Hi Friends,

In this video we will learn about the internal structure of JAVA. What are the components available in java.

Here we can see the Structure of Java.

On High level there is JDK and JDK contains JRE with devtools. And JRE contains JVM and Libraries.

So each container in this image has its own feature. Now we will see one by one.

JVM:

As we know Java code is compiled into Byte code and using interpreter byte code is converted into machine code.

JVM – java virtual machine.

Java Virtual Machine, or JVM, loads, verifies and executes Java bytecode. It is known as the interpreter and the JAVA program is executed in JVM.

SO JVM is responsible for converting bytecode to machine understandable code.

Libraries:

These libraries are just a collection of classes which are predefined in JAVA. And these classes we get when we install the Java.

JRE-

The Java Runtime Environment (JRE) is a software package which bundles the libraries (jars) and the Java Virtual Machine. With the help of libraries JVM will execute or run the java application.

So to execute any java program, you need JRE installed in that machine. With out JRE we can not run the program.

JDK-

In the image we saw that JDK contains JRE and development tools. We have seen what is the use of JRE.

development tools are used for developing, debugging, and monitoring Java applications.

For developing the java program we need JDK. And for running the java program we need JRE.

Compiler , jar creation are some of the tools available in development tools.

JVM Architecture.

Hi Friends in This video we will see the JVM architecture and what happens in JVM.

In this image you can see the different modules present in JVM. As you all know JVM will load verifies and execute byte code. And all this happens in step by step.

The first module in JVM is class loader.

The class loader is used for loading the class files. The three phases of this loader is loading, linking , initialization.

Loading-

When java program is executed at first java class file will be loaded and all the dependencies present in that class files will be loaded. So JVM will try to load all the files and its dependencies.

The dependencies which were defined in a class file will not present at one place. They might be from different modules. Based on this loading has again three phases.

For example IF JVM tries to load some abc.class dependency. The first bootstrap class loader will try to find the class in rt.jar which was present in JRE lib folder.

IF class is not found then next extension class loader will try to find the class in jre\lib\ext folder. Even if extension class loader didn’t find the class. Then next application class laoder will search all the jar files and classfiles in the folders provided in classpah environment variable of system.

After running all the loaders still the class is not found then classnotfoundexception will be thrown.

Linking.

After the class is loaded by the loader next linking is performed. In this phase the verifier will verify weather the generated bytecode is proper or not. IF verification fails then error will be generated.

And in this linking stage memory will be allotted to static variables and for all the methods present in class files.

Initialization

In this phase all the static variables will be assigned with the values defined and all the static blocks will be executed.

Next module is JVM memory. We will see this JVM memory in separate video. Because it need lot to discuss.

Next is execution engine

As we saw that the class loader will load the classes and the loaded byte code will be executed in execution engine. This execution engine will communicate with all the memory modules and execute the code.

Execution engine uses interpreter and Jit compiler to execute the code.

Interpreter will read the byte code and convert to machine and executes in a line by line manner. Here as it executed line line the same and common code will also be executed twice and performance if the system will be reduced.

Jit Compiler- jit compiler will consider the block of byte code and convert it into machine code and executed the code. It will not consider line by line as interpreter does and because of this performance if JIT compiler is better than interpreter.

In java Jit compiler is enabled by default. We can also disable the JIT compiler. IF JIT compiler is disables then jVM user interpreter to execute the code. But disabling jit compiler is not recommended.

Native method interface and native method libraires.

Native methods are basically the code which is not in JAVA. So If you want to run the native methods in JVM then we can use JNI and can run those I JVM.

JNI is an interface that allows Java to interact with code written in another language.