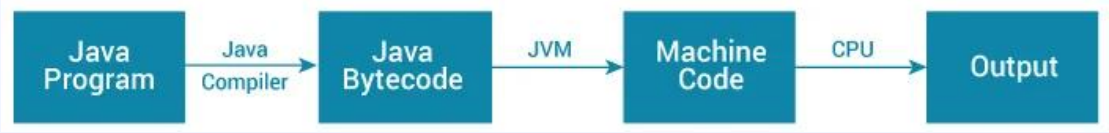
Programming is a way to instruct the computer to perform various task. It is a collaboration between humans and computers, in which humans create instructions for a computer to follow (code) in a language computers can understand. Thus we have programming languages to solve this issue.
Programming languages can be grouped into variety of categories.

- Machine language – A low level language that consist of 0's and 1's (binary). High level languages are compiled into machine code so the code can be executed by the computer.
- Procedural languages – This approach goes through a series of procedures before a program is executed on the computer. (For example, Cobol , Basic, C, Fortran, Pascal,Julia etc)
- Scripting languages – These languages often times don't need to be compiled but rather interpreted. Interpreted means an interpreter will read and execute the code instead of being compiled into machine code. (For example, JavaScript and PHP)
- Functional languages – This works with the idea of building complex programs through a collection of smaller functions. It is used in situations where we have to perform lots of different operations on the same set of data like ML(Machine Learning).
- Object-oriented languages – This works with the idea of building programs around collections of objects. This kind of language makes it easier to develop, debug, reuse and maintain software. (For example, Java and Python)

Java Follows procedural and object oriented

JVM, JRE and JDK



- The code written is java is human readable and is saved using the extension "**.java**" . This known as source code.
- Java compiler converts the source code into byte code with the extension "**.class**", This is known as byte code.

The byte code will not run directly on a system

JVM

- **Java Virtual Machine** is needed to run the byte code. A reason why Java is platform indepdent.
- The Java intepreter converts the byte code to machine code (0's and 1's)

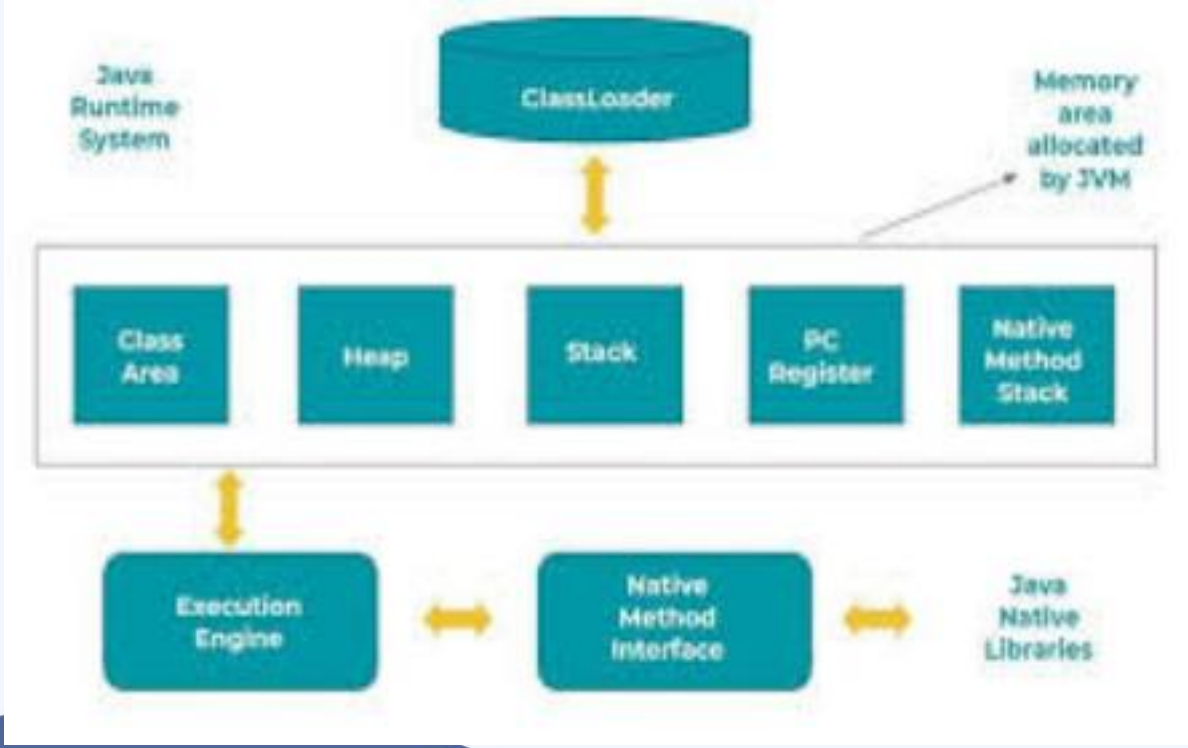
JRE

- **Java Runtime Environment** is the package which provides Java class libraries , JVM, and other components to run a Java program.
- If you need to run Java Programs, but not to develop them , JRE is what you need

JDK

- **Java Development Kit** is the software development kit required to develop applications in Java
- In addition to JRE , it contains compilers, JavaDocs , Debuggers, archiver, intepretor

Memory Management



Lets take a look at **heap** memory

- The heap is the main memory area used by JVM to store objects and their associated data.
- Examples include instances of classes, arrays , and objects created using the "new" keyword.
- The heap is divided into two parts: namely **Young** and **Old** Generation
- Young Generation is further divided into **Eden Space** and two S**urvivor Space**.
- When the Eden Space becomes full, a process is performed which is called minor GC(Garbage Collection). In this process the objects from the Eden Space are moved to one of the Survivor spaces. Objects which are not moved are garbage collected.The Survivor spaces stores the objects which survive the GC operation
- The **Old Generation** is the place where long-lived objects are moved to which have survived numurous minor GC's. This space can be still subjecte to a full GC but it is a more expensive operation than the minor GC

For example , let's say a program creates a lot of short-lived objects , These objects will be created in the Eden space and will be garbage collected during the minor GC's. But if there is a program which creates number of long-lived objects, such objects which represent a permanant data structure will be moved to Old Generation which will not be garbage collected until a full GC is performed.
JVM uses differernt algorithms and tuning parameters to manage the Young and Old Generation. The main objective is to provide a balance between GC efficiency and application performance. The heap size can be controlled using -Xmx and -Xms agruments while starting the JVM.

Lets take a look at **stack** memory

- The stack is a memory area which is used to store the execution context of a thread.
- Each thread has its own stack, it will contain information such as method calls, local variables and the address of the instruction currently being executed.

Refer: https://www.digitalocean.com/community/tutorials/java-heap-space-vs-stack-memory