Assignment No -3

1 Explain the components of JDK

→

- The Java Development kit (JDK) is a software development environment which is used to develop Java applications and applets.
- The JDK contains a private Java virtual machine (JVM) and a few other resources such as a interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) etc. to complete the development of a Java Application.

Components of JDK

1) appletviewer : This tool is used to run and debug Java applets without a web browser.

2) apt : It is an annotation-processing tool

3) extcheck : It is a utility that detects JAR file conflicts

4) idlj : An IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.

5) jabswitch : It is a Java Access Bridge. Exposes assistive technologies on Microsoft Windows System

6) Java : The loader for java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler.

7) Javac : It specifies the java compiler, which converts source code into Java bytecode

8) Javadoc : The documentation generator, which automatically generates documentation from source code comments.

9) Jar : The specifies the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.

10) Javafxpackager : It is a tool to package and sign Javafx application

11) Jarsigner : the jar signing and verification tool.

12) Javah : the c header and stub

13) Javap : the class file disassembler

14) Javaws : the Java web start launcher for JNLP applications

15) Jconsole : Java Monitoring and Management console

16) Jdb : the debugger

17) Jhat : Java Heap Analysis Tool (experimental)

18) Jinfo : This utility gets configuration information from a running Java process or crash dump

19) Jmap : Oracle jmap - Memory Map- This utility outputs the memory map for java and can print shared object memory maps or Heap memory details of a given process or core dump

20) Jmc : Java Mission control

21) JPS : Java Virtual Machine Process Status Tool lists the ~~instrumental~~ instrumented HotSpot Java Virtual Machines (JVMs) on the target system

22) Jrunscript : Java command line script shell

23) Jstack : It is a utility that prints Java stack traces of Java threads

24) Jstat : Java virtual Machine statistics monitoring tool.

25) Jstatd : Jstat daemon

26) keytool : It is a tool for manipulating the keystore.

27) pack200 : JAR compression tool

28) Policytool : It specifies the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources

29) VisualVM : It is a visual tool integrating several command-line JDK tools and lightweight performance and memory profiling capabilities

30) wsimport : It generates portable JAX-WS artifacts for invoking a web service

31) xjc : It is the part of Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes

2) Difference between JDK, JVM and JRE

->

1) JDK (Java Development Kit):

- A software development kit for developing Java applications
- It includes tools for compiling, debugging and monitoring Java code
- Comes JRE (Java Runtime Environment) as an Integral ~~Java code~~ part
- Platform-dependent (different JDK for different platforms)
- Used for development purposes

2) JRE (Java Runtime Environment)

- A software package that provides the necessary components to run Java Applications
- It includes the JVM (Java Virtual Machine), class libraries, and other supporting files.
- Platform-dependent (different JRE's for different platforms)
- Used ~~for~~ to execute Java programs

3) JVM (Java Virtual Machine)

- An abstract machine that provides a runtime environment for executing Java bytecode
- Platform-independent (can run on multiple platforms)
- Specifies the requirements for JVM implementations.
- Responsible for loading, verifying, executing, and managing memory for Java code.

Key Differences:

- JDK is for development, JRE for execution and JVM is the runtime environment
- JDK includes development tools, JRE includes class libraries & supporting files and JVM includes runtime environment
- JDK & JRE are platform-dependent, while JVM is platform independent.

~~Hard~~

* What is the role of JVM in Java? & How does the JVM execute Java code?

→
- JVM (Java Virtual Machine) is an abstract machine.
- It is the engine that executes the Java code. It converts bytecode into machine code.

1) Java compiler produces code for a virtual machine
2) JVM compiled java code into byte code. This bytecode gets interpreted on machines
3) Between host system & Java source, Bytecode is an intermediate language
4) JVM is responsible for allocating a memory space.

* JVM Architecture ?

1) Class loader:

Classloader is a subsystem of JVM which is used to load the class files into the memory

2) Class Area [Method Area]

Class Area stores per-class structures such as the runtime constant pool, fields and methods, code for methods

3) Stack: [JVM Memory Management system]

Java Stack stores frames. It holds local variables and partial results and plays a part in method invocation and return.

Each thread has a private JVM stack, created at same time as thread.

A new frame is created each time a method is called. A frame is destroyed when its method call completes.

4) Native Method Stack

Native Method Stack contains all the native methods used in applications

5) Heap

Heap is the runtime data area in which objects are allocated and stored.

6) Program Counter Register:

PC register contains the address of Java Virtual Machine instruction currently being executed

3) Execution Engine

Execution Engine containg:

i) A virtual processor

ii) Interpreter : Read bytecode stream then execute the instructions

iii) Just-In-Time (JIT) Compiler:

- It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at same time, and hence reduces the amount of time needed for compilation.

Working of JVM:

JVM perform : 1) Loads the java code
2) Verifies the code
3) Executes the code
4) Provides Runtime Environment

Compiler: Java compiler converts high level java code into bytecode.

Interpreter : It converts bytecode into native machine code

Hence Java is both compiled & interpred language

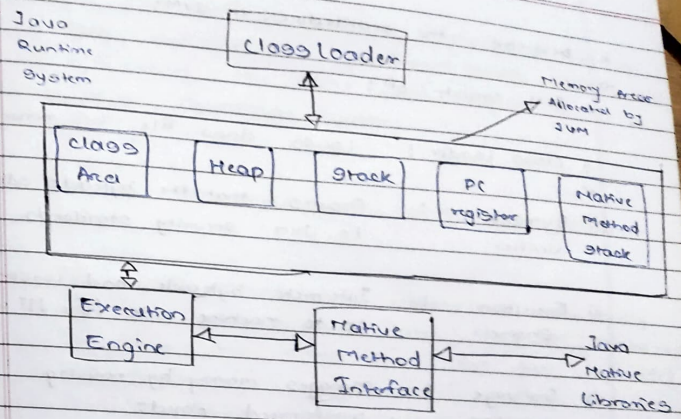

fig JVM Architecture

* What are the JIT compiler and its role in JVM? What is bytecode and why is it important in Java?

→

- The Just-In-Time compiler is a part of JVM that improves performance by compiling bytecode into native machine code at runtime, which can then be executed directly by the CPU.
- This reduces interpretation overhead and increases execution speed.
- Bytecode is an intermediate, platform-independent representation of Java code, generated after compilation.
- It allows Java to achieve platform independence, as any JVM can interpret bytecode, regardless of the underlying operating system.

* Describe the architecture of JVM:

→

JVM consist of:

1) Class Loader : Loads class file into memory

2) Bytecode Verifier : Ensures that the bytecode adheres to JVM security standards

3) Execution Engine : Interprets bytecode and convert it to machine code (via JIT compiler)

4) Garbage collector : Manages memory by removing unreferenced objects

5) Runtime Data Area : Includes the heap, stack, method areas, and other for storing variables, object and execution context

* How does Java achieve platform independence through the JVM?

→

- Java achieves platform independence through the use of bytecode.
- When Java code is compiled, it is compiled, it is converted into bytecode, which is platform-neutral,
- The JVM on each platform interprets or compiles this bytecode into platform-specific machine code, allowing the same Java program to run on any operating system.

8 What is the significance of the class loader in Java?

→

- The class loader is responsible dynamically loading Java classes into the JVM at runtime.
- It finds and loads the class file (bytecode) as needed.
- Java's class loader allows on-demand loading and the possibility of creating custom class loaders for specific tasks, such as loading classes from non-standard sources

Garbage Collection:

- Java's garbage collector [GC] automatically frees memory by removing objects that are no longer in use.
- It mainly focus on the heap, using algorithm like mark-and-sweep, generational GC, and others.
- GC helps in preventing memory leaks and ensures efficient memory usage.

Page

N What are the four access modifiers in Java, and how do they differ from each other?

→

Public : Accessible from any other class

Protected : Accessible within the same package and subclasses

Default (Package-Private) : Accessible only within the same package

Private : Accessible only within the same class

What is difference between public, protected, and default access modifier?

→

Public : visible everywhere.

Protected : Visible within the same package and to subclasses outside the package

Default (Package-Private) :

Visible only to classes in the same package. It is the default when no modifier is specified.

Classmate
Date
Page

Can you override a method with a different access modifier in a subclass?

→

- No, you cannot reduce the visibility of an overriding method in subclass.
- For eg, if a superclass method is protected, the overriding method in the subclass cannot be private, but it can be made public or retain protected visibility.

What is difference between protected and default (package-private) access?

→

protected : Allow access within the same package and also in subclasses, even if they are in different packages.

Default (Package-Private) : Access is restricted to the same package only, with no visibility to subclasses outside the package

Is it possible to make a class private in Java? If yes, where can it be done, and what are limitations?

→

- yes, a class can be private, but only for inner classes (nested classes).
- A top level class cannot be private or protected ; it can only be public or package private.

Q. Can a top-level class in Java be declared as protected or private? Why or why not?

→
- No, top-level classes cannot be declared as protected or private.
- Java enforces this because top-level classes must be accessible by the JVM or other classes, and restricting them would prevent this access.
- Only public or package-private access is allowed for top-level classes.

Q. What happens if you declare a variable or method as private in a class and try to access it from another class within the same package?

→
- A private variable or method is not accessible from any other class, even if it is in the same package.
- private members are restricted to the class they are declared in.

Q. Explain the concept of "package-private" or "default" access. How does it affect the visibility of class members?

→
- Package-private (also known as default access) means that class members are accessible to all other classes within the same package, but not outside it.
- It is the default access level when no access modifier is specified.
- It provided a balance between encapsulation and accessibility within the package.