How Java Code executes -

Java file (Human readable)	Compiler (Entire file) →	.class file (byte code)	interpreter (Line by line) →	Machine code (0 & 1)

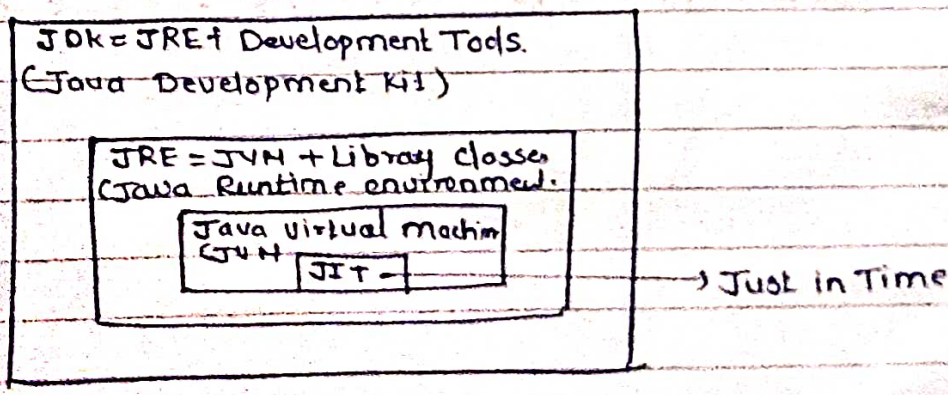
this is the source code.

- this code will not directly run on a system
- we need JVM to run this.
- Reason why Java is platform independent.

More about Platform Independence -

- IL is means that byte code can run on all operating systems
- We need to Convert source code to machine code so Computer can understand.
- Compiler helps in doing this by turning it into executable code.
- this executable code is a set of instructions for the Computer.
- After Compiling c/c++ code we get .exe file Whis is platform dependent
- In Java, we get the bytecode JVM Converts this to machine code.
- Java is platform-Independent but JVM is platform dependent. (bytecode).

JDK vs JRE vs JVM vs JIT



JDK:

- Provides environment to develop and run the Java program.
- It provides package that includes:

1. development tools - to provide an environment to develop your program.
2. JRE - to execute your program.
3. a compiler - javac
4. archiver - Jar.
5. docs generator - javadoc
6. interpreter / loader.

JRE:-

- It is an installation package that provides environment to only run the program.
- It consists of,

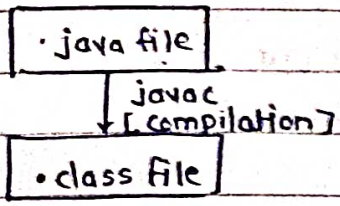
1. Development technologies.
2. User Interface toolkits.
3. Integration libraries.
4. Base libraries.
5. JVM

- After we get the .class file, the next things happen at runtime:

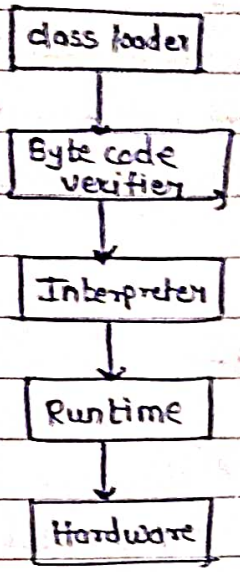
1. class loader loads all classes needed to execute the program.
2. JVM sends code to Byte code verifier to check the format of code

JVM:-

Compile time



Runtime.



(How JVM works) class Loader.

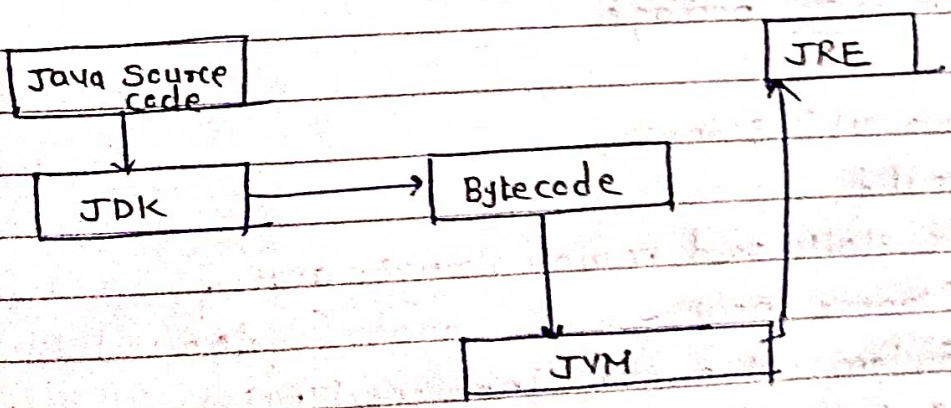
- Loading
  - Reads class file and generate binary data
  - an object of this class is creat in heap.
- Linking
  - JVM verifies the class file
  - allocate memory for all class variables & default values.
  - Replace symbolic references from the type with direct references.
- Initialization
  - all static variables are assign with their values defined in the code & static block.

JVM Execution

- Interpreter.
- Line by line execution
- when one method is called many time it will interpret again and again.

- JIT
- those methods that are repeated JIT provides direct machine code so re-interpretation is not required
- makes execution faster
- Grabage collector.

JVM Contains the static and Heap memory allocations.



First Program:-

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

↳ What is Package ?
→ Package is a Folder

Q System.out.println("Hello World") what it is ?
→ System is class in lang package.
out is a object of PrintStream class.
println() is a method of PrintStream class.
It is used to print in monitor.

Inputs to Java

↳ using Scanner class.
↳ which is in 'util' package.

```
import java.util.Scanner;
class Input{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.println(input.nextInt());
        System.out.println(input.next()); // Take only first word.

        System.out.println(input.nextLine()); // Take whole line of string.
    }
}
```

Data Types:-

↳ Primitive Data Types is type that we can't break the datatype
↳ Have the specific memory. (fixed memory).

```
class Primitives {
    public static void main(String[] args) {
        int milno = 64;
        char letter = 'r';
        float marks = 98.67f;
        double largeDecimalNumbers = 4567654.4567;
        long largeInteger = 9456783456787654 3L;
        boolean check = true;
    }
}
```

Comments In Java:-

1] Single Line Comment → // Hello.
2] multi Line Comment → /*
Hello
*/

Literal & Identifier:-

```
int a = 10;
    ↓
 identifier.
```

* Sum of two Numbers

ans = a + b
a = 10
b = 20
ans = 30

Type Conversion and Casting :-

```
class TypeCasting {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        float num = input.nextFloat();
        System.out.println(num);
    }
}
```

o/p
67
67.0

- Two types should be compatible
- Automatic convert integer to float.
- left hand side is greater than right.

Convert Integer to float:-

using Casting: Compressing bigger Number into the smaller.

```
int num = (int)(67.56);
System.out.println(num);        // 67.
```

automatic type promotion in expressions.

```
int a = 257;
byte b = (byte)(a);    // 257 % 256 = 1
System.out.println(b);    // 1
```

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = (a * b) / c;    (2000)/100
System.out.println(d);    // 20.
```

```
Int number = 'a';
System.out.println(number);  // 97.
```

Rules for Type Promotion:

- All the byte, short and char promote to Integer.
- Any one of the operand is Long whole is Promote to Long.
  Same for float and double.
  ex. 3 * 5.6 → float.

```
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = 0.1234;
double result = (f * b) + (i / c) - (d - s);
System.out.println((f * b) + " " + (i / c) + " " + (d - s));
System.out.println(result);
```

If-Condition (statement)

```
if (condition) {
    body
}
```

Note: Condition is true then goes into the if statement body.

While-loop

```
int count = 1;
while (count != 5) {
    SOP(count);
    count++;
}
```

o/p => 1
2
3
4

for loop

```
for (count = 1; count != 5; count++) {
        SOP(count);
}
```

O/P =>
1
2
3
4

Q: Temperature Conversion program.

```
import java.util.Scanner;
class Temperature {
    public static void main (String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Please enter temp in C: ");
        float tempC = in.sc.nextFloat();

        float tempF = (tempC * 9/5) + 32;
        System.out.println(tempF);
}  }
```