4 ⟶ 1
30 % 5 ⟶ 0

Input (assignment) :-

// Build a Scanner object + Use it.
↓
(OOPs)

```
import java.util.Scanner;
Scanner sc = new Scanner(System.in); //magic  (keyboard input)
// Use Scanner to read input data
int x = sc.nextInt();
System.out.println(x);
```

```
int x1 = sc.nextInt();
int x2 = sc.nextInt();
System.out.println(x1+x2);

long myMoney = sc.nextLong();
Sout(myMoney);
```

String i/p :-

```
String s1 = sc.next();   → one string
String s2 = sc.nextLine();   → for more word with space
Sout(s1);
Sout(s2);
```

Sc.nextLine()