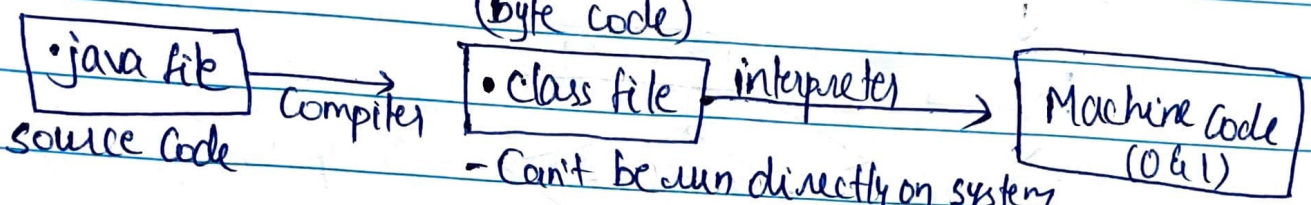
3. Java

Architecture & Installation

→ How Java code executes

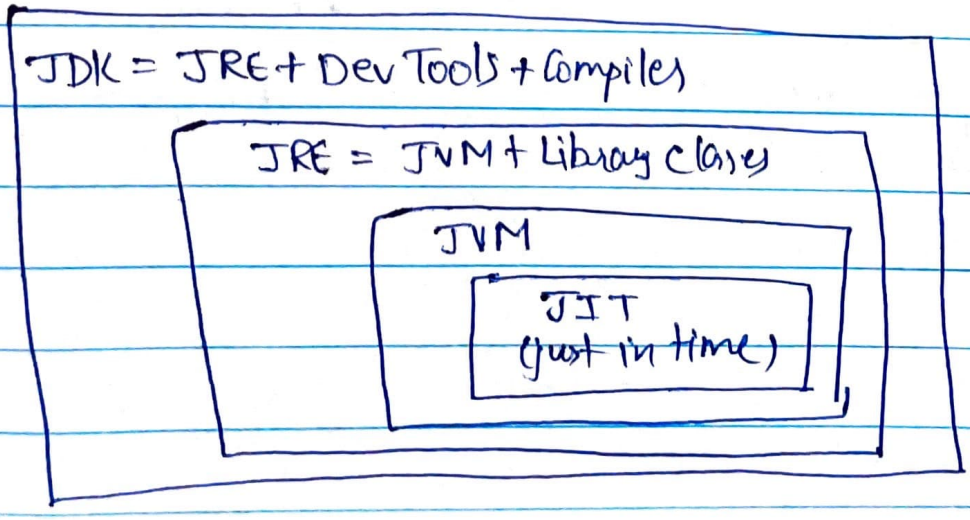


- Can't be run directly on system
- we need Java virtual Machine
- this is why Java is platform independent

Platform independence :-

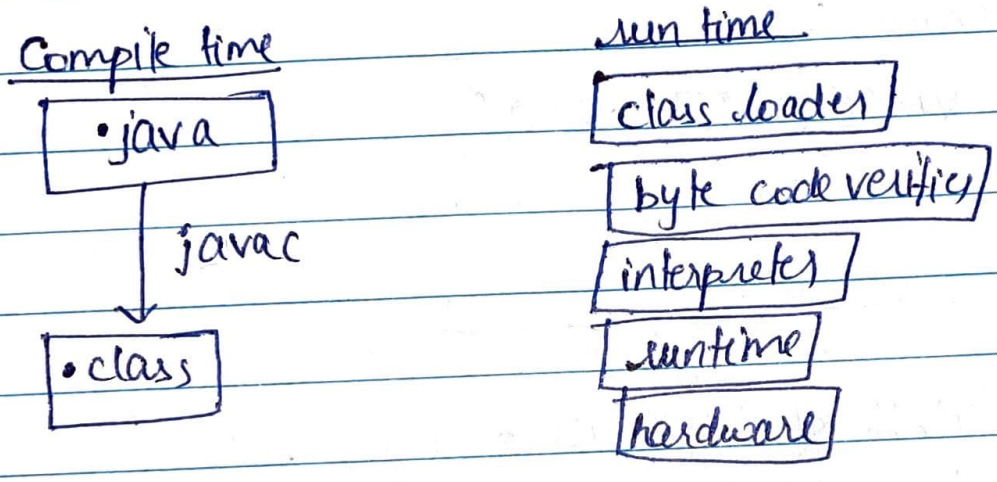
- byte code can run on all OS. We've to convert source code to machine code so computer can understand. In Java, we get byte code which JVM converts to machine code. Java is platform independent but JVM is.

Architecture :-



JDK → provides environment to develop and run Java program
→ includes 'javac', interpreter + loader.

JRE → installation package for running program



JVM → Contains stack & heap memory allocations

working :-

Loading : reads .class & generates binary data
an object is created in heap

Linking :

JVM verifies .class
allocates memory for class variables

4. Java program

→ Everything we write in .java file is a class itself.
named group of properties & functions

```
public
class <Filename>{
        psvm(String[] args){
        }
}
```

① Filename should be starting with an uppercase

② 'public class' → means making this class be accessable by anyone anywhere

③ function is collection of code. So, we need a main function from which program starts it is PSVM

```
public class Hello {
    public static void main(String[] args){
      Sout println("Hello world");
    }
}
```

main function is entry point of program.

if the code is main, it has to be run from anywhere, so public.
static means, we access function/variables etc when we have objects of that class, to ~~create a variable~~ run program without creating object of class. As variables & functions

java -d .

clonot depend on objed, they are static. we create a main function without creating class Demo.
an object of

void is return type of value. we dont want it to give any value in return

String[] args → Command line arguments. This is array, Collection of strings.
⇒ whatever input we give in terminal, will be stored as an array in args.

⇒ Package
→ Folder in which java file lies. like this file should only be accessed by this particular package.

⇒ System is a class that contains "out", a printstream, adds functionality to another output stream, which in-turn has "println".

Taking inputs:

⇒ System class has out while Scanner class has in for input.

```
import java.util.Scanner;
```

```
Scanner input = new Scanner(System.in);
```

Constructor initializing "input"

```
public class Hello{
    public static void main(String[] args){
        System.out.println("Hello, world!");
        Scanner input = new Scanner(System.~~out~~in);
        input.nextInt();
    }
}
```

Primitives (Data types) :- Any data type that can't be further broken to other data types.

integer → int
decimal → float (98.67f)
character → char
double → double (large decimals) (default)
large int → long (1234567891011121314L)
boolean → true or false

```
public class Input{
    public static void main(String[] args){
        System.out.println("Enter student data");
        System.out.println("Enter name: ");
        Scanner name = new Scanner(System.in);
        name.nextLine();
        System.out.println("Enter roll: ");
        Scanner roll = new Scanner(System.in);
        roll.nextInt();
    }
}
```

```
int a = 10;
```

Primitive data type — identifier — literal

Q) Sum of 2 numbers?

A)
```
public class sum{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        int num1 = input.nextInt();
        int num2 = input.nextInt();
        int sum = num1 + num2;
        System.out.println("Sum = "+sum);
    }
}
```

Type Conversion & casting:-

If one type of data is assigned to another type of data variable, then type conversion occurs, provided.

1) 2 types should be compatable (int & float)
2) destination type should be larger than given. (int to float)

```
int num = (int)(67.67f) // casting
```

Automatic type promotion :-

→ When in performing calculation, if range is exceeded, Java automatically promotes to better datatype that has more space

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = (a*b)/c;
```

byte → int

String → ASCII Unicode

↓ is automatically promoted to integer

rules :-

① all short/byte are to integer, float depending on operation.