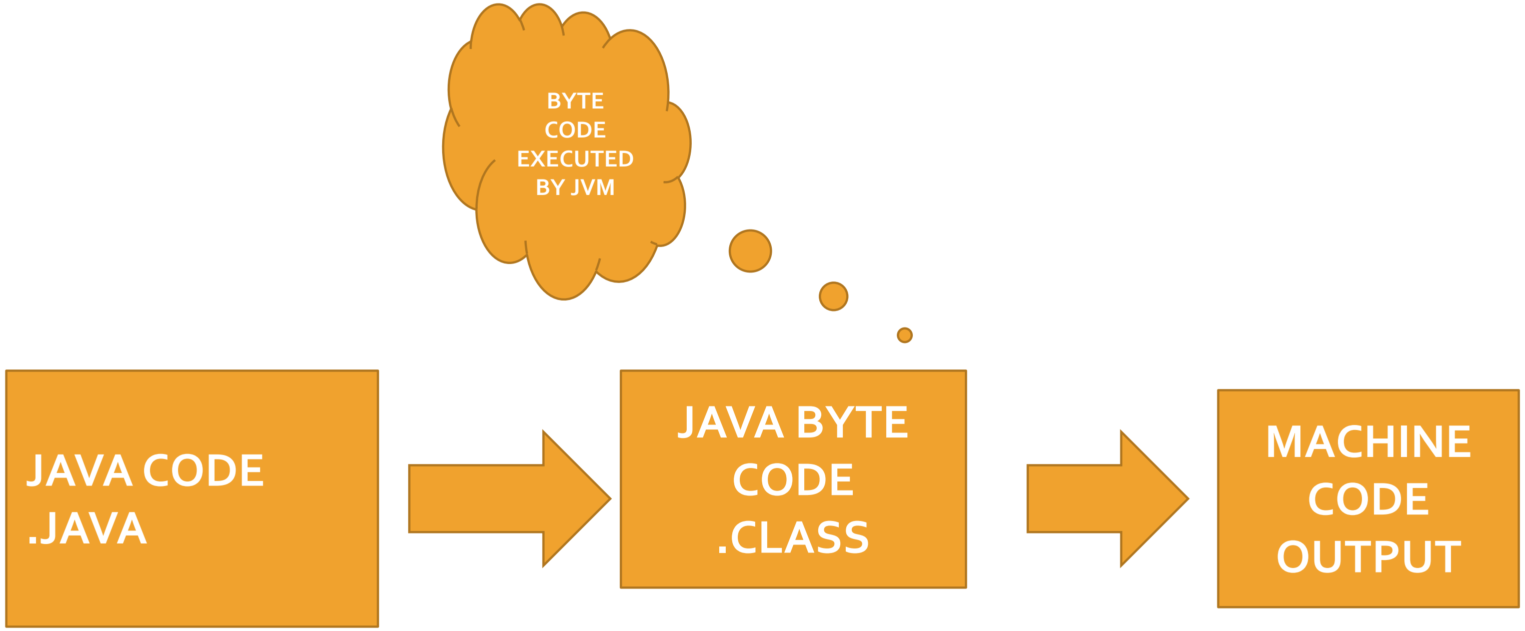
JVM (Java Virtual Machine) Architecture

WELCOME

WHAT IS JVM
Architecture

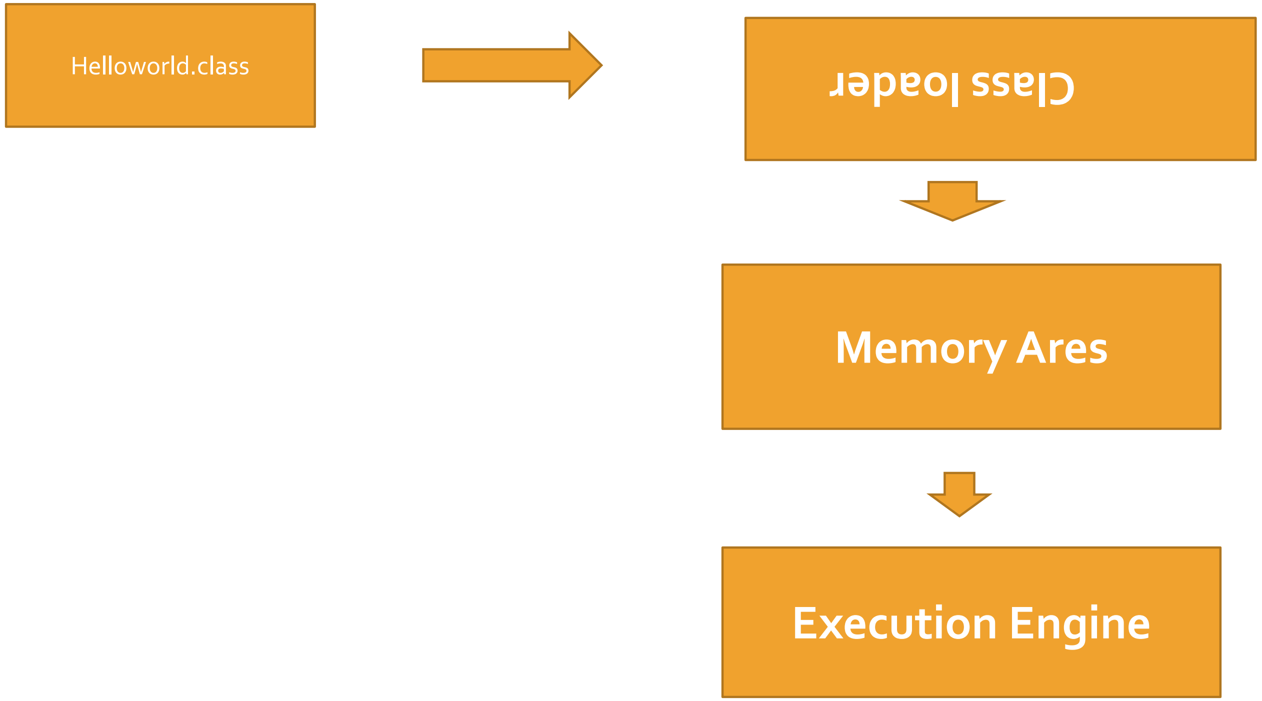
- ***JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.***

STEP-1



The.class file (for example helloworld.class file)goes into the various step when we execute it.

Jvm mainly divided into three parts:



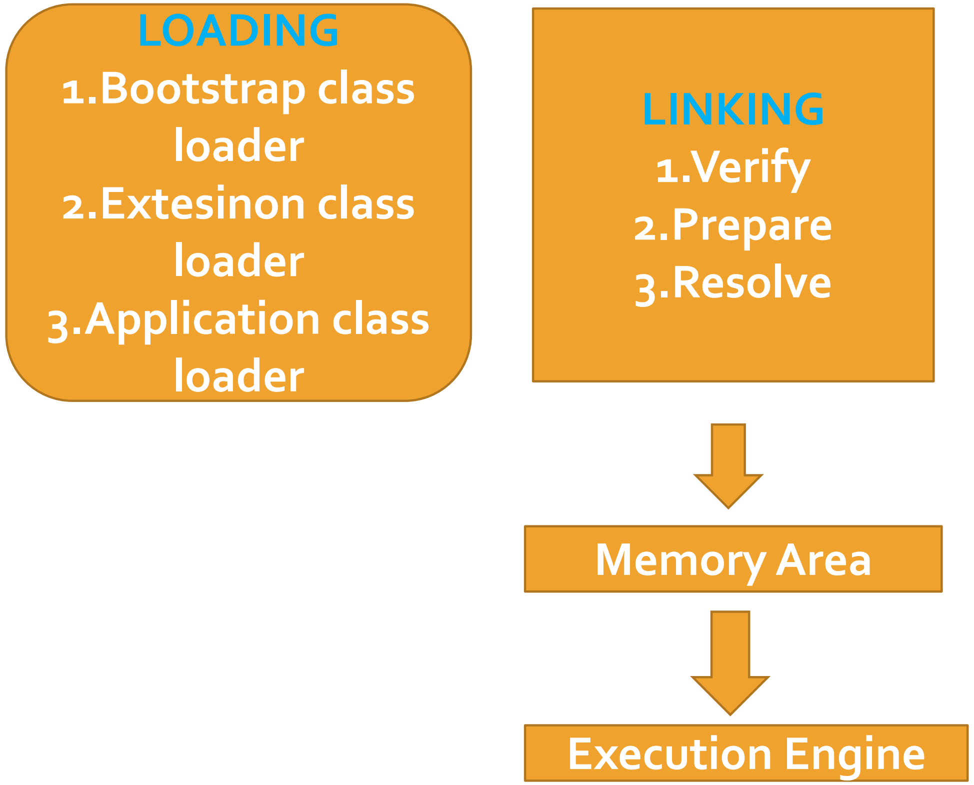
Helloworld.class

Class loader

Memory Ares

Execution Engine

Class loader Subsystem



LOADING

1.Bootstrap class loader

2.Extesinon class loader

3.Application class loader

LINKING

1.Verify

2.Prepare

3.Resolve

INIITIALIZATION

1.Al static variable are assign with values

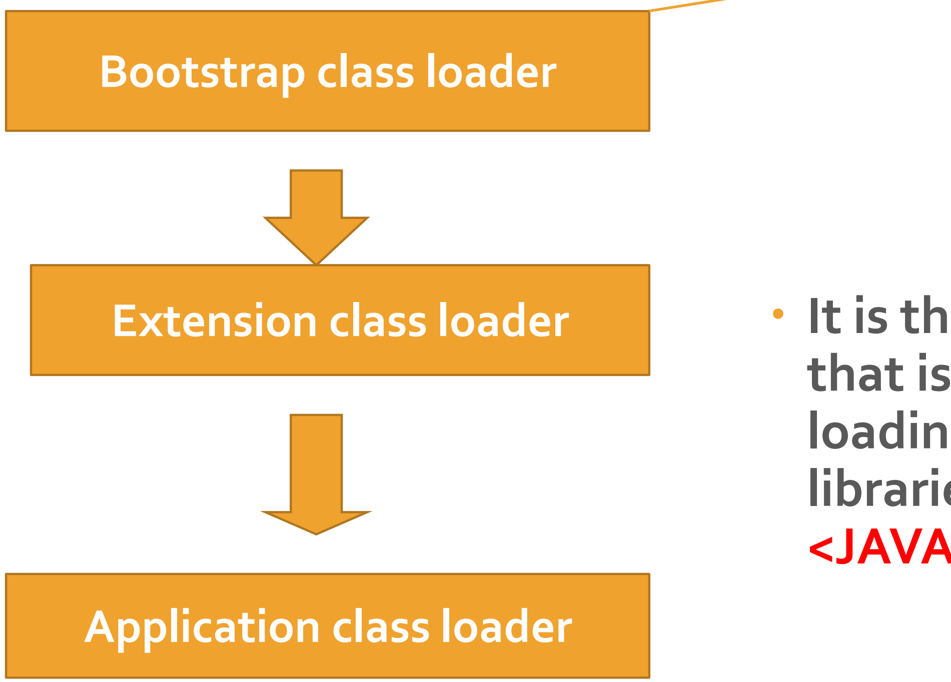
2.Static block will be executed from top to bottom

Memory Area

Execution Engine

JVM CLASS LOADER

LOADING

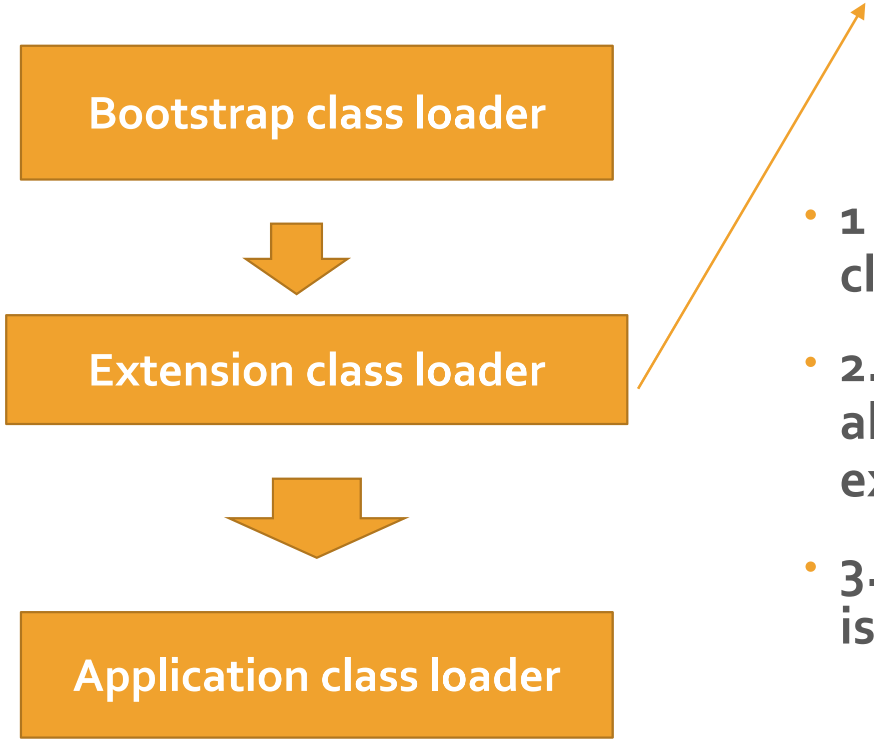


Bootstrap class loader Explanation

- It is the first class loader that is responsible for loading the core java libraries located in the **<JAVA_HOME>/jre/lib/*.jar.**

JVM CLASS LOADER

LOADING

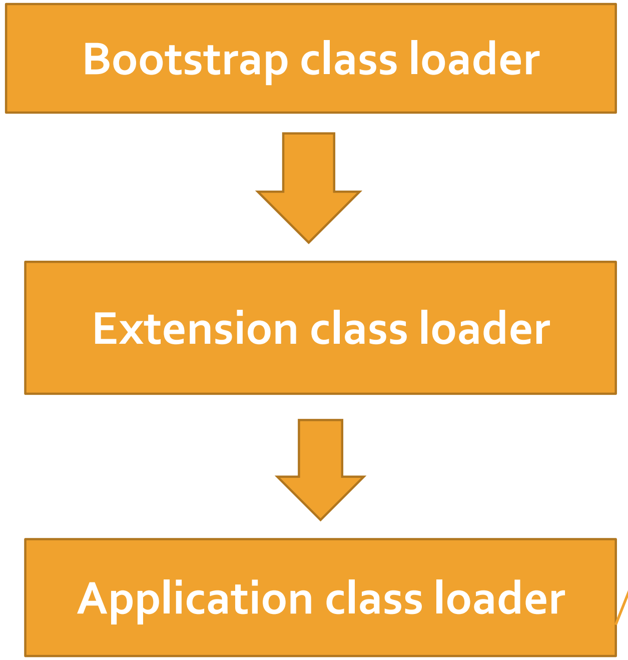


Extension class loader Explanation

- 1 it is child class of Bootsrap class loader.
- 2.it is responsible of loading all classes from the extension class path in java.
- 3.Extension class path is:<JAVA_HOME/jre/lib/exb

JVM CLASS LOADER

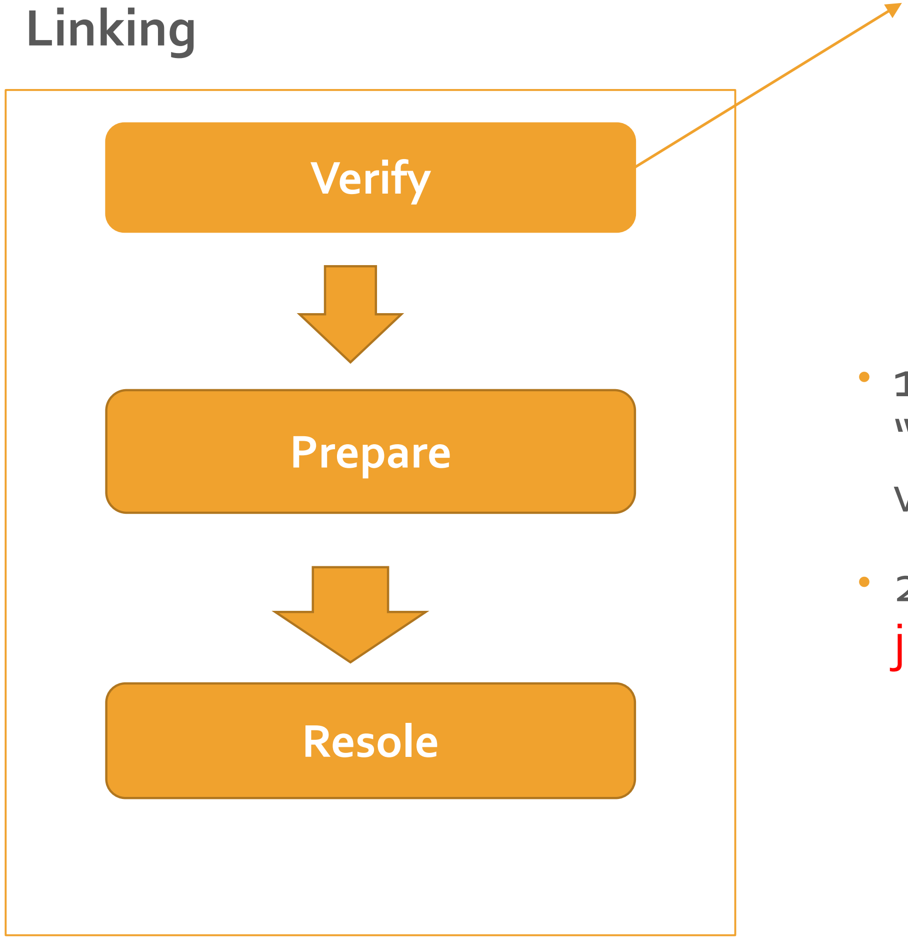
LOADING



Application class loader Explanation

- 1.it is chaild class of Extension class loader.
- 2.it is responsible of loading classes from the application classpath.
- 3.Application class loader deling with application-specific classes.

JVM CLASS LOADER

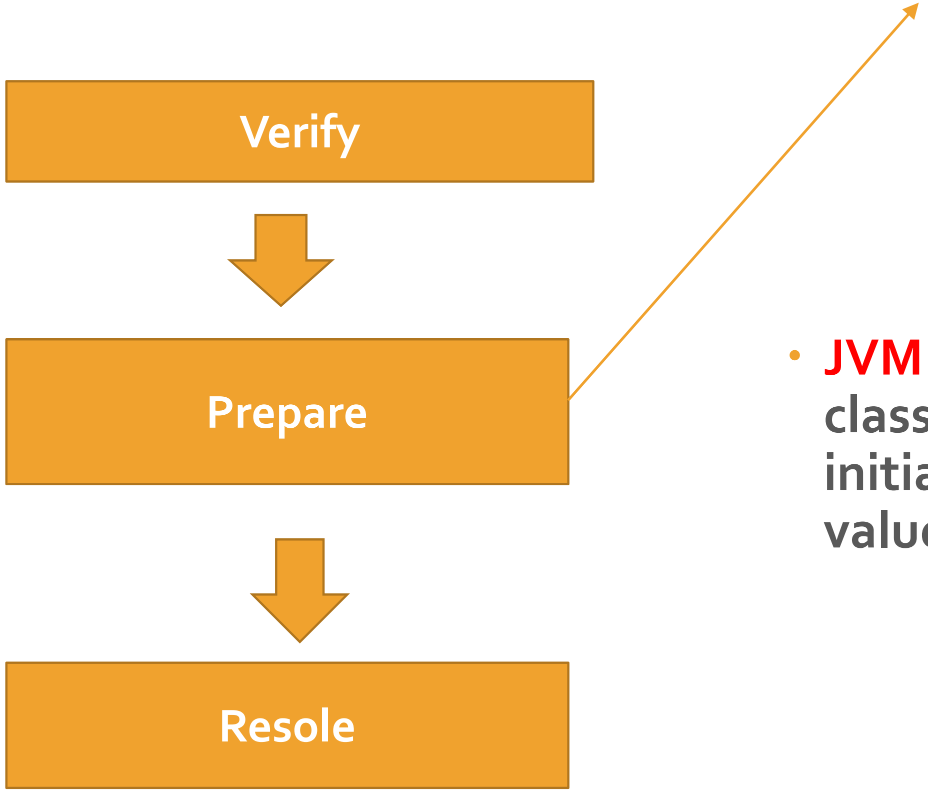


Verify Explanation

- 1.it checks whether the ".class" file is generated by valid compailer or not.
- 2.if verifaction fails,it thrown java.lang.verifyError.

JVM CLASS LOADER

Linking



Verify

Prepare

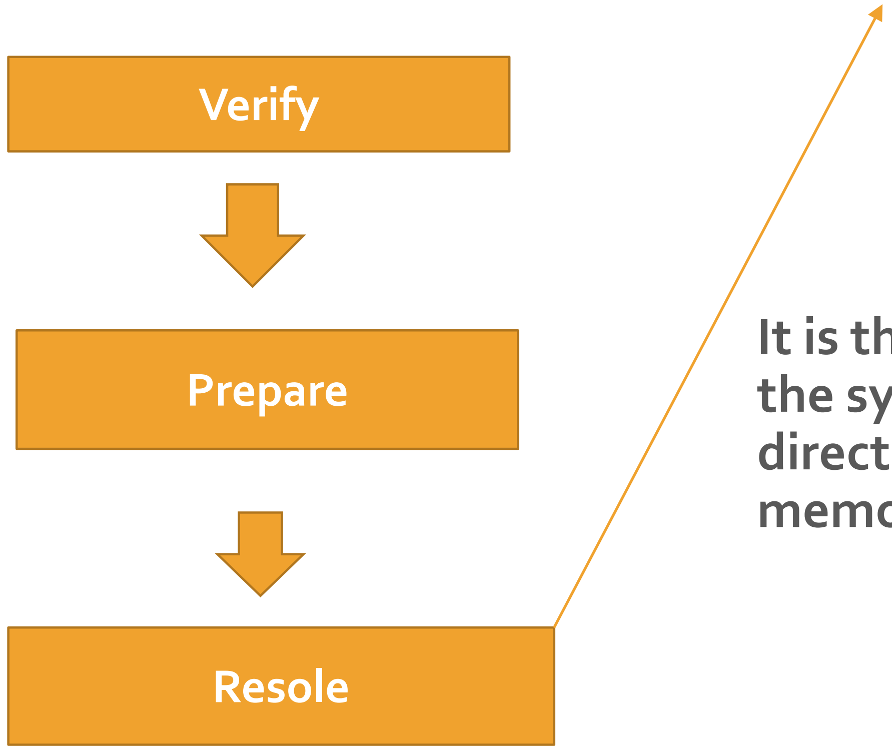
Resole

Prepare Explanation

- JVM allocates memory for class variables and initializes them with default values.

JVM CLASS LOADER

Linking



Verify

Prepare

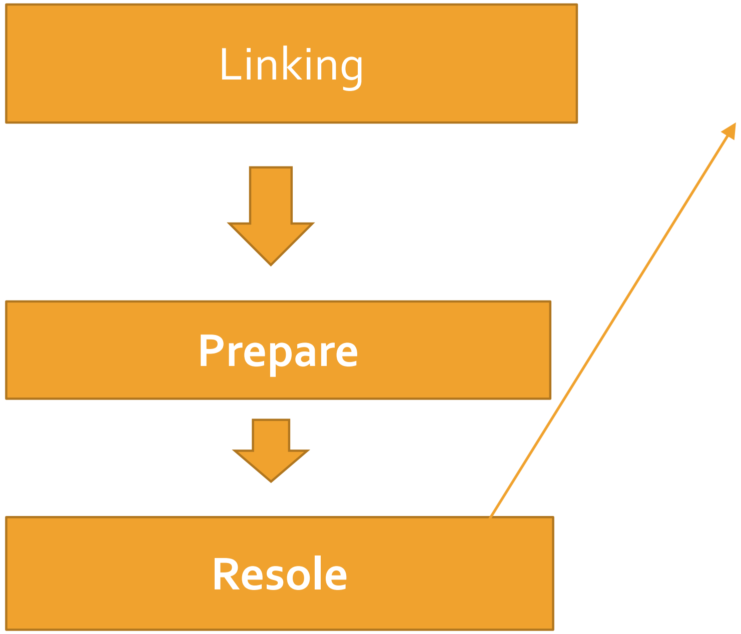
Resole

Resole Explanation

It is the process of replacing the symbolic reference with direct reference to actual memory addresses.

JVM CLASS LOADER

Linking



Linking

Prepare

Resole

Resole Example

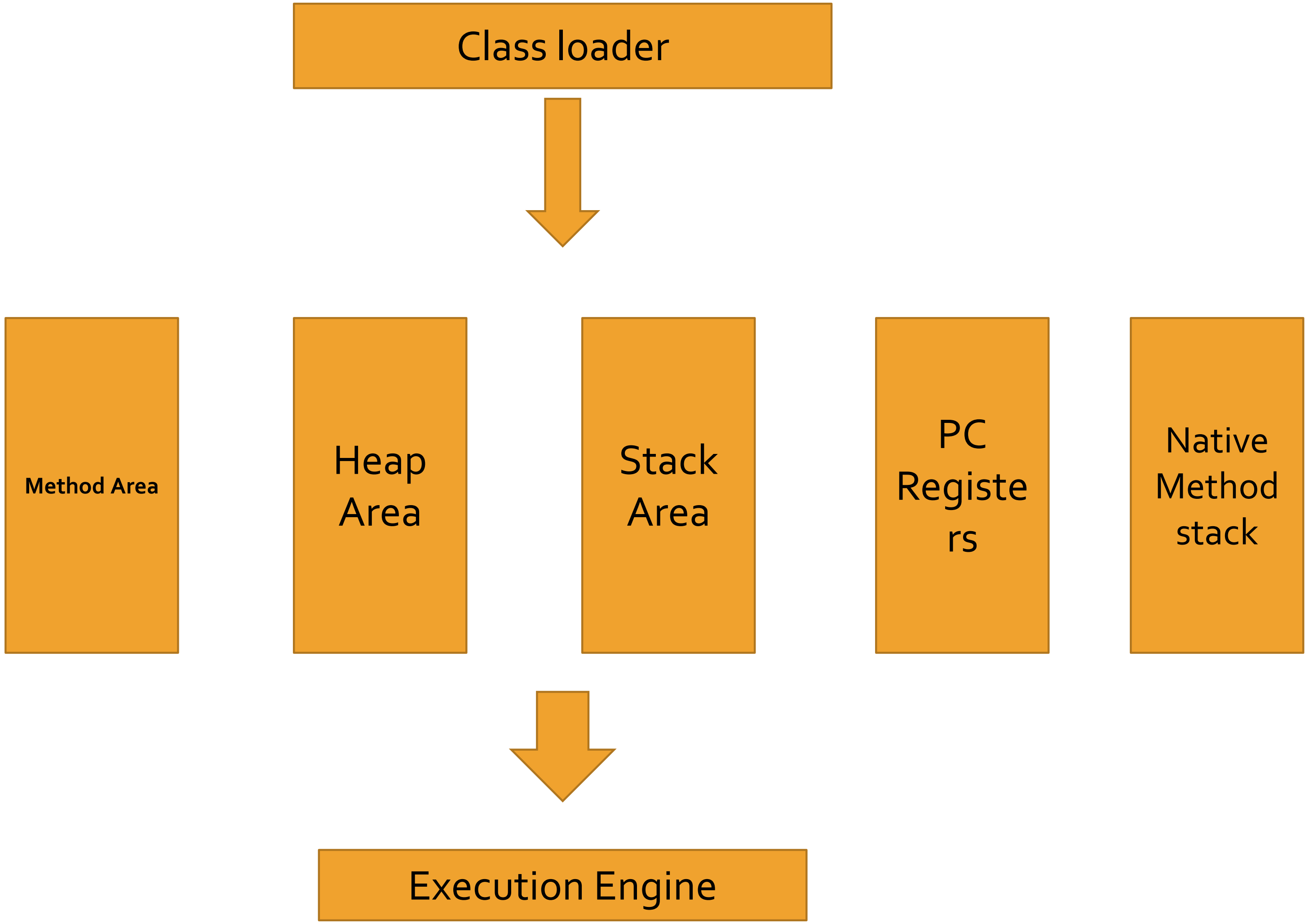
- *Public class Main{*
- *Public static void main(string[] args) {*
- *Helper helper=new helper();*
- ***Helper.performTask();***
- *}*
- *}*
- *The refrence to Helper and its method PerformTask in the Main class is* ***symbolic****.the actual memory location are not knows at compile time.*
- *During the* ***linking*** *phase,the* ***jvm loads tha Main class****. At this point,symbolic references to Helper and its method need to be resolve.*
- *The* ***resolution*** *step finds the actual memory addresses for the Helper class and its PerformTask method.it ensures the the* ***Main*** *class can call the correct method on the correct object.*

Initialization

1.All static variables are assign with values.

2.Staic block will be executed from top to bottom.

JVM-Memory Area



Class loader

Method Area

Heap Area

Stack Area

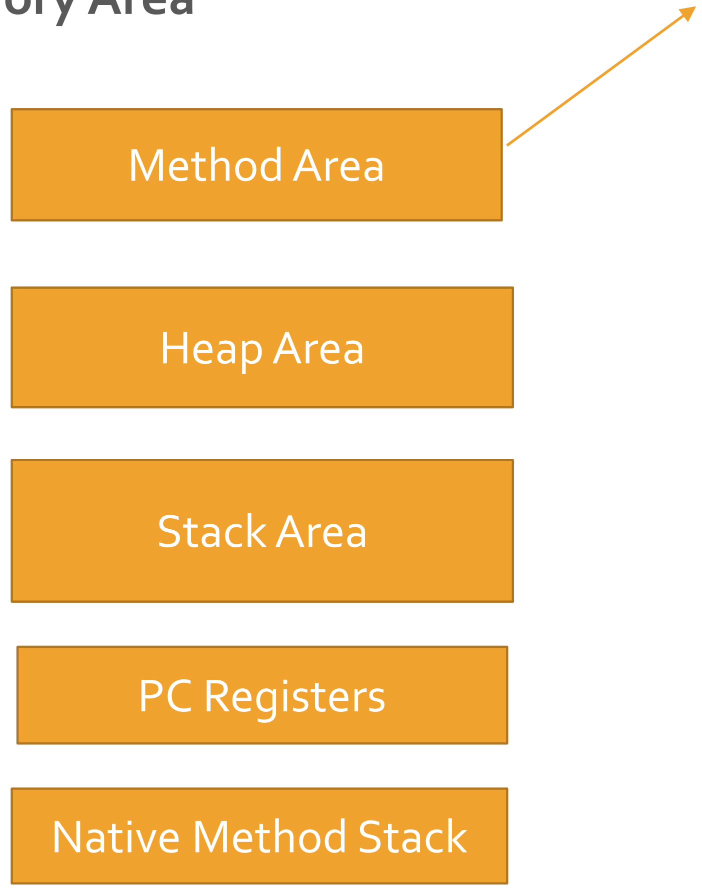
PC Registe rs

Native Method stack

Execution Engine

JVM-Memory Area

Memory Area



Method Area

Heap Area

Stack Area

PC Registers

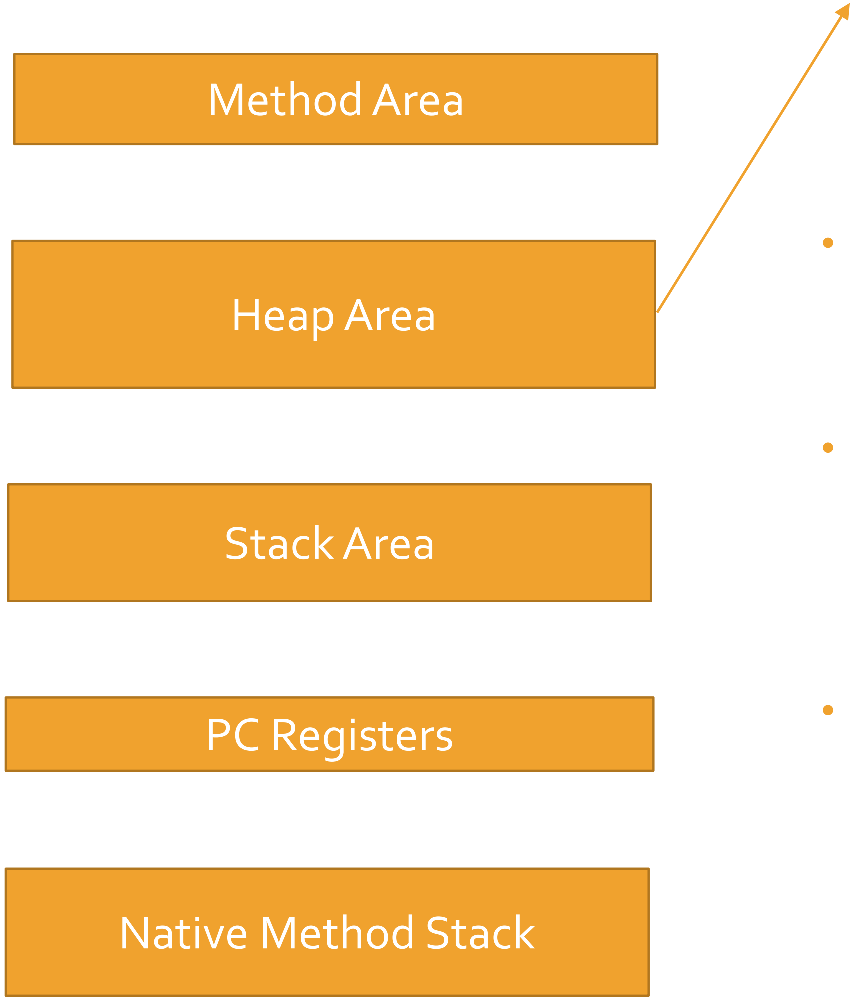
Native Method Stack

Method Area Explanation

- 1.it stores class level information,class name,method and variable information.
- 2.static variables.
- 3.The method area is shared among all the threads running in the jvm.(not thread safe.)
- 4.only one Method area per jvm.

JVM-Memory Area

Memory Area



Method Area

Heap Area

Stack Area

PC Registers

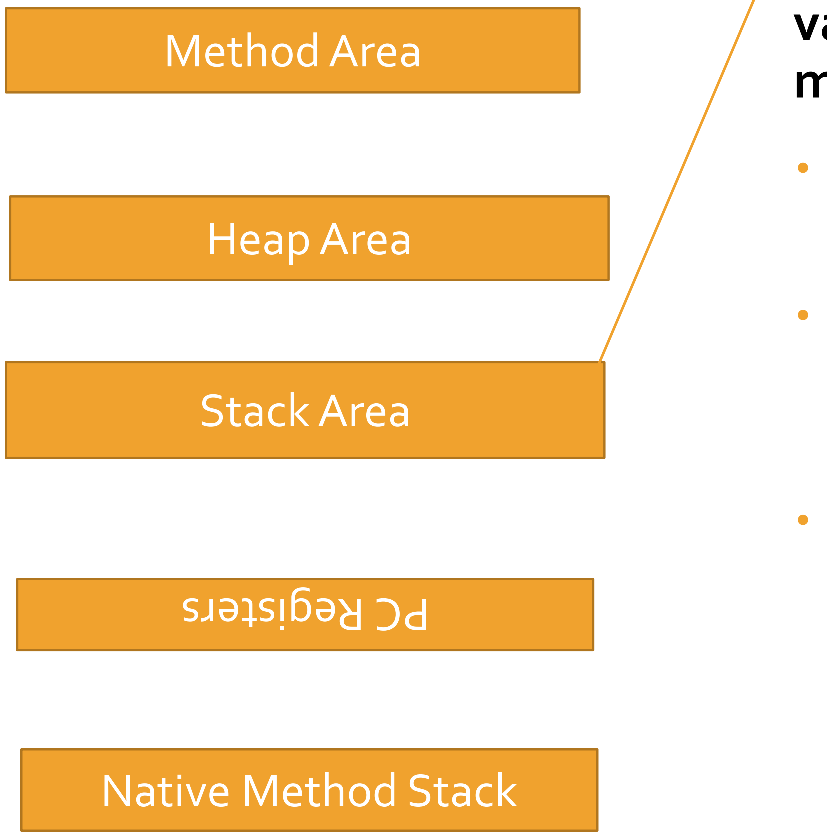
Native Method Stack

Heap Area Explanation

- **1.it stores objects,arrays,instance variable.**
- **2.the heap area is shared among all the threads running in the jvm.(not thread safe)**
- **3.only one Heap area per JVM.**

JVM-Memory Area

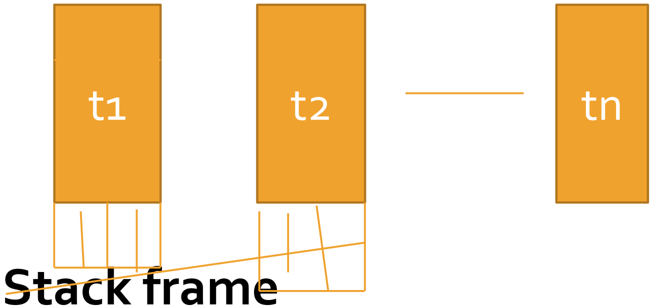
Memory Area



Stack Area Explanation

1.it stores local variables,current running methods.

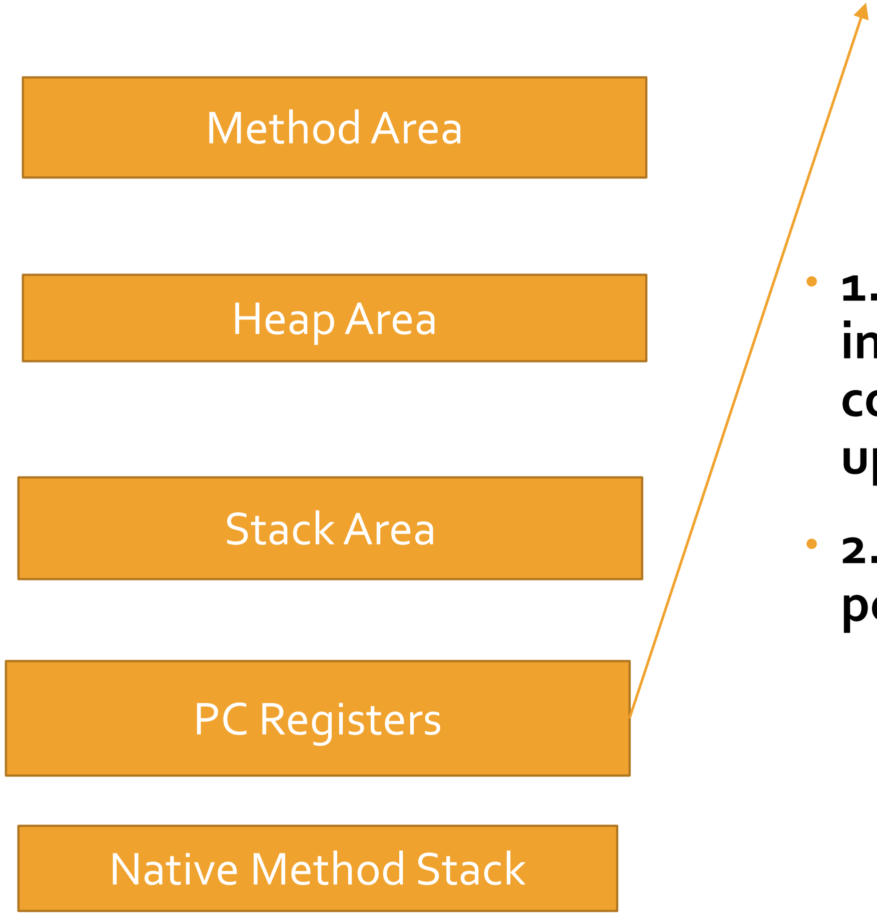
- 2.for every thread,JVM creates one runtime stack.
- 3.Each block of stack is called activation record/stack frame.
- 4.Each frame contains:local variable,frame data and operand stack.



- Stack frame

JVM-Memory Area

Memory Area



Method Area

Heap Area

Stack Area

PC Registers

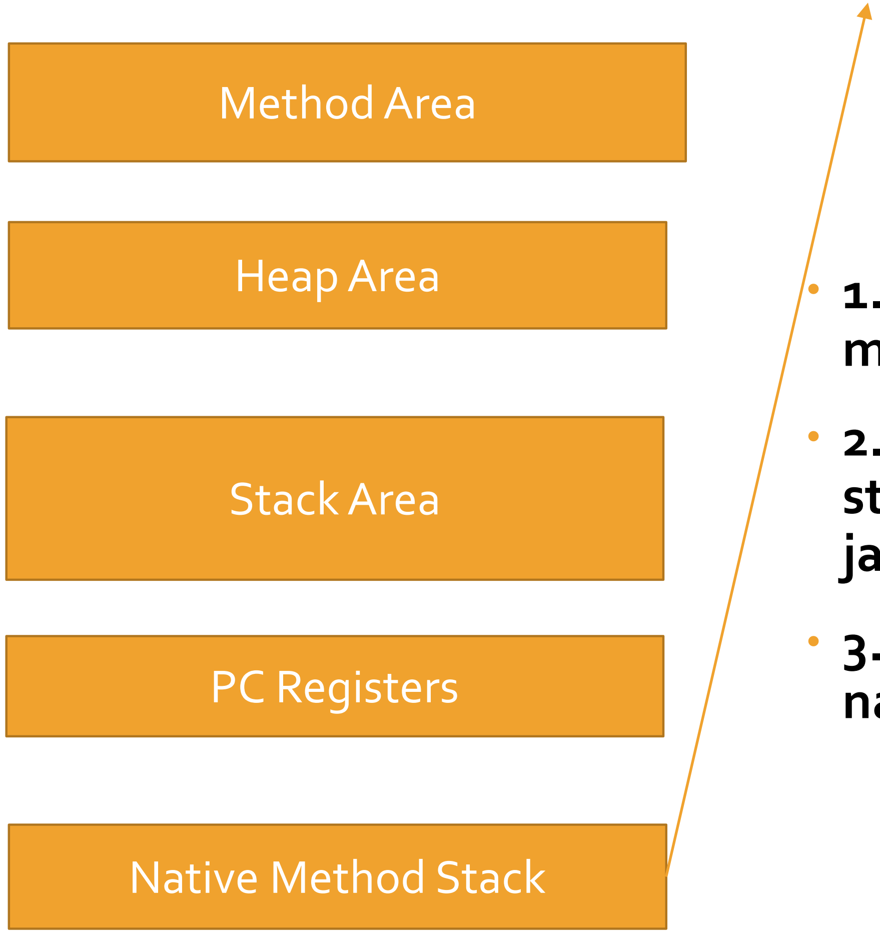
Native Method Stack

PC Registers Explanation

- 1.it stores current exeution instruction,once it completes,Automatically update the next pc Register.
- 2.Each thread has seprate pc Registers.

JVM-Memory Area

Memory Area



Method Area

Heap Area

Stack Area

PC Registers

Native Method Stack

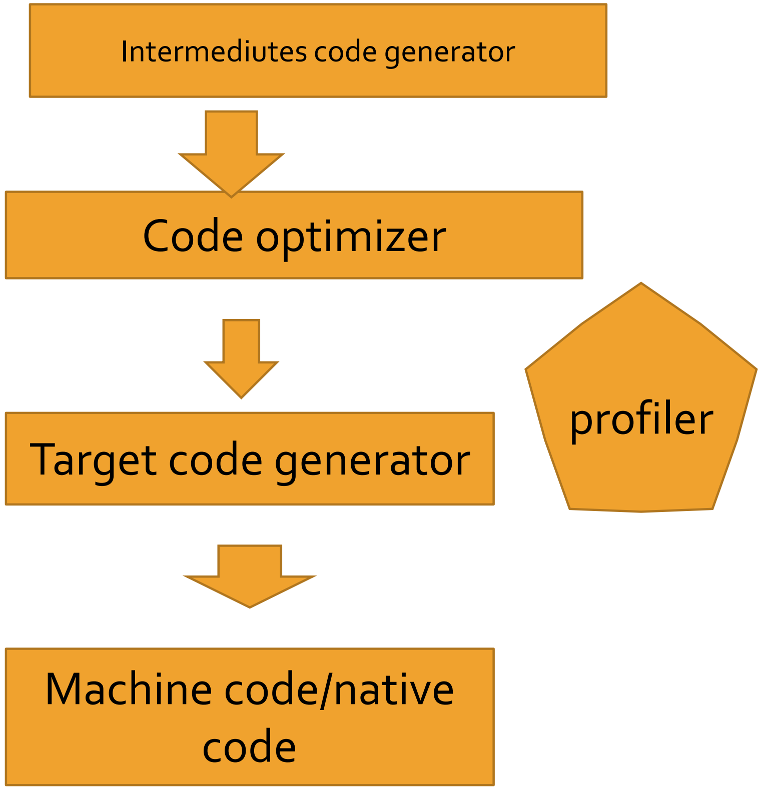
Native Method Stack Explanation

- **1.Memory used for native method execution.**
- **2.it is separate from the java stack to handle native(non-java)like c and c++ code.**
- **3.for every thread separate native stack is created**.

Execution Engine

Interpreter

JIT compiler



Other component like:

- **Garbage collector**
- **Security** Manager**.**

Interpreter Explanation

- **It is responsible to read byte code and interprets into machine code line by line.**
- **The problem with interpreter,is it interprets every time even if repeated method called.Which effects the performance.**
- **To overcome this problem JIT compiler introduced in 1.1 version**.

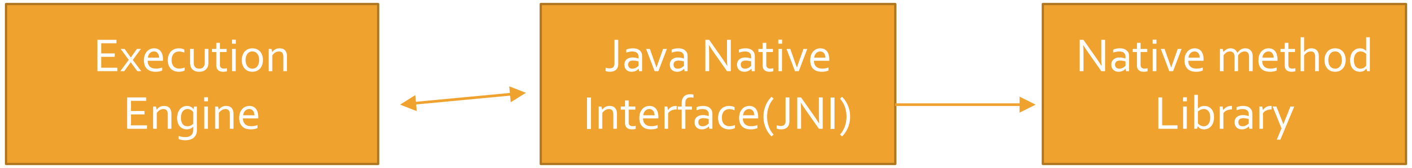
JIT(Just In Time) compiler Explanation

- **The main purpose of JIT compiler is to Improve the Performance.**
- **It compiles entire byte code and convert into the machine ode.**
- **Whenever interpreter sees repeated method calls JIT compiler starts working on this.**

- **Profiler:**
- It is responsible to identified the repeated method call(Hotspot).

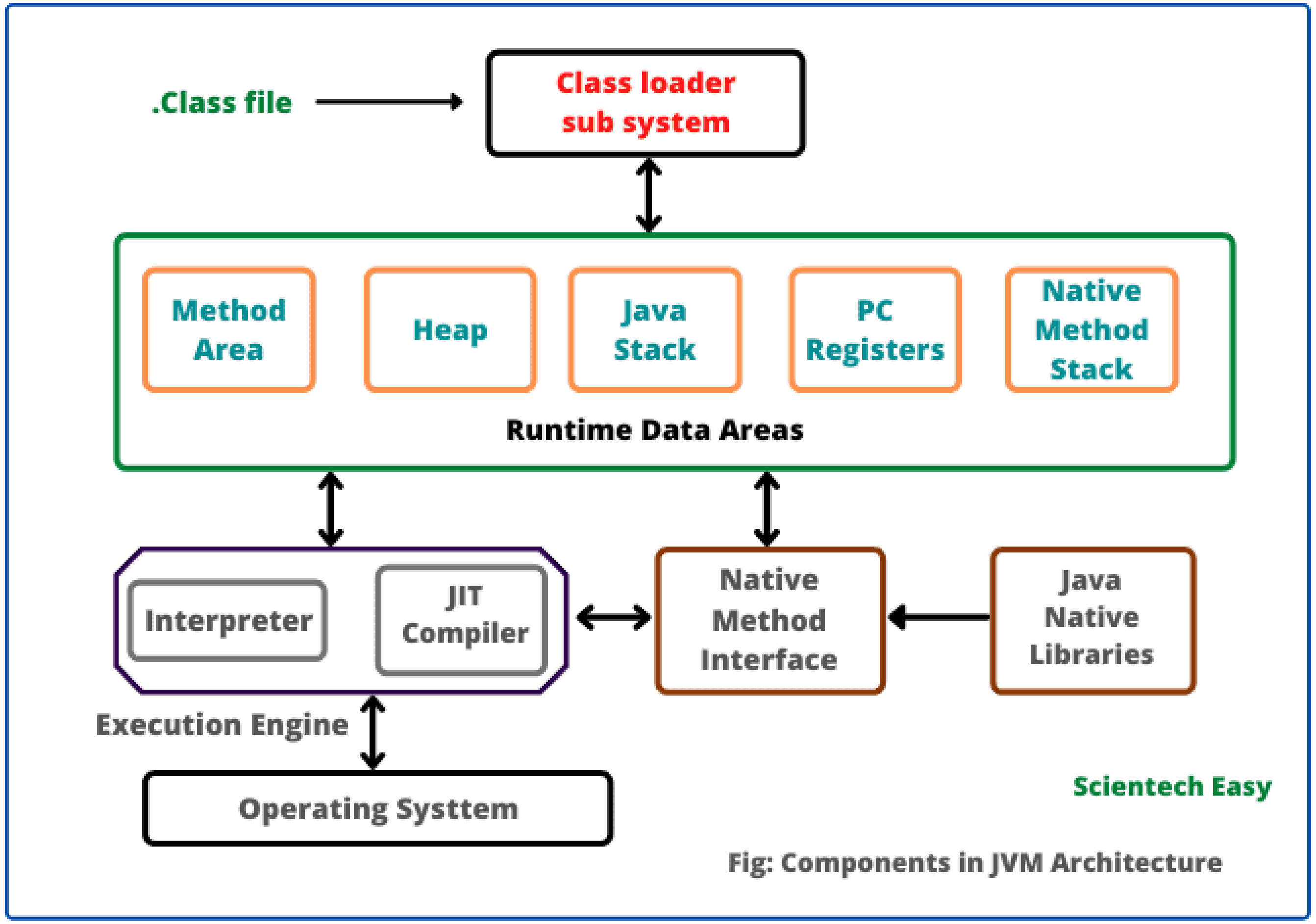
- **Other component:**
- **There are several other component like Garbage collector,Security Manager etc.**

Java Native Interface



- JNI interact with the Native Method Library and provides the native method library to execution engine.
- In other word you can say ,JNI is responsible to provide the native information of JVM.

JVM ARCHITECTURE FULL DIAGRAM



Scientech Easy

Fig: Components in JVM Architecture

JVM ARCHITECTURE