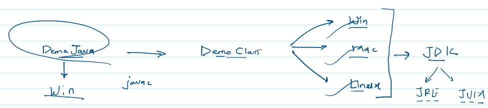
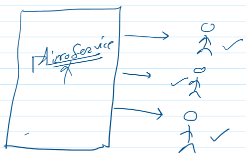
13 May 2025 05:32 PM

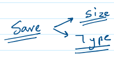
Unit - 01.

Higher -> LL → Java, Python, JS (Human)
Middel -> LL → C lang [%d %f]
Lower -> LL → Ambedded/Amssembly (Machine)

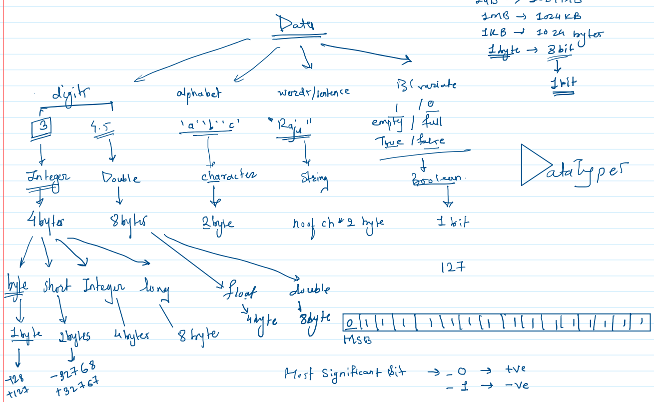




robust



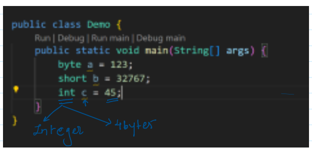
1GB → 1024 MB
1MB → 1024 KB
1KB → 1024 bytes
1 byte → 8 bit
↓
1 bit



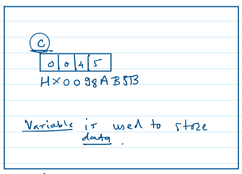
DataTypes

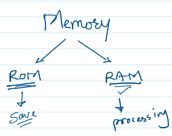
Most Significant Bit → - 0 → +ve
- 1 → -ve

RAM >> Program Memory.




```
public class Demo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        byte a = 123;
        short b = 32767;
        int c = 45;
    }
}
```



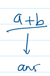


Variable ✓

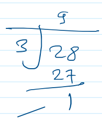
* A name of memory space which use to store and manage data is known as Variable

Operator:

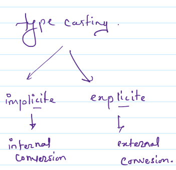
Addition Operation
a, b - operand
+ - operator



1. Arithmatic → +, -, *, /, % (Mod)
2. Assignment → =, +=, -=, *=, /=, %=

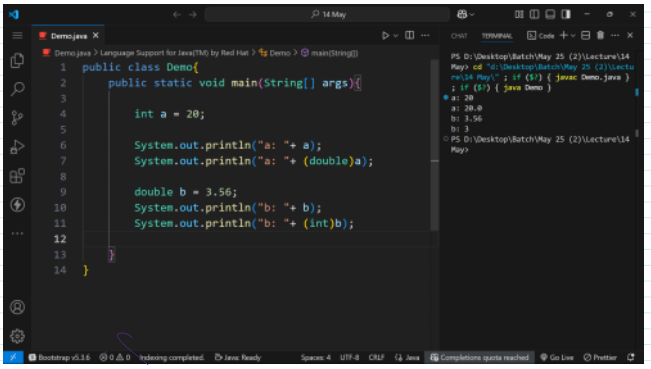


type casting.

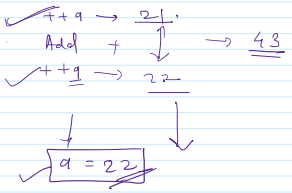


double ans = 4/2.5
= 4.0/2.5

implicite.

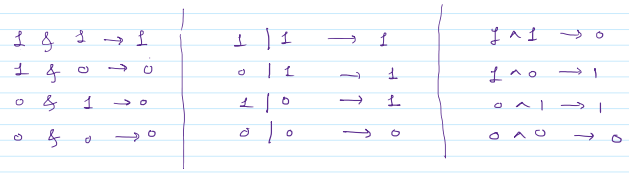



```
int a = 20;

// ++a; // pre increament
// a++; // post increament

System.out.println(++a + ++a);
System.out.println("a: "+ a);
```



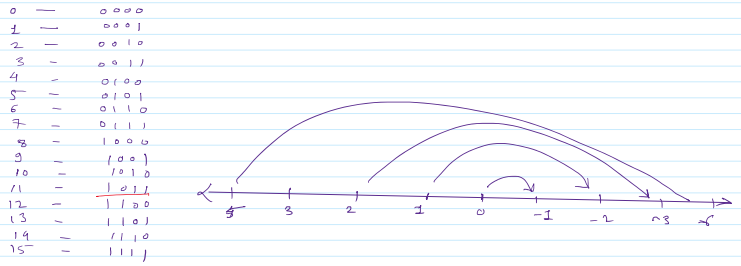
a = ~~20~~ ~~21~~ 22

Bitwise Operator : & | ^ ~ << >>

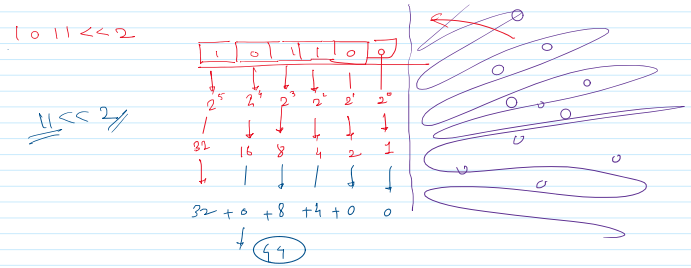


1 & 1 → 1
otherwise → 0

0 | 0 → 0
otherwise → 1

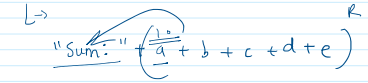


<< >>



( )

Operators	Associativity	Type
++ --	Right to left	Unary postfix
++ -- + - ~ ! (type)	Right to left	Unary prefix
* / %	Left to right	Multiplicative
+ -	Left to right	Additive
<< >> >>>	Left to right	Shift
< <= > >=	Left to right	Relational
== !=	Left to right	Equality
&	Left to right	Boolean Logical AND
^	Left to right	Boolean Logical Exclusive OR
|	Left to right	Boolean Logical Inclusive OR
&&	Left to right	Conditional AND



Operators	Associativity	Type
++ --	Right to left	Unary postfix
++ -- + - ~ ! (type)	Right to left	Unary prefix
* / %	Left to right	Multiplicative
+ -	Left to right	Additive
<< >> >>>	Left to right	Shift
< <= > >=	Left to right	Relational
== !=	Left to right	Equality
&	Left to right	Boolean Logical AND
^	Left to right	Boolean Logical Exclusive OR
|	Left to right	Boolean Logical Inclusive OR
&&	Left to right	Conditional AND
||	Left to right	Conditional OR
?:	Right to left	Conditional
= += -= *= /= %=	Right to left	Assignment

"Sum: " + (a + b + c + d + e)

"Sum: " + "10" + b + c + d + e

"Sum: 10" + b

```java
public class Demo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {

        int a = 10;
        int b = 34;
        int c = 35;
        int d = 23;
        int e = 94;
        System.out.println("Sum: " + (a + b + c + d + e));

    }
}
```

```
May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
12  2
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
Sum: 1034352394
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
Sum: 196
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May>
```

```java
public class Demo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {

        int a = 10;
        int b = 34;
        int c = 35;
        int d = 23;
        int e = 94;
        int f = a + b + c + d + e;
        System.out.println("xyz: " + f);

    }
}
```

```
May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
12  2
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
f: 196
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
xyz: 196
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May>
```

Demo.java

```java
public class Demo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {

        int n = 5;

        int sq = n*n;
        int cube = n*n*n;
        double sqrt = Math.sqrt(n);

        System.out.println("Square: "+ sq);
        System.out.println("Cube: " + cube);
        System.out.println("Square root: " + sqrt);

    }
}
```

```
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May> cd "d:\Desktop\Batch\May 25 (2)\Lecture\14 May\" ; if ($?) { javac Demo.java } ; if ($?) { java Demo }
Square: 25
Cube: 125
Square root: 2.23606797749979
PS D:\Desktop\Batch\May 25 (2)\Lecture\14 May>
```