Assignment 2

Q. Explain the Components of the JDK.

Ans → a part

(1) Java Virtual Machine (JVM)

(1) Java Development Tools

→ javac
→ Java
→ javap
→ jstack
→ jdb

javac :- it is compiler converts java source code (.java files) into bytecode (.class files) that can be executed by the Java Virtual Machine (JVM).

javap :- it is part of the Java Development kit (JDK) and it known as the Java class file Disassembler. Its primary purpose is to provide detailed information about the contents of compiled Java bytecode files (.class files).

syntax : javap class_name

Displaying Detailed Bytecode information.

→ javac -c classname.

jstack

→ it is used to generate a thread dump for a given java Virtual Machine (JVM) process).

→ Command :-
jstack <pid>

→ A thread dump is a snapshort of all the threads that are currently running in a java Application

→ it is perticularly used for diagnosing issues related to thread contention, deadlock, or hight cpu usage.

jdp :- java Discovery protocal is a feature in the Java development kit that enable java processes to discover other running java processes on the same network or machine

→ used → distributed systems or environments where multiple java process need to communicate or be aware of each other.

(2) Java API Documentation
it provide detail information about class, methods, and other elements in code

- javadoc

(3) Java API Libraries
rt.jar

→ Jar is used to bundle multiple files into a single archive file (usually with a .jar extens

JAR file can contain Java class files, metadata, resource (like images and properties files), and more. They are comnly used to ~~distrute~~ distribute java application and Libraryes.

(4) Execution Environment.
→ Java Virtual Machine (JVM)
→ it is the engine that runs java Applications. it reads and executes the Compile bycodes.
→ The JVM is a platform Independent, means it allows Java programs to run on any device or OS that has a Compatible JVM implementation.

(5) Code Samples.
→ src.zip.
→ it contains the source code for the java standar library.
→ it provide for reference and educational purpose, allowing developers to explore how the java standard class and method are implements.

→ Location :- c:\programfiles\Java\jdk-1.8 ... Find... in tha directory.

3.55

DOMS | Page No.
Date / /

Q2 Difference bet^ JDK, JVM, JRE

→ JDK → writing Java applets and applications needs development tools like JDK.

→ Java developers are initially presented with two JDK tools, Java and javac.

→ Both are run from the command prompt

→ It's easy for both new and experienced programmers to get starts.

JRE → JRE is an acronym for Java Runtime Environment.

→ it is also written as Java RTE

→ The Java Runtime Environment is a set of software tools.

→ It is physically exists.

→ It contains a set libraries + other files that JVM uses at runtime.

JVM →

→ JVM is a engine that provide runtime environment to drive the Java code or applications.

→ It converts Java byte code into machines language

→ Java compiler produces code for a virtual machine known as Java Virtual Machine.

Q8 What is the role of the JVM in java & How does the JVM execute Java code?

→ platform independance
→ Bytecode execution
→ memory Management
→ Security
→ Multithreading.

JVM Execution Java code

① Compilation

Java source code (.java) it is compiler (javac) into bytecode (.class) file. this bytecode is the platform-independent and can be executed any system with a Compatible JVM.

② class loading.

JVM starts by loading the required class files This is done by the class Loader subsystem which loads, Liks, and initializes classes dynamically during runtime.

③ Bytecode verification.

Once the bytecode is loaded, the JVM bytecode verifier cheaks it to ensure it doen't violate java's security Constraints or try to perform illegal operations. The step is crucial for security and stability

④ Execution

JVM executions the bytecode using either an Interpreter or a just-in Time (JIT) Compiler.

Date / /

Interpreter – Converts bytecode into machine code line by line at runtime. it's simple but slower becouse each instruction is interpreted every time it's encountered.

~~JIT~~ JIJ Compilar – Convert entire sections of bytecode into native machine code on the fly, which the CPU can execute directly.

⑤ Runtime Environment :–

JVM provides a runtime environment, which includes garbage collector that automatically manage memory by cleaning up unused objects, and a runtime stack for managing method call and Local variable.

⑥ Execution Engine

This is a core component that actually execute byte code it consists of the interpreter and the JIT compiler, as well as native method interface that allow java interact with native libraries written in other Language.

Q4. Explain the memory management system of the JVM.

→

Memory Management process.

→ ① Memory allocation :

JVM allocats memory for object, array, classes as needed.

→ ② memory Deallocation.

JVM reclaims memory acc ped by the object no longer in wed syarbey collections

(3) Garbage collection.
JVM periodically identifiers and reclaims memory occupied by unreactribse object
(4) Memory compaction.
Jvm periodically identifiers and reclaims memory occupied by unractati
(4) Memory Compaction
JVM Compact memory to reduce fragmentation and improve performance

(5) what are the JFT Compiler and its role in the JVM? what is the bytecode and why is it important for Java?

→ (1) Improve perform by compiling frequently executed code into native machine code
(2) optimize code for the specfice CPU architecture
(3) feduce interpretation overhead.

Importance of Bytecde.
(1) platform indepen dence.
(2) Effective interpretation
(3) Security
(4) Dynamic loading.
(5) Compactness.

Bytecode is impertant for Jave
(1) write once, fan anywhere.
(2) effcient Execution
(3) security.
(4) Dynamic Nature

Q6 A Describe the architecture of the JVM

→ ① class loader subsystem
② four runtime Data areas
- Method area
- Heap
- Stack
- Nativ
- PC register
③ Execution engine
- Intepreter
- Just-in-time (JFT)
④ Native method interface
⑤ & Security management
⑥ Garbage collecter
⑦ Thread management
⑧ function environmention.
⑨ Native libraies.

⑦ How does Java active platform independance through the JVM ?

→ Java archieves platform independance through in the JVM in the following ways.
① Bytecode
② JVM interpretation
③ JVM runtime environment
④ Dynamic loading.
⑤ Native method interface
⑥ platform independent Data types.
⑦ JVM's platform independant.

Memory

* Here's a step-by-step explanation:
① Java code is compiled into bycode (class files)

(2) The JVM loads the bytecode into memory.
(3) The JVM interpret the bytecode, converting it into platform specific machine code
(4) JVM executed the machine code, using platform-independent libraries and APIs
(5) The JVM dynamically loads classes and libraries as needed.
(6) The JVM memory memory allocation and dellocation ensuring platform independence

98. what is the significance of the class loader in java? what is the process of garbage collection Java?

→ class loader:

The class loader in java is responsible for loading classes and their dependencies into the java virtual machine (JVM) It's significance. includes:

(1) Dynamic loading.
(2) Namespace management
(3) Security
(4) flexibility.

Garbage collection
(1) marking.
(2) ~~Normal~~ Normal deletion.
(3) compacting.

(1) Generational Garbage collection:

~~Divide~~ divides object into generation. based on their lifespan.

② Concurent garbage collection:
- funs garbage collection concunetly with the application

Benefits of garbage collection include:
① Memory safity: prevents memory leaks and dangling pointers.
② Feduced memory usage:
frees up memory.
③ Improved performance:
fedeces the need for manual memory management.

Q ⑨ whater the four acess modifiers in javas and has do they differ form each other.

→ The four acces modifiers in java are
① public: Accesible from anywher, both with and outside the class, parkage, or subclass
② Private:
Accessible only within the same class, not from outside the class or its subclasses
③ protected:
Accessible within the same class its subclassed, & clases in the same packagu

Here a summary of their differency:
① Scope: public has the widest scope, while private has the hamowest

② Accessibility:
public is accessible from anywhere, while private is only accesible within the same class.

③ Inheritance:
protected memory are inheritance by subclass while private numbers are not

⑩ what is the difference between public, protected and default access modifiers.

→ ① public:
Accessible from anywhere (same class, same package, different package, sub class).
• No restrictions on access.

② Protected.
Accessible within the same class and it's subclasses.
• Accessible within the ~~same class and its~~ same package that not from outside the package.
• more restrictive than public less restrictive than default and private.

③ Default modifiers:
⊙ Accessible whin the same class and same package.
⊙ Not accesible from outside the package or from subclasses.
⊙ more restrictive that protected less restrick than private

⑪ Can you override a method with a different access modifier in a subclass for example can a proted method in a subsystem be overriding within a private method in a Subclass!

Explain

→ A public method can do be overridden within public method. with a less accessible acess mo.

→ A protected method can be orriden with a proted methode.

→ However you cannot override a method with a less accessible access modifier for example. A public method cannot be overrriden with a protected, default or private method. A protected method cannot be overridden with default or private methode.

(12) what is the difference betⁿ protected and defaults (package - private) access?

→ proteded. :-

→ Accessible within the same clas.

→ Accessible within subclass

→ Accessible within the same package

Default (package - private):

- Accesible within the same class
- Accessible within the same package
- Not accessible from subclass outside the packa

(18) Is it possible to make a class private in Java? If yes, where can it be done and what are the limitation?

→ However:

you can declared a nested class (a class inside another class) as private Here's an example

```
public class outerclass {
  private class innerclass {
// Innerclass is only accessitole within outclass
```

}

In this code, inner class is only accessible whin outer class and not from outside

limitations:

- A private nested class can't be accessed from outside the enclosing class
- A private nested class can't be used as a type for method parameters or return type

Q (14) Can a top level class in java be as protected or private? why or why not?

→ Here's why:

protected access modifier: A protected class would imply that it's accessible from subclass but since, top-level class can't be subclassed from outside its package this's access modifier doesn't make sense.

- private access modifier: A private class would imply that it's only accessible within ~~imply that it's only~~ o wither, the same class, but since a top level class is already a separate entity, declarring it private would mean it can't accessed at all which is not usefull.

Here's a summary.

public: Accessible from any where.

package private (default): Assible whin the same package

- protected : Not allowed for loop top-level classes
- private : Not allowed for top level classes.

(15) what happens if you if declared variable or method as private in class a try to classes it from another class within within the same package.

→ if you declared a variable or method as private in a class you cannot access it directly from another class, even if it's within the same package.

private member members are only accessible within the same & class. not from outside the class, including. class in the same package. To access the private numbers, you would need to.

① make the member public or package - private (defaults access).

② provide a public getter or seter method in the original class to access the ~~im~~ private membe

③ use interitance tif- and access the member through a protected accesor.

(16) Explain the concept of package - private or default access ~~the~~ Hence does it effect the visibility of class members?

→ Package:- private access is useful when implementa details within a package.

- you want to allow classes within the same package to shere ~~se~~ re curses or or functionlity.
- you want to hide internal implementation details from exterred packages

Effect desirability:

- visible to classes not within the same package.
- not visible to classes in other package.
- not visible do subclasses in other package.