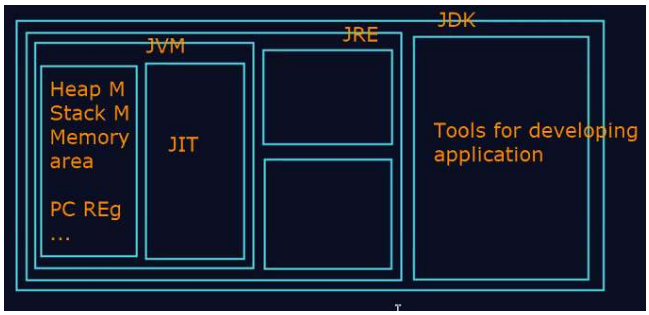
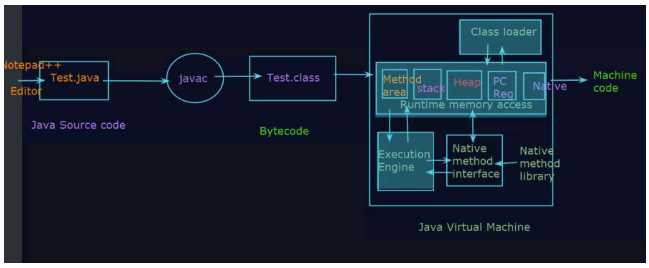
Day_2_OOPJ_Sanket_Shalukar

Tuesday, August 26, 2025 10:13 AM

Topics :

- Java Tokens : Keywords, Identifiers, literals, operators
- Declaring variables and methods
- Data type compatibility
- Programs





2. **Runtime Data Access (Memory)**

- These memory regions are managed by JVM at runtime.

Method Area:

- Stores the class level data
- Share among all threads

Heap:

- Stores all objects and their instance variables
- Share among all threads

Java Stack:

- Stores frames for each method call (local variables, intermediate values)
- Each thread has its own stack

PC Register:

- Native Methods Stack: -Stores the current instructions address being executed for each thread.

Native Methods Stack:

- Support native (nan-java) method execution written C/C++

3. **Execution Engine :**

- Just-in-time (JIT) Compiler
- Converts bytecode into native machine code at runtime
- Improves performance by caching compiled code

Interpreter

- Read and execute bytecode line by line
- Slower compared to compiled code

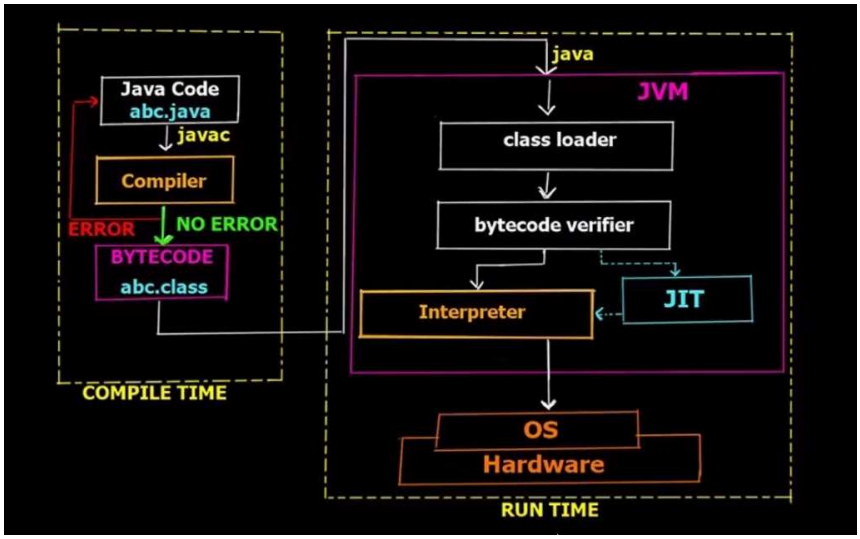
Garbage Collector (GC)

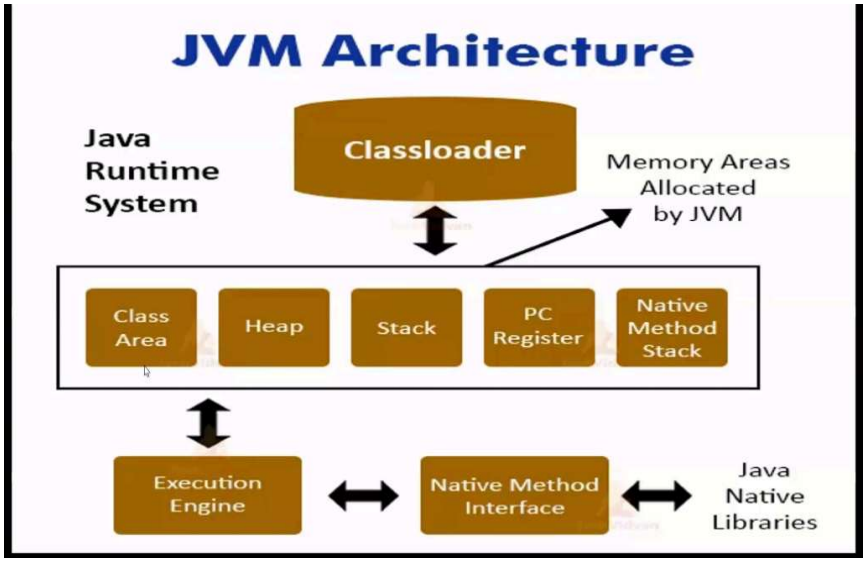
- Automatically reclaims memory used by unreachable objects
- Runs in the background to free Heap memory space.

4. **JVM Workflow :**

1. Source code (.java) -> compiled by javac -> bytecode (.class)
2. Class loader loads the .class file into JVM
3. Bytecode verifies for security
4. Loaded clases stored in Method area; objects store in Heap.

5. Execution Engine excutes instructions using Interpreter or JIT
6. Garbage Collector manages unsused objects.



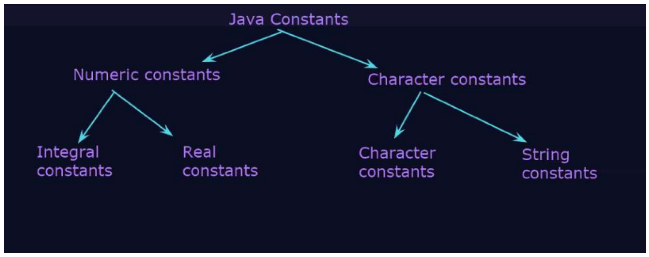


4. **JVM Thread**

- Main Thread: Execute the main () method
- User defined threads (Thread (class) or Runnable (interface))
- public static void main (String args []) {}
- Reference Hadler Thread
- Finalizer Thread
- Signal Dispatcher Thread
- Compiler thread
- Garbage Collector

Literals : Java Constants

1. Numeric Constans
2. Character constants



Floating Point Literals:

- Default: double
- float = suffix f/F

- double = suffix d/D (optional)

Ex:

float f = 123.456f;

double d = 123456.789;

double d = 123456.789D;

float d = 123456.789;

Boolean Literals:

- **true or false**

Ex:

boolean b = 0; //Error:

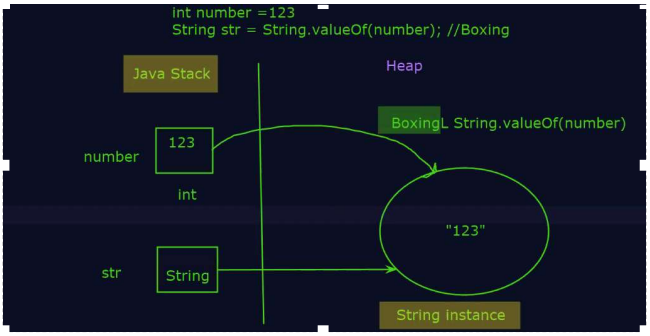
```
class PrimitiveData{


        public static void main(String args[]){

            byte b =127;
            short s = 345;
            int  i =66666;

            System.out.printf("Byte = "+b);
            System.out.printf("Short = "+s);

    }


}
```

```
    public static void main(String args[]){

        byte b =127;
        short s = 345;
        int  i =66666;
        long l =99999999999L;
        float f = 45.005f;
        double d = 345.456789;
        boolean b1 = false;
        char ch = 'A';

        System.out.println("Byte = "+b);
        System.out.println("Short = "+s);
        System.out.println("Int= "+i);
        System.out.println("Long = "+l);
        System.out.println("Float = "+f);
        System.out.println("Double = "+d);
        System.out.println("Boolean = "+b1);
        System.out.println("Char = "+ch);
    }
                println
```

```
class ConcatOp{


        public static void main(String args[]){

        int i = 10;
        int j = 20;
        int k = i+j;

        System.out.println(k);
        System.out.println("Sum = " + k);
        System.out.println("Sum of "+i+"and "+j+"is ="+k);

    }
}
```

```
C:\WINDOWS\syster
C:\Test>javac ConcatOp.java

C:\Test>java ConcatOp
30
Sum = 30
Sum of 10and 20is =30

C:\Test>
```

```
class Casting{


        public static void main(String args[]){

        byte b =127;
        int i = b;// Upcasting: byte (1byte) = int (4bytes)
        System.out.println(b);
        System.out.println("Int i ="+i);

        int j = 127;
        byte b1 = (byte)j;// Downcasting: int(4 byte) = byte(1byte)


    }


}
```


int number =123
String str = String.valueOf(number); //Boxing
Java Stack
Heap
BoxingL String.valueOf(number)
number
123
int
"123"
str
String
String instance