Varun Verma 24060693 KHargar

NOTES

Assignment - 3

DATE 1

Q) Explain components of JDK.

It consists of 3 main components :=>

1) JRE (Java Runtime Environment)
provides the libraries, Java Virtual Machine, other components to run Java applications.

2) JVM (Java Virtual Machine) : executes java bytecode providing an environment for running java applications.

3) Development tools — includes tool like javac (compiler), java (launcher java apps), javadoc, jdb (java debugger).

Q) Differentiate JDK, JVM & JRE

JDK	JVM	JRE
Complete kit for JAV developers	An abstract machine that runs Java Bytecode	runtime enviro. for executing java apps
Used for developing java applications	Java code to run on any device with a JVM	Used by end-users to run java apps not for development

PAGE

Q) Role of JVM in Java ? & How does the JVM execute JAVA Code.

Java Virtual Machine is crucial in Java's "write once, run anywhere". It serves as an abstraction layer b/w Java code & underlying hardware, allowing Java apps to run on any device with JVM installed.

How JVM executes java code

Compilation: Java source code (.java files) is compiled by java compiler (javac) into bytecode (.class files), which is platform independent.

Class Loading: JVM loads the compiled bytecode into memory using classLoader.

Bytecode Verification: JVM verifies bytecode to ensure it is safe.

Execution: JVM executes bytecode via the Just-in-Time (JIT) compiler, which converts bytecode into native machine code at runtime for faster execution

Memory Management: JVM manages memory through automatic Garbage Collection, which reclaims memory used by objects no longer in use.

Q) Explain memory management system of JVM.

NOTES DATE

1) Heap

Purpose: stores objs & arrays.

Structure

Young generation: where new objs are allocated. It is further divided into

Eden space: where most new objs are intially created.

Survivor space: holds objs that have survived garbage collection in Eden space

Old generation: store objs that have survived multiple garbage collection cycle in young generation.

Garbage Collection: Automatic memory management system that removes objs no longer in use.

2) Stack

Purpose- stores method call names, including local variables method params, & return addresses.

Structure- Each thread has its own thread stack, and each method call creates a new frame on the stack.

Lifecycle- managed in L.I.F.O manner. frames are removed when method completes.

NOTES DATE

3) Method Area

Purpose - stores class level data
Structure - includes runtime constant pool, method data, field data
Garbage Collection - this area is also subjected to garbage collection particularly for classes & interned strings

4) Program Counter -

Purpose - Holds address of currently executing JVM instruction for each thread.
Structure - Each thread has its own PC register.

5) Native Method Stack:

Purpose - holds states of native (Non-JAVA) method calls used by JVM
Structure - Each thread has its own native method stack.

6) Garbage Collection process -

Minor GC - cleans the young generation

Major GC - cleans the entire heap.

NOTES DATE

Q) Describe architecture of JVM.

It is designed to ~~to~~ provide an envirn. for executing JAVA bytecode in platform indepedent manner.

1) Class Loader Subsystem -
loads class file into memory, it is responsible for locating, loading & initialising classes & interfaces.

2) Runtime data Area -
consits of Heap, Method Area, Stack, Program Counter Register, Native method stack

3) Execution Engine -
executes the bytecode loaded into memory by class Loader.

4) Native method Interface (~~NMI~~ JNI) -
provides a framework that allows JAVA Code running in JVM to call & be called by native apps (programs written in C, C++).

5) Native libraries -
Contains native code libraries required for interfacing with the underlying O.S.

6) JVM Memory Management -
It manages memory allocation & deallocation

PAGE

NOTES

DATE

optimising the use of system resources through GC, stack management, heap management.

Q) How does Java achieve platform independence through the JVM?

→ It achieves the same through the use of JVM & bytecode.

1) Compilation to Bytecode which is stored in .class files.
2) JVM as an abstraction layer (available for many OS)
3) Platform independence in practise.

Q) What are JIT compiler & its role in JVM
What is Bytecode & its importance.

JIT (Just in Time Compiler)

Function

The JIT compiler is part of JVM that optimises the execution of Bytecode by compiling it into native machine code at runtime.

NOTES

DATE

Bytecode and its importance

It is an intermediate platform independent code generated by JAVA compiler from JAVA source code. It is stored in .class file

Importance

1) Platform independent
2) Security
3) Portability

Q) what are four access-modifiers & how they differ from each other.

1) Public A.M –
The class, method or variable is accessible from any other class.

2) Protected –
The class, method or variable is accessible within the same package & by subclass, even if they are in different packages.

3) Default (Package level private) –
(C,M,V) only accessible within same package. package private

4) Private
(C,M,V) accessible only within the class

PAGE

NOTES DATE

created.

Q) Diffo Can you override a method with a different access modifier in a subclass? For eg- can a protected method in a superclass be be overridden with a private method in subclass. Explain.

→ No.
Because doing so ~~with~~ would reduce the accessibility of the method, which violates the principle of substitutability.

Q) Difference b/w protected & default (package-private) access.

→ Protected –
The member is accessible within the same package & also by the subclasses, even if they are in different packages. This allows subclassing outside the package while still restricting general access

Default (Package-private)
member is only accessible within the same package. It cannot be accessed from classes outside the package, even if they are subclasses.

NOTES DATE

Q) Is it possible to make a class private in JAVA? If yes where it can be done, & what are the limitations?

Yes, it is possible but it can only be done with nested classes

where it can be done

A private class can be defined within another class (a top level class cannot be private)

Limitations

- private class is only accessible only within the outer class that contains it
- It cannot be accessed directly from outside the outer class

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

If you declare a variable or method as private in a class & try to access it from another class within the same package, you will encounter a * compilation error.

* The private access modifier restricts access to the members to the class in which it is declared.

Error

PAGE

When another class tries to access the private member, the compiler will throw an error indicating that member isn't visible.

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

→ Concept

When a class member is declared without an access modifier, it is said to have package-private access.

Visibility of Class Members

* Within the same package -
The member is fully accessible to any other class within the same package.
* Outside the Package -
The member is completely inaccessible to any class outside the package, including subclasses in other packages.