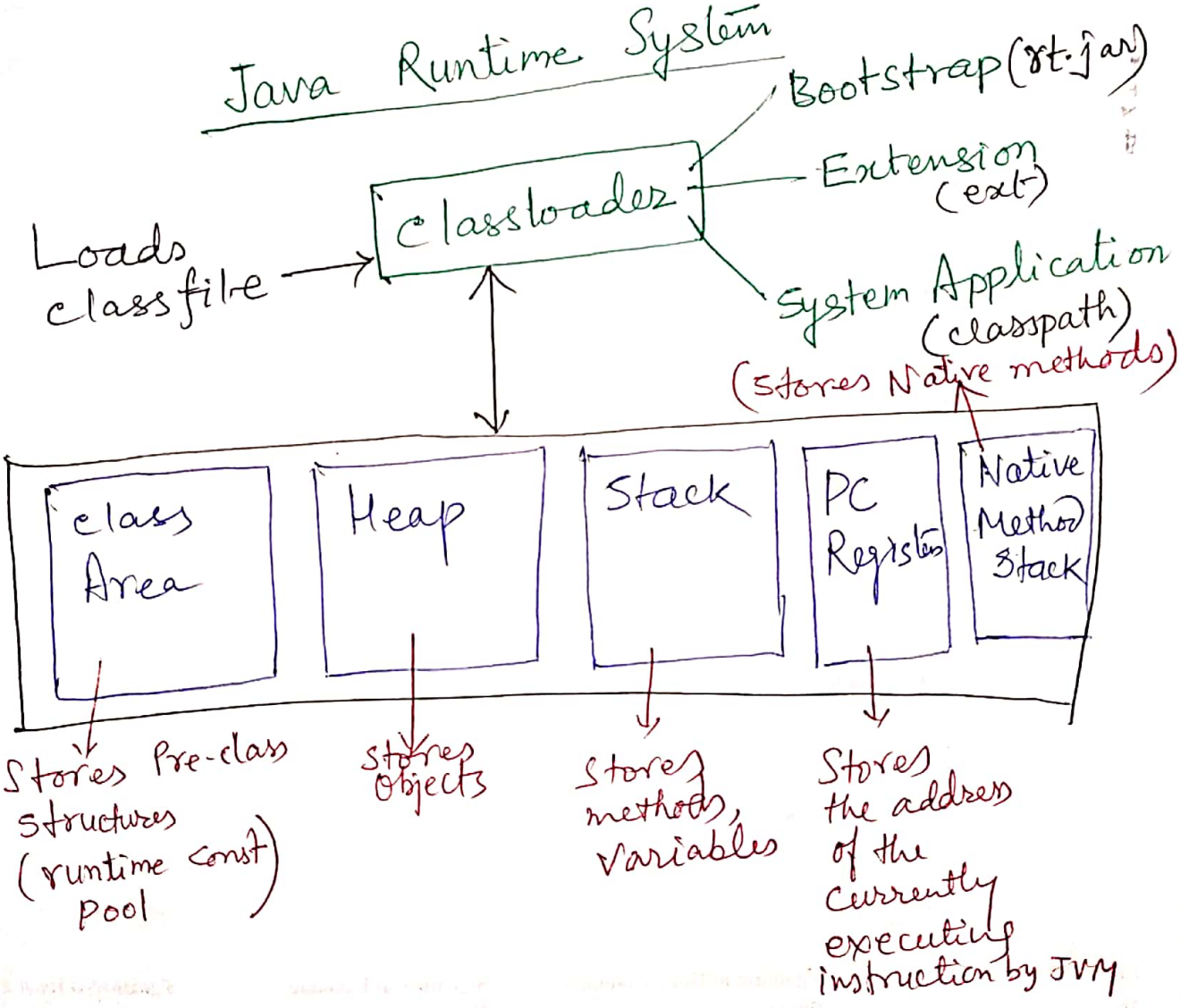
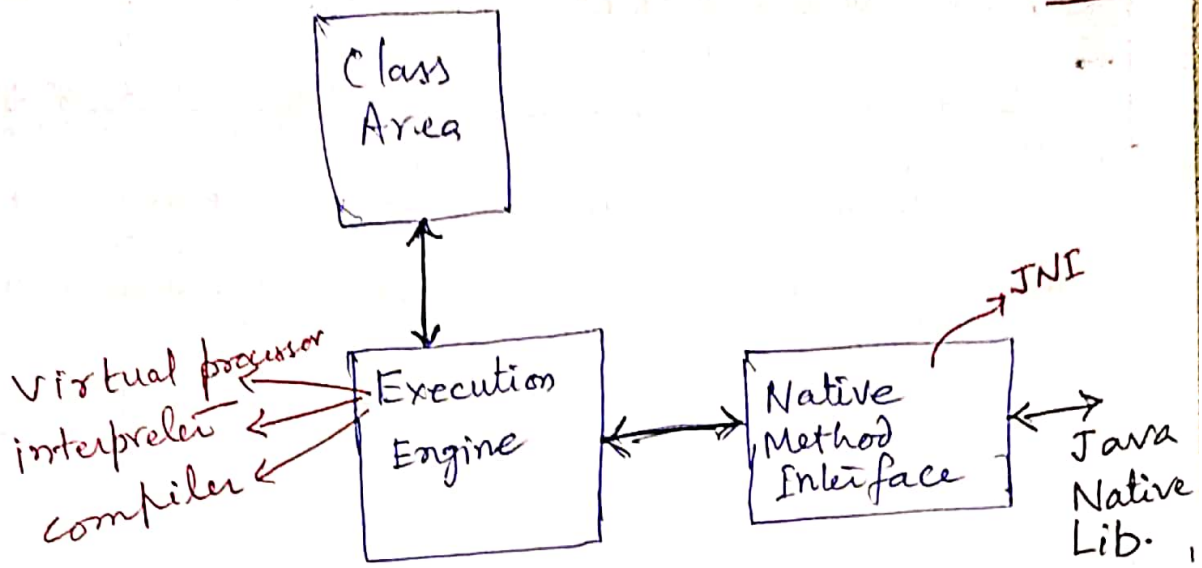
Architecture of Java Virtual Machine Architecture:

Java Virtual Machine — It is an abstract machine. It provides runtime environment to execute the bytecode.

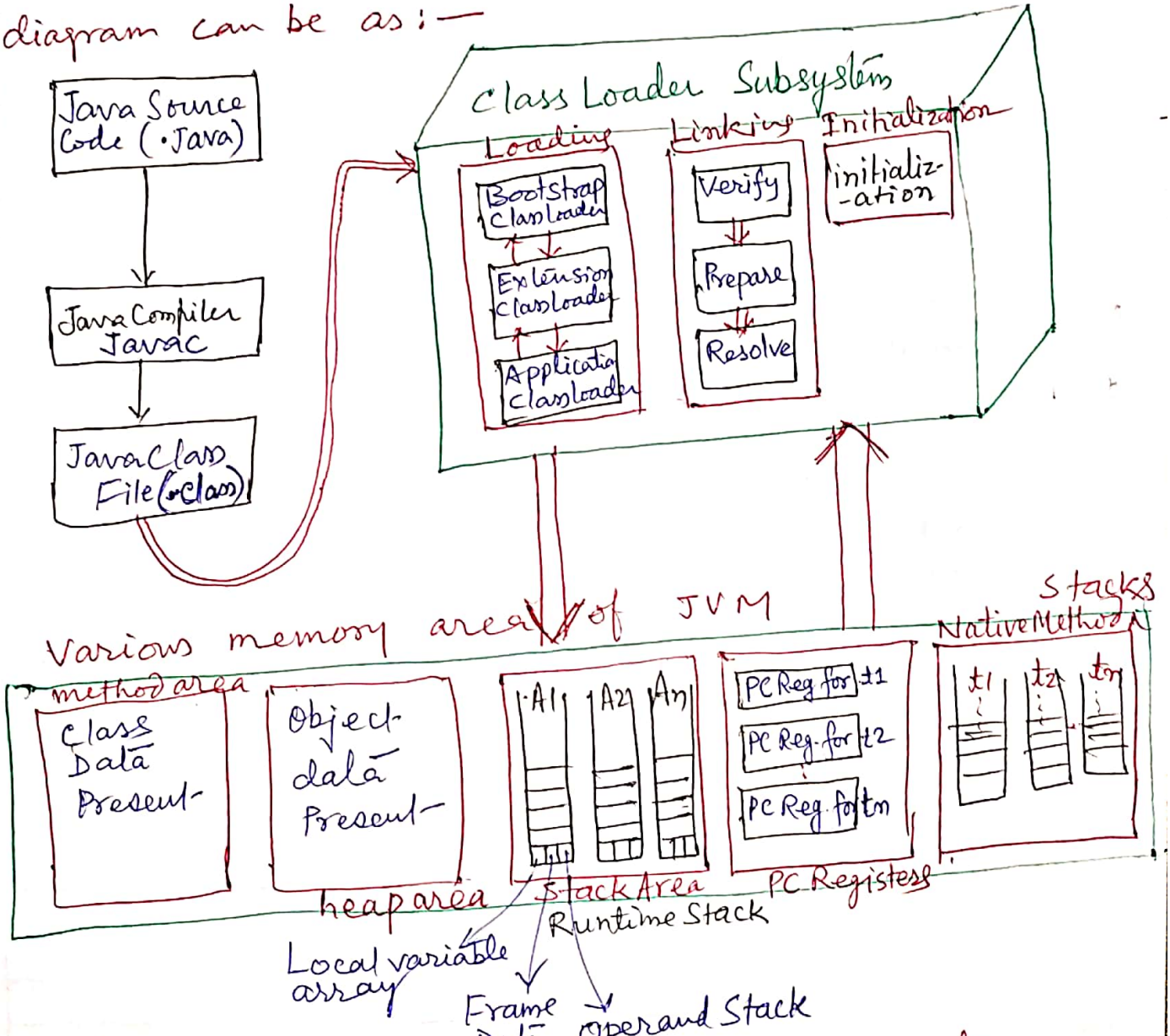
operations by JVM:—

→ Loads the code
→ Verifies the code
→ Executes the code
→ Provides Runtime Environment.



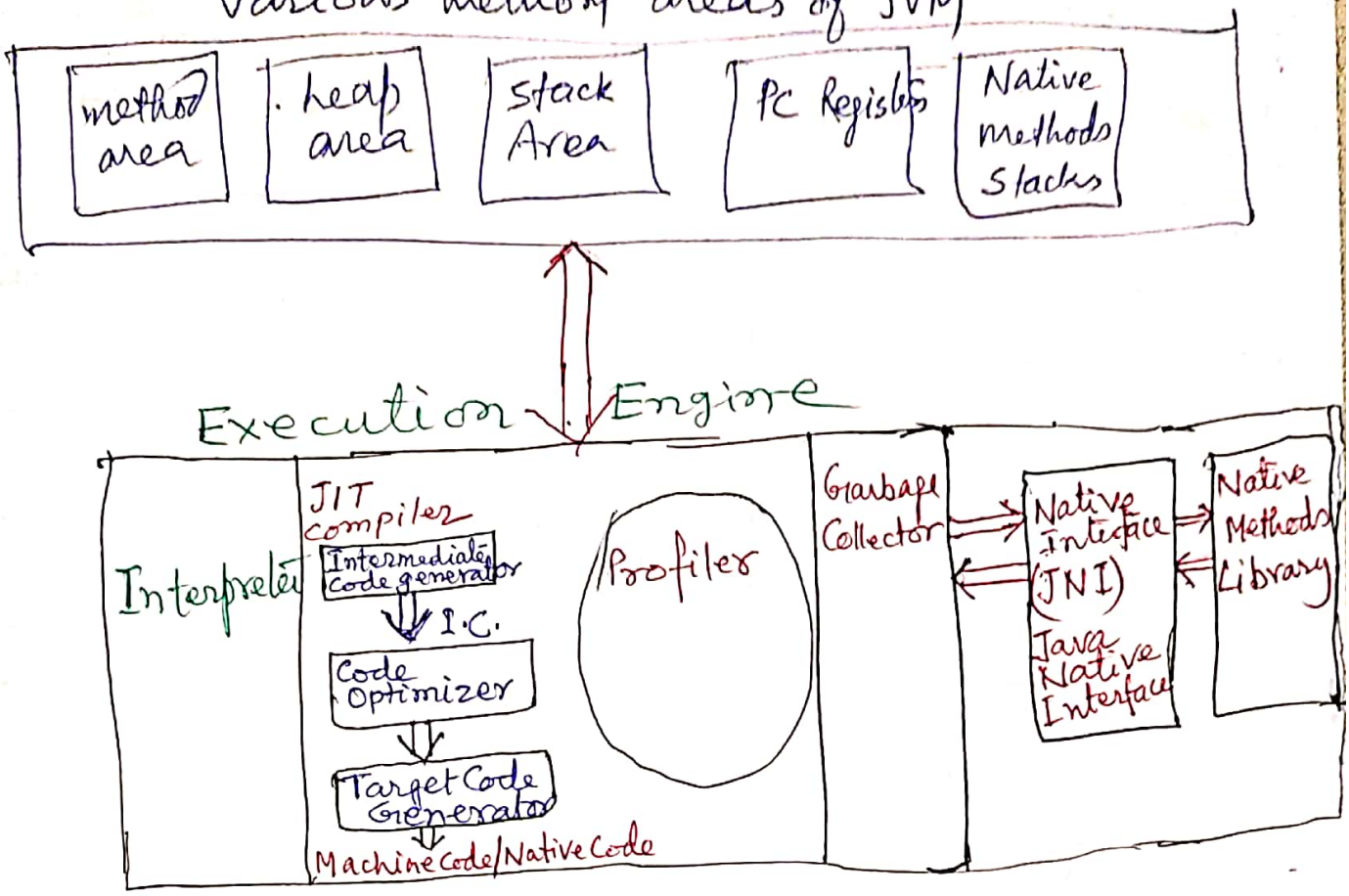


Hence complete working of Java compile & Run process diagram can be as:—



Execution Engine is responsible to execute our Program which is communicating with various Memory area of JVM.

Various memory areas of JVM



→ Heap area and Method area are the shared memory for multiple threads running.

→ But for every thread has its separate run-time stack which is in the Stack area.

→ every data in stack area is thread-safe.

→ for every thread, 1 PC Register will be created to hold address of next executing instruction.

→ for every native method invokation, JNI is responsible.

⇒ Heap area is very important for programmers.

⇒ Execution Engine is responsible for executing of our program — Read, Interpret and Execute java program line by line. If any method is repeatedly required then instead of interpreting every time just compile once, JIT compiler

compiles at least once and executes many times. The profiler decides which method is called many times.

If at run-time any native method internally required to call — JNI is responsible for that type of native method calling. That's how java program compiles and executes.

Let us summarize the components of JVM architecture :—

The Virtual Machine is of types —

1. Hardware Based Virtual Machine
2. Application Based Virtual Machine

The Basic Architecture of JVM is — divided in three sub-systems :—

I. Classloader sub-system —

① Loading
② Linking
③ Initialization

Types of class Loaders —

1. Bootstrap class Loader
2. Extension class Loader
3. Application class Loader

How ClassLoader works —

If we are not satisfied with the default class Loader mechanism, then we can also define new class loader mechanism — Customized class Loader.

We can define Pseudo Code for Customized class Loader.

II. Various Memory areas of JVM are —

1. Method area
2. Heap area
3. Stack area
4. PC Registers
5. Native Method Stacks

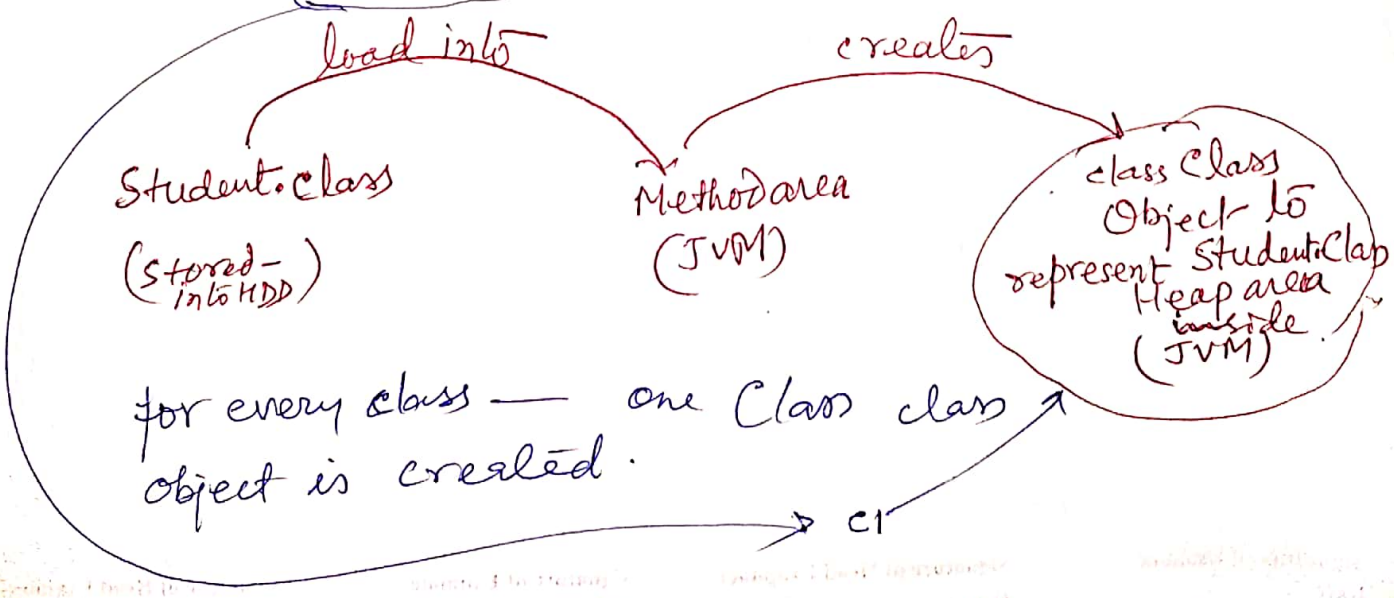
Program to display heap memory statistics.
How to set maximum & minimum heap size?

III. Execution Engine — responsible for line by line execution.
   1. Interpreter
   2. JIT Compilers

Java Native Interface (JNI) — responsible for any native method called by the execution engine.

Being a programmer's point of view, we should know the actual happening when we create a object of a class as:—

```
Student a1 = new Student();
Class c1 = a1.getClass();
```



Suppose we have made another object a2:-

```
Student a2 = new Student();
```

Then there is no creation of Class class object by the JVM. Since it happens only first time.

Now, c2 is also pointing to the same Class class object.

```
Class c2 = a2.getClass();
```

We can see this by using following code:-

```
System.out.println(c1.hashCode());
System.out.println(c2.hashCode());
System.out.println(c1==c2);
```

Note:- For every loaded type only one class object will be created even though we are using the class multiple times in our program.

Let us do the complete program:- Test.java

```
import java.lang.reflect.*;
class Students
{
    public String getName()
    {
        return null;
    }
    public int getMarks()
    {
        return 10;
    }
}
```

```
class Test
{
    public static void main(String args[])
    {
        Students s1 = new Students();
        Class c1 = s1.getClass();
        Students s2 = new Students();
        Class c2 = s2.getClass();
    System.out.println(c1.hashCode());
    System.out.println(c2.hashCode());
    System.out.println("Both objects are Same or Not:"
                        +(c1==c2));

    } //close of main function.
} //close of Test class
```

output shows that both c1 and c2 has same hashCode (Address Numbers). Both objects are also same hence (true) will be printed.

We have seen ~~[illegible]~~ here is that, even though we are creating Students class object two times but only one [Class] class object gets created.

Note — Class is a class made by JVM for every java class object we have created — first time only.