Java

Programming Language - medium of communication between human and machine

Types of PL

low - language understandable my machine. Ex - binary(0,1)

mid - which has a few keywords called mnemonics. Machine understands them using

assembler - it is a software that converts mnemonics into machine understandable language.

high - python, Java, JavaScript

only run on Mac OS not on windows or Linux.

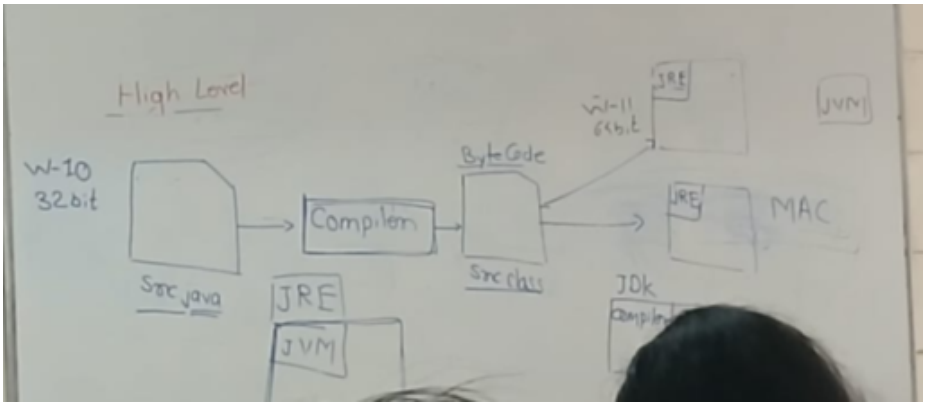
src.java --- compilation --- **byte code** (src.class)--- jvm(Java virtual machine) --- **0101**

JRE - java runtime enviornment(jvm will already be present and some libraries)

JDK - Java development kit(compiler, libraries and JRE)

Java is both complied and interpreted

when we want to make an application we need both compiler and JRE so we download a JDK



types - number , char(len=1), string and boolean(true and false)

int num = 10+2 , this is declaration that will create a memory block

num 's value is 12

variable - named block of memory

declaring variable - datatype VariableName;

declare and initialize variable - datatype VariableName = value;

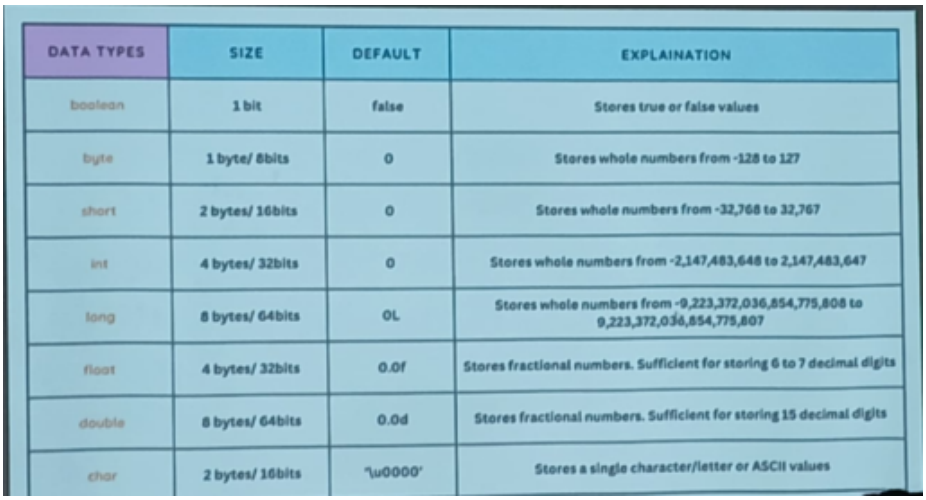
```
int a = 10;
a = 20; //re-initialization
```

```
int a  =  30; // not possible as a can't be declared again
```

DATA TYPE

Primitive - byte, short, int, long, float, double - store numbers

char,Boolean



DATA TYPES	SIZE	DEFAULT	EXPLAINATION
boolean	1 bit	false	Stores true or false values
byte	1 byte/ 8bits	0	Stores whole numbers from -128 to 127
short	2 bytes/ 16bits	0	Stores whole numbers from -32,768 to 32,767
int	4 bytes/ 32bits	0	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes/ 64bits	0L	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes/ 32bits	0.0f	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes/ 64bits	0.0d	Stores fractional numbers. Sufficient for storing 15 decimal digits
char	2 bytes/ 16bits	'\u0000'	Stores a single character/letter or ASCII values

Boolean size is not defined, it depends upon the system.

true and false are literals not keyowrds(as per oracle documentation)

Non-Primitive data type

user defined data type - class, interface, array

Can we create a variable using a non-primitive data type ? Yes

```
class obj;
```

ex-

```
Demo a;
```

values that can be stored in **a** are null and object of **demo class**

- Use of non primitive data type - create non primitive variable
- Use of non primitive variable - to store a non primitive value

Types of Variables

Based on scope

- static
- not static
- local -the variable created inside any other block than the class block

Based on data type

- primitive
- non primitive

In java we do not have global variable

Characteristics of local variable

- we can access local variable only inside its own block
- we values are not considered

Operators

Precedence	Operator	Type	Associativity
15	() [] .	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ! ~ (*type*)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right
11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

operators are predefined symbols that have predefined tasks

10 + 2 = 12

every time operator will return you the result

Precedence - it is priority of execution

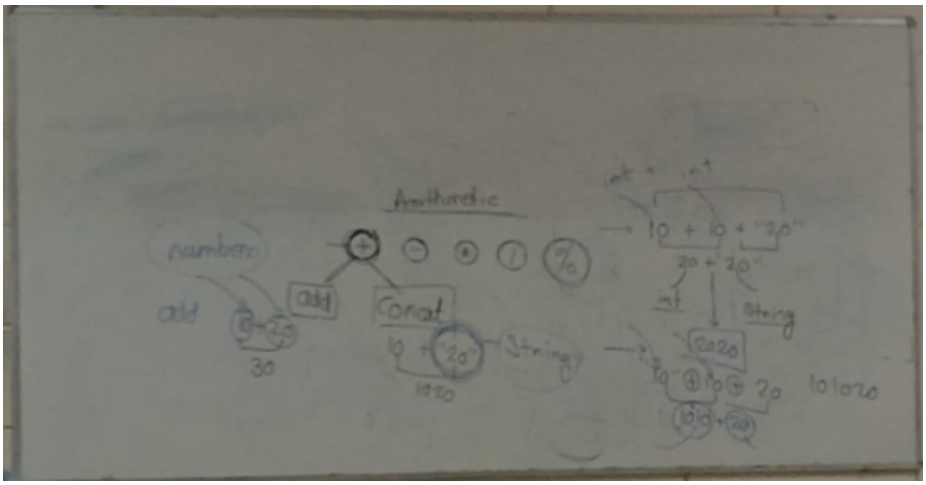
Associativity - direction of execution(used when we have operators with same precedence)

most common left to right

Types of Operators

Arithmetic

- + (add numbers and concat string) , number + Boolean is compile time error, string + Boolean is ok
- - string - number gives compile time error
- * - Gives ASCII code of any char ex "A"*1 = 65
- /
- %



Relational Operator

- >
- <
- >=
- <=
- ==
- !=

It gives Boolean results.

LHS should be variable.

println - the next print command gets printed in the next line

print - the value gets printed in the same line

example:

```
System.out.println(20);

System.out.print("Hi ");

System.out.println("Bye");
```

Output

```
20

Hi Bye
```

DAY 3

System.out.println()

System - class

out- variable

println - method

Logical Operator

- && - AND operator
- || - OR operator
- | - NOT operator

return Boolean value

Conditional Operator(Ternary Operator)

condition?exp/value:exp/value

: True : False

takes 3 operands: 1st - condition

2nd - if condition is True then it will get executed

3rd - if condition is False then it will get executed

Example:

```
System.out.println(10>20?:"Hi":"Bye")
```

Output

```
Bye
```

Increment Operator - "++"

- pre -increment: ++a
- post-increment: a++

```
int num1 = 40;
System.out.println(num1++);
System.out.println(num1);
System.out.println(++num1);
Output
40
41
42
```

```
int a =10;
int b = a++;
int c = ++a;
```

Output

```
12
10
12
```

```
int a = 30
int b = ++a+a  //62
int c = --a //30
c++;
a = b--; //62
```

Output

```
a= 62
b = 61
c = 31
```

```
a = 13;
```

```
c = a+++a;  // c =27 a = 14

b = --a ; //b = 13

a--; //a = 12
```

Output

```
12

13

27
```

```
int a = 10;

a = a++;

a = a++;

a = a++;

system.out.println(a)
```

Output

```
10
```

We cannot use increment operator with constants, only with variables.

```
byte u = 127;

u++;

System.out.println(u);
```

Output

```
-128
```

Decrement Operator - "--"

- pre -decrement: --a
- post-decrement: a--

works same as increment operator

```
int a = 10;

a = a--;

a = a--;

system.out.println(a)
```

Output

```
10 // beacuse of = the value is assihned from RHS to LHS and the a gets reassigned to 10 again  and again
```

Compound Assignment operator

- +=
- -=
- /=
- *=
- %=

```
byte s = 10;

byte t = 20;

s+=t;  // this will not give error beacuse the compound operator will cast the int into byte

System.out.println(s);
```

Output

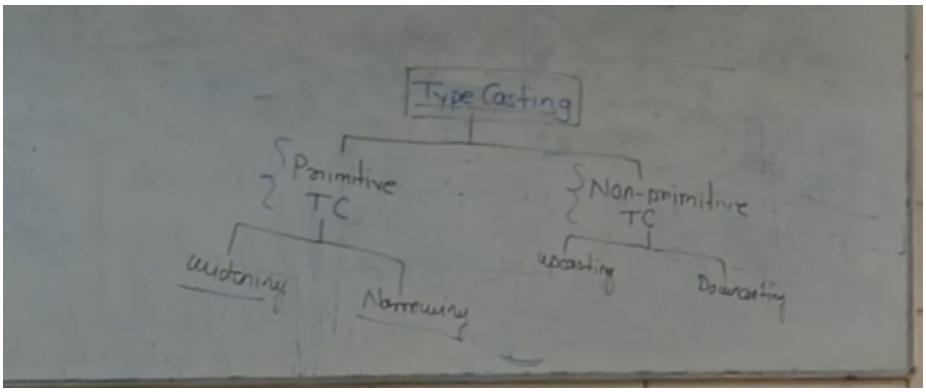
```
30
```

```
string s = "hi"

s+=10;

System.out.println(s)
```

Output

```
hi10
```

Type Casting



- Primitive TC - Widening Narrowing
- Non-primitive TC - Upcasting Downcasting

Widening

ex- int -> double

```
double d = 21;

System.out.println(d)
```

Output

```
21.0
```

Narrowing

Converting large data type into small data type

```
double d = 21;

System.out.println(d);

int i = d; // narrowing cannot be done by compiler

System.out.println(i);
```

Output

```
error

Type mismatch: cannot convert from double to int
```

cast operator allows type conversion

```
int i = (int)21.21;

System.out.println(i);
```

Output

```
21
```

```
int i = (int)true;
```

Output

```
Cannot cast from boolean to int
```

```
boolean b = (boolean) 1;
```

Output

```
Cannot cast from int to boolean
```

Note- we can convert primitive data type into non-primitive data type and vice versa

Method/Function

Method is a set of code or block of instructions which is used to perform a particular task.

[modifiers] return-type name([Formal arguments]){

[return statement]

}

modifiers, formal arguments and return statement are optional.

Modifiers

All the modifies are keywords. They will modify the behavior of a member or methods.

Types

1. **Access Modifiers**

- Public - use that method anywhere inside the project
- Private - access that method only inside the class

  we can access private members outside the class using reflection api
- Protected - access that method inside the package

  can access protected method outside package using inheritance
- Default - package

We cannot create a method with a default keyword or modifier inside a class.

2. Non-Access modifiers

- Static
- Final
- Abstartct
- Native - these are written in C/C++ , we can use this in java using java native interface
- Synchronized

Return type

- void - keyword
- Primitive Datatype
- Non-primitive data type

It will define what type of data method will return after execution.

If void is written then the method will not return anything.

name + formal argument = method signature

modifies + return type+ method signature = method declaration

method declaration + method block = method definition

main method is called by JVM

DAY-4

Types of Methods

1. Number formal Argument

- No argument - which do not have any formal argument

  example: public static void m1(){ ------}
- parameterized
- Variable argument

Length and data type of formal and actual argument should be the same.

Formal arguments are local variables for that particular method.

2. Modifiers

- Static
- non-static
- final
- native
- synchronized

whenever method with the same data type is not present inside the class, then we will go for type promotion.

(searching for data with higher data type in formal arguments)

inside a printing statement we can call a method which will return some data

Return Data Type

we can return data using **return** keyword

it will return the control inside the calling method

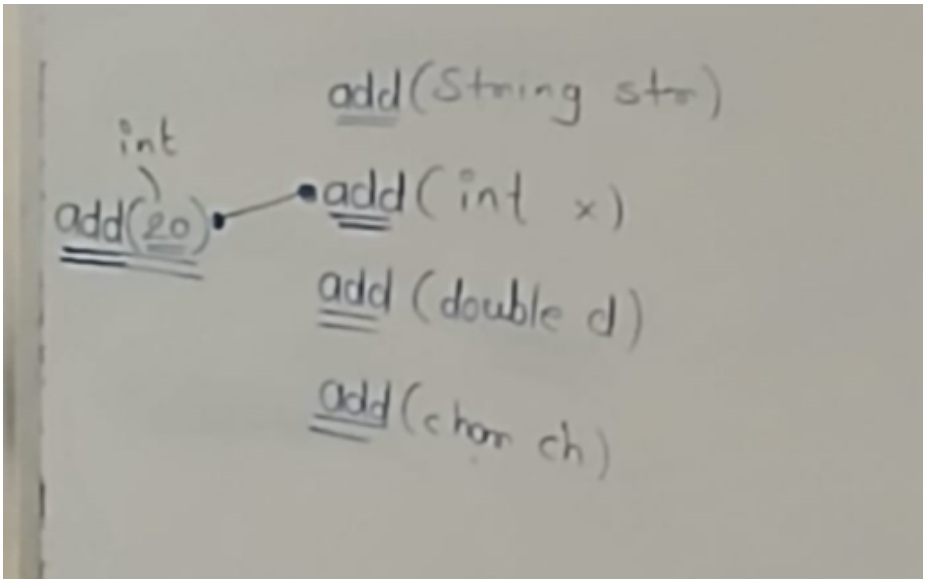
if the return type is other than void, then return statement is mandatory

return statement should be the last statement inside the block.

Method Overloading

If inside a class we have method with the same name and different formal arguments.

Also known as compile time binding or early time binding.



Compiler will bind the method called with method declaration during compile time. This happens during the early stage of program.

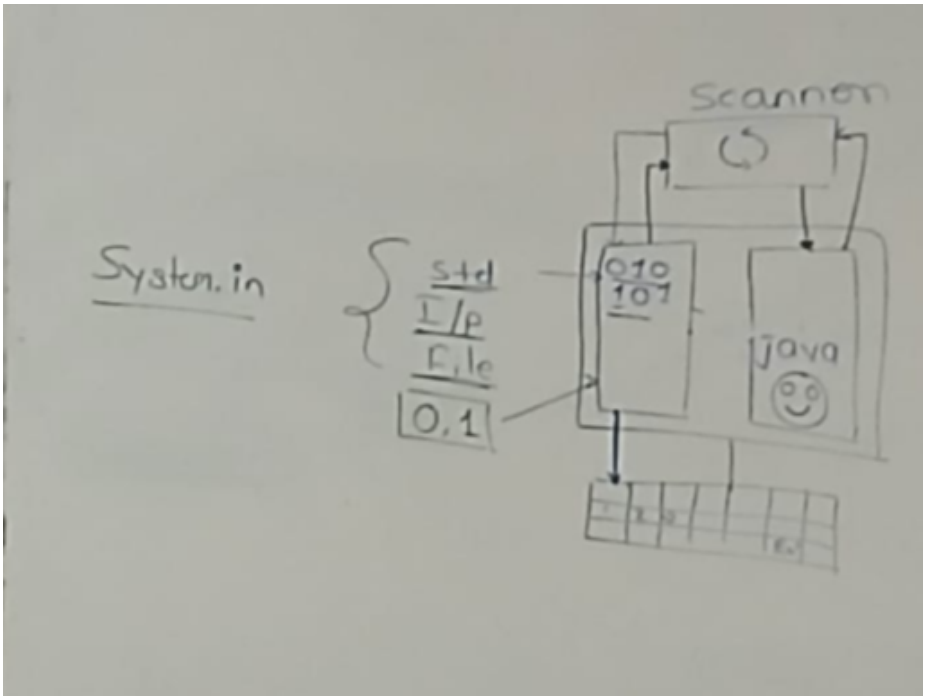
How to perform method overloading?

->Method name should be same and formal arguments should be different.

-> formal arguments can differ in three ways:

- length
- type
- sequence

System.in



The input given from the keyboard is in the form of binary which java cannot understand, so it uses a utility class called **Scanner** which converts the binary into Java understandable language.

Scanner class is present inside **util** package

```
import java.util.Scanner;

Scanner ip = new Scanner(System.in); //object of Scanner class

ip.nextInt() // for taking int as input

ip.next()      // both for String input

ip.nextLine()

ip.nextBoolean()// not case sensitive in terms of input
```

whatever classes that are present in lang package will automatically get imported. They are already there in every file.

```
import java.util.Scanner;
public class Input {
   public static void main(String[] args) {
        Scanner ip = new Scanner(System.in);
        System.out.println("Enter a num:");
        int num = ip.nextInt();
        System.out.println(num);
   }
}
```

Output

```
Enter a num:
23
```

23

DAY-5

next() - only takes the first word of the string as input

nextLine() - takes the whole string as input

hasNext() - gives Boolean data. return true if next element is present otherwise return false

```
String str = "10 20 30";
Scanner sc = new Scanner(str);
System.out.println(sc.next());
System.out.println(sc.hasNext());
System.out.println(sc.next());
System.out.println(sc.hasNext());
System.out.println(sc.next());
System.out.println(sc.hasNext());
```

Output

```
10
true
20
true
30
false
```

useDelimiter("Pattern"); - specify the limitter or seperator in string while using next() .

```
public static void main(String[] a) {
   String str = "10:20:30";
   Scanner sc = new Scanner(str);
   sc.useDelimiter(":");
   System.out.println(sc.next());
   System.out.println(sc.hasNext());
   System.out.println(sc.next());
   System.out.println(sc.hasNext());
```

```
   System.out.println(sc.next());

   System.out.println(sc.hasNext());

}
```

Output

```
10

true

20

true

30

false
```

Static Method

Method created using static keyword.

How to call a static method inside the same class?

- Directly
- with the help of method name as reference

to call from different class -> **classname.method();**