Assignment 3

Q 1. What is Java?
ANS :
Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than 3 billion devices run Java.

It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more!

Q 2 . What is a package in Java? List down various advantages of packages.
ANS :
A java package is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

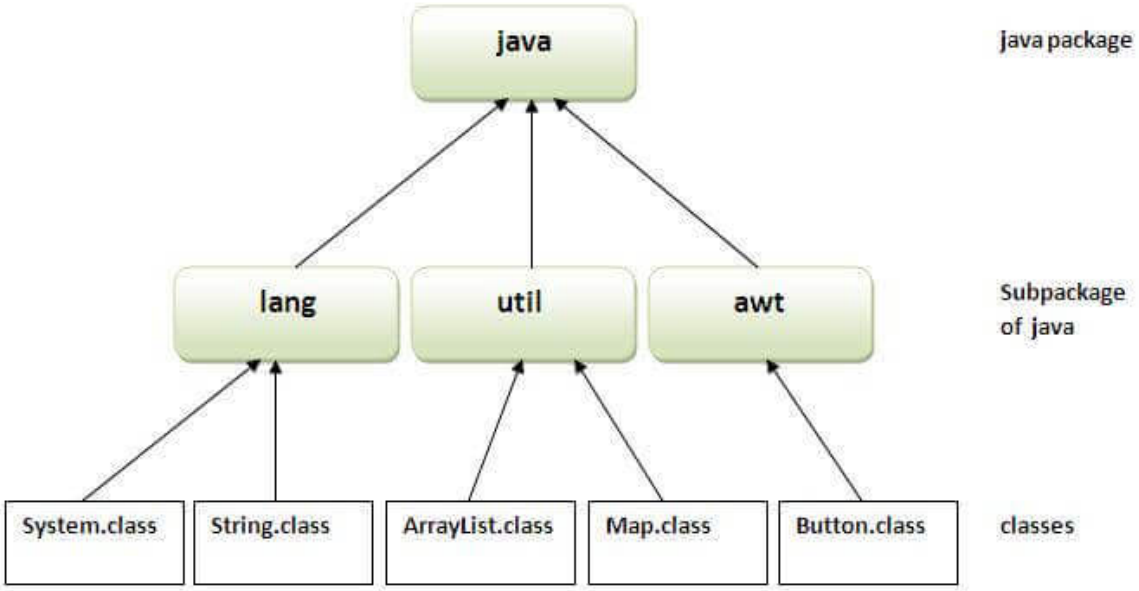
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.



Q 3 .Explain JDK, JRE and JVM?

ANS :

JVM :

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

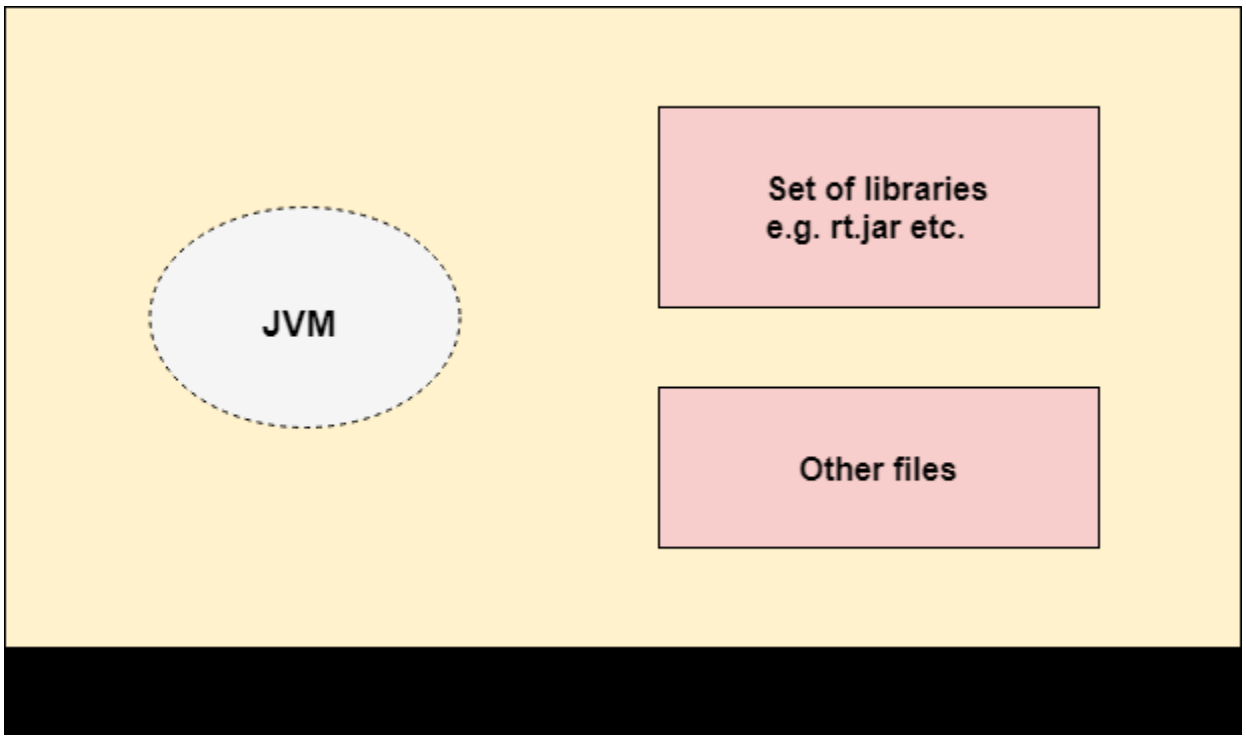
JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

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 which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



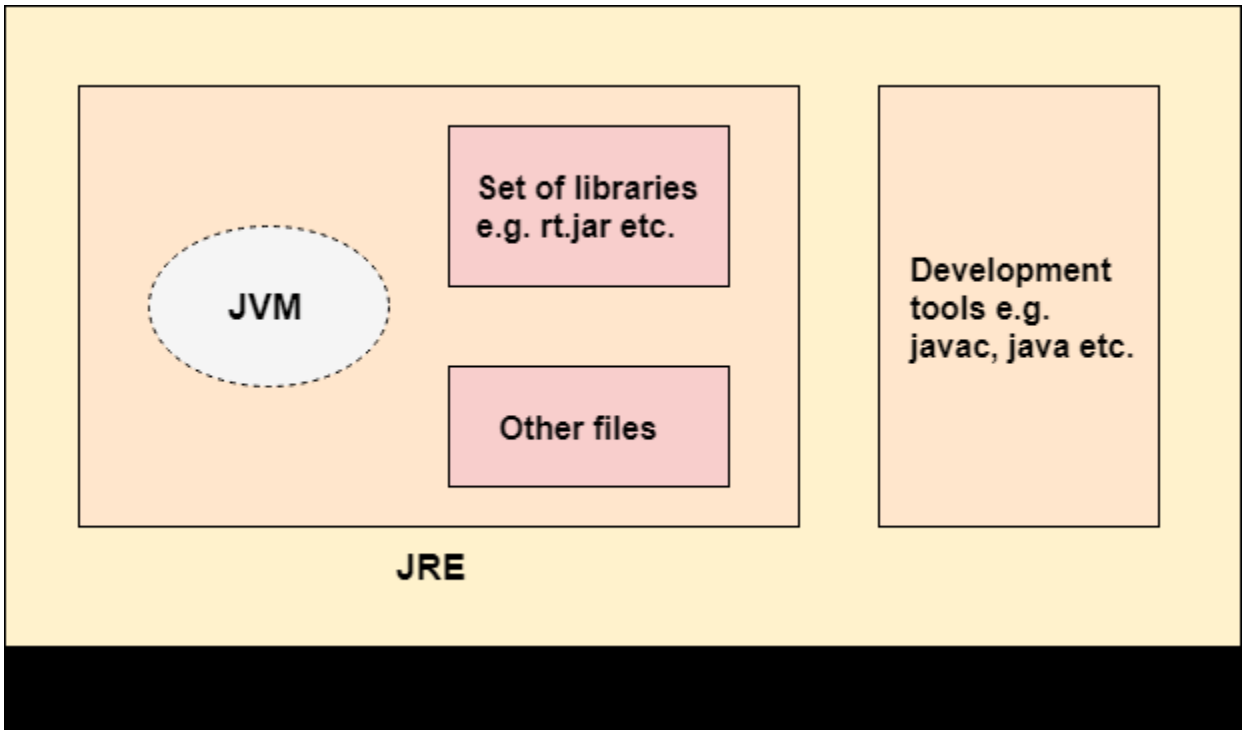
JDK :

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

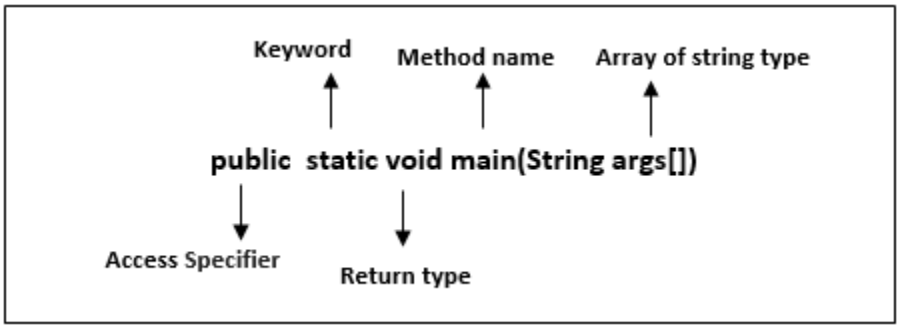
JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Q 4. Explain public static void main(String args[]) in Java.
ANS :
The main() is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program.
Syntax :



public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method static by using the keyword static. We should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the main() method.

void: In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

main(): It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

String args[]: The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

Example :

```
class  Demo
{
        static                  //static block
        {
                System.out.println("Static block");
        }
        public static void main(String args[])  //static method
        {
                System.out.println("Static method");
        }
}
```

Q 5 .What are the differences between C++ and Java?
ANS :

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.

Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of c programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interface in java.
Operator Overloading	C++ supports operating overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write a pointer program in C++.	Java supports pointers internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses a compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses a compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.

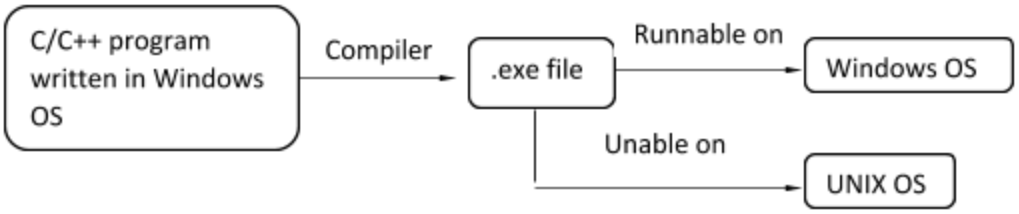
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

Q 6.Why Java is platform independent?
ANS :

Java is platform-independent because it does not depend on any type of platform. Hence, Java is platform-independent language. In Java, programs are compiled into byte code and that byte code is platform-independent.

same java program can run on any operating system. If you write a code in Java, then the program will be sent to the compiler for compilation. The compiler creates a . class file that is readable for JVM.(java virtual machine).and every Operating System have separate JVM that is capable to read .class file.

JAVA is Platform-Independent but JVM is platform dependent.



Q 7 . What are wrapper classes in Java?
ANS :

Wrapper classes provide a way to use primitive data types (int, boolean, etc..) as objects.

The table below shows the primitive type and the equivalent wrapper class:

Primitive Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Sometimes you must use wrapper classes, for example when working with Collection objects, such as ArrayList, where primitive types cannot be used

Example

```
ArrayList<int> myNumbers = new ArrayList<int>(); // Invalid

ArrayList<Integer> myNumbers = new ArrayList<Integer>(); // Valid
```

Creating Wrapper Objects

To create a wrapper object, use the wrapper class instead of the primitive type. To get the value, you can just print the object:

Example

```
public class Main {
  public static void main(String[] args) {
    Integer myInt = 5;
    Double myDouble = 5.99;
    Character myChar = 'A';
    System.out.println(myInt);
    System.out.println(myDouble);
    System.out.println(myChar);
  }
}
```

Also there are some methods to convert wrapper object to string i.e toString()
Also having method to get value associated to corrosponding Wrapper object i.e intValue(), byteValue(), shortValue(), longValue(), floatValue(), doubleValue(), charValue(), booleanValue() etc.

Q 8 .Why pointers are not used in Java?
ANS :
Reasons of not to using pointers in java

1. Because pointers are unsafe. Java uses reference types to hide pointers and programmers feel easier to deal with reference types without pointers. This is why Java and C# shine.
2. Java is created for simple use.But by adding pointer it is too complex for a programmer.
3. Memory allocation is done automaticaly it is managed by Jvm. So to avoid direct access to memory by user pointers dont allowed in java
4. Java Language does not use pointer for the reason that it works on Internet. Applets are used on the internet. and Pointers are used to identify the address of variables,methods, and therefore even big programmer can find out the secrets of other user on the internet using pointer. so if there could be pointer in java, it could be harmful for leakage of the important information.

Q 9.List some features of Java?
ANS :

Following are the notable features of Java:

1. Object Oriented

   In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

2. Platform Independent

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

3. Simple

   Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

4. Secure

   With Java's secure feature it enables to develop virus-free, tamper-free systems. techniques are based on public-key encryption.

5. Architecture-neutral

   Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

6. Portable

   Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

7. Robust

   Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

8. Multithreaded

   With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

9. Interpreted

   Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

10. High Performance

    With the use of Just-In-Time compilers, Java enables high performance.

11. Distributed

Java is designed for the distributed environment of the internet.

12. Dynamic

    Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

Q 10.Why is Java Architectural Neutral?

ANS:

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Q 11.How Java enabled High Performance?

ANS :

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

At run time, the JVM loads the class files and performs the appropriate computation.

- Due to the additional processor and memory usage during interpretation, a Java application performs more slowly than a native application.

Java enabled High performance by introducing JIT- Just In Time compiler , JIT helps the compiler to compile the code On demand basis i.e which ever method is called only that method block will get compiled making compilation fast and time-efficient. This makes the java delivering high performance.

Q 12 . Why Java is considered dynamic?

ANS:

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

Q 13. What is Java Virtual Machine and how it is considered in context of Java's platform independent feature?

ANS:

In Java, the main point here is that the JVM depends on the operating system – so if you are running Mac OS X you will have a different JVM than if you are running Windows or some other operating system. This fact can be verified by trying to download the JVM for your particular machine – when trying to download it, you will be given a list of JVMs corresponding to different operating systems, and you will obviously pick whichever JVM is targeted for the operating system that you are running. So we can conclude that JVM is platform-dependent and it is the reason why Java is able to become "Platform Independent".

- In the case of Java, it is the magic of Bytecode that makes it platform independent.
- This adds to an important feature in the JAVA language termed as portability. Every system has its own JVM which gets installed automatically when the jdk software is installed. For every operating system separate JVM is available which is capable to read the .class file or byte code.
- An important point to be noted is that while JAVA is platform-independent language, the JVM is platform-dependent. Different JVM is designed for different OS and byte code is able to run on different OS.

Q 14 . List two Java IDE's?

ANS :

Because of this popularity, Java has many IDE's that you can use. These IDE's or Integrated Development Environment provide immense help in the application development process. By using them, you can save time and effort as well as create a standard development process in your team or company. While the most popular Java IDE's in the world are Eclipse, IntelliJ IDEA, NetBeans, etc. there are many other IDE's that you can use according to your specifications.

So check out the Best Java IDE's in this article so that you can decide on the one you wish to use based on their features and your needs.

1. Eclipse

Eclipse is a Java IDE that is one of the 3 biggest and most popular IDE's in the world. It was written mostly in Java but it can also be used to develop applications in other programming languages apart from Java using plug-ins. Some of the features of Eclipse are as follows:

- PDE (Plugin Development Environment) is available in Eclipse for Java programmers that want to create specific functionalities in their applications.
- Eclipse flaunts powerful tools for the various processes in application development such as charting, modeling, reporting, testing, etc. so that Java developers can develop the application as fast as possible.
- Eclipse can also be used to create various mathematical documents with LaTeX using the TeXlipse plug-in as well as packages for the Mathematica software.
- Eclipse can be used on platforms like Linux, macOS, Solaris and Windows.

2. NetBeans

NetBeans is a Java IDE that is one of the 3 biggest and most popular IDE's in the world. This is an open-source IDE that allows Java programmers to build various applications using module sets. Some of the features of NetBeans are as follows:

- NetBeans is available for a variety of operating systems like Windows, Linux, macOS, Solaris, etc. It is also available in a feature-limited OS-independent version.
- It is very easy to create custom software applications using NetBeans as it highlights Java code syntactically as well as semantically. Also, there are many tools that help in writing bug-free code.
- While NetBeans is primarily a Java IDE, it has extensions for working in other programming languages such as C, C++, PHP, HTML5, JavaScript, etc.
- NetBeans can be used on platforms like Linux, macOS, Solaris and Windows.

Q 15 . Why Java is called as "Platform" ?

ANS :

Java (with a capital J) is a platform for application development. A platform is a loosely defined computer industry buzzword that typically means some combination of hardware and system software that will mostly run all the same software. For instance PowerMacs running Mac OS 9.2 would be one platform. DEC Alphas running Windows NT would be another.

There's another problem with distributing executable programs from web pages. Computer programs are very closely tied to the specific hardware and operating system

they run. A Windows program will not run on a computer that only runs DOS. A Mac application can't run on a Unix workstation. VMS code can't be executed on an IBM mainframe, and so on. Therefore major commercial applications like Microsoft Word or Netscape have to be written almost independently for all the different platforms they run on. Netscape is one of the most cross-platform of major applications, and it still only runs on a minority of platforms.

Java solves the problem of platform-independence by using byte code. The Java compiler does not produce native executable code for a particular machine like a C compiler would. Instead it produces a special format called byte code. Java byte code written in hexadecimal, byte by byte, looks like this:

CA FE BA BE 00 03 00 2D 00 3E 08 00 3B 08 00 01 08 00 20 08

This looks a lot like machine language, but unlike machine language Java byte code is exactly the same on every platform. This byte code fragment means the same thing on a Solaris workstation as it does on a Macintosh PowerBook. Java programs that have been compiled into byte code still need an interpreter to execute them on any given platform. The interpreter reads the byte code and translates it into the native language of the host machine on the fly. The most common such interpreter is Sun's program java (with a little j). Since the byte code is completely platform independent, only the interpreter and a few native libraries need to be ported to get Java to run on a new computer or operating system. The rest of the runtime environment including the compiler and most of the class libraries are written in Java.

All these pieces, the javac compiler, the java interpreter, the Java programming language, and more are collectively referred to as Java.

Q 16.Is Java Pure-Object oriented Language ?

ANS :

Java language is not a Pure Object Oriented Language as it contain these properties:Primitive Data Type ex. int, long, bool, float, char, etc as Objects: Smalltalk is a “pure” object-oriented programming language unlike Java and C++ as there is no difference between values which are objects and values which are primitive types. In Smalltalk, primitive values such as integers, booleans and characters are also objects.

In Java, we have predefined types as non-objects (primitive types).

```
int a = 5;

System.out.print(a);
```

- The static keyword: When we declares a class as static then it can be used without the use of an object in Java. If we are using static function or static

variable then we can't call that function or variable by using dot(.) or class object defying object oriented feature.

Wrapper Class: Wrapper class provides the mechanism to convert primitive into object and object into primitive. In Java, you can use Integer, Float etc. instead of int, float etc. We can communicate with objects without calling their methods. ex. using arithmetic operators.
String s1 = "ABC" + "A" ;

- Even using Wrapper classes does not make Java a pure OOP language, as internally it will use the operations like Unboxing and Autoboxing. So if you create instead of int Integer and do any mathematical operation on it, under the hoods Java is going to use primitive type int only.

```
public class BoxingExample

{

   public static void main(String[] args)

   {

        Integer i = new Integer(10);

        Integer j = new Integer(20);

        Integer k = new Integer(i.intValue() + j.intValue());

        System.out.println("Output: "+ k);

   }

}
```

- In the above code, there are 2 problems where Java fails to work as pure OOP:
    1. While creating Integer class you are using primitive type "int" i.e. numbers 10, 20.
    2. While doing addition Java is using primitive type "int".

Q 17 .Which version of java have u learned? Name some of the new features added to it.

ANS :

Java 11 Features

Java 11 (released on September 2018) includes many important and useful updates. Let's see the new features and improvements, it brings for developers and architects.

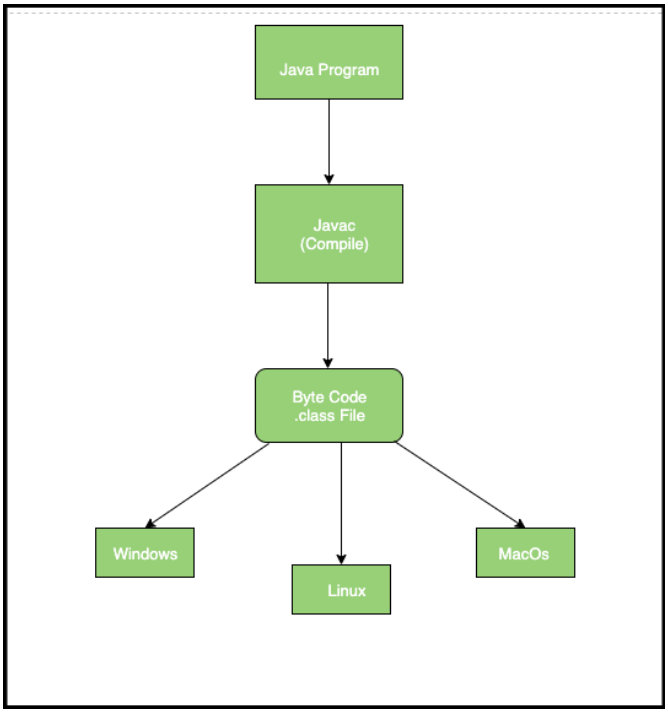
- HTTP Client API
- Launch Single-File Programs Without Compilation
- String API Changes
- Collection.toArray(IntFunction)
- Files.readString() and Files.writeString()
- Optional.isEmpty()

Q 18.What gives Java its 'write once and run anywhere' nature?

ANS :

JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in Java code. JVM is a part of the JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.



Q 19.Difference between path and classpath.
ANS :
PATH :

a) An environment variable which is used by the operating system to find the executables.

b) PATH is nothing but setting up an environment for operating system. Operating System will look in this PATH for executables.

c) Refers to the system

CLASSPATH :

a) An environment variable which is used by the Java compiler to find the path, of classes i.e in J2EE we give the path of jar files.

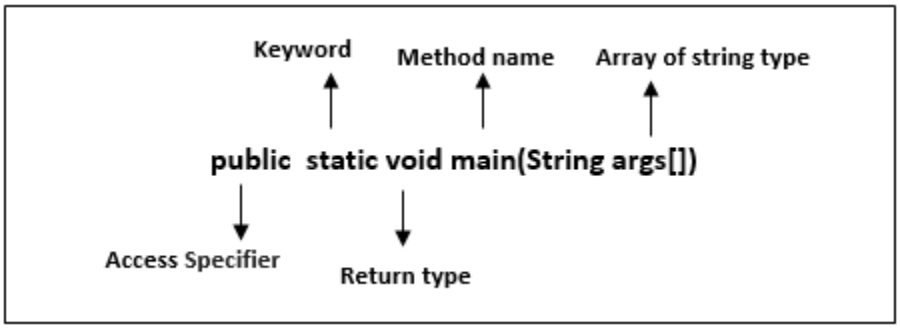
b) Classpath is nothing but setting up the environment for Java. Java will use to find compiled classes.

c) Refers to the Developing Environment.

Q 20.What is the signature of main function in java ?

ANS :

The main() is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program. The syntax of the main() method is:



public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method static by using the keyword static. We should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the main() method.

void: In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

main(): It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

String args[]: The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

Q 21.What is the difference betweenJDK and JRE?

ANS :

Sr. No	Key	JDK	JRE
1	Definition	JDK (Java Development Kit) is a software development kit to develop applications in Java. In addition to JRE, JDK also contains number of development tools (compilers, JavaDoc, Java Debugger etc.).	JRE (Java Runtime Environment) is the implementation of JVM and is defined as a software package that provides Java class libraries, along with Java Virtual Machine (JVM), and other components to run applications written in Java programming.
2	Prime functionality	JDK is primarily used for code execution and has prime functionality of development.	On other hand JRE is majorly responsible for creating environment for code execution.
3	Platform Independence	JDK is platform dependent i.e for different platforms different JDK required.	Like of JDK JRE is also platform dependent.

4	Tools	As JDK is responsible for prime development so it contains tools for developing, debugging and monitoring java application.	On other hand JRE does not contain tools such as compiler or debugger etc. Rather it contains class libraries and other supporting files that JVM requires to run the program.
5	Implementation	JDK = Java Runtime Environment (JRE) + Development tools	JRE = Java Virtual Machine (JVM) + Libraries to run the application

Q 22 .What is JVM ? What it does?

ANS :

It Is:

1. A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
2. An implementation Its implementation is known as JRE (Java Runtime Environment).
3. Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

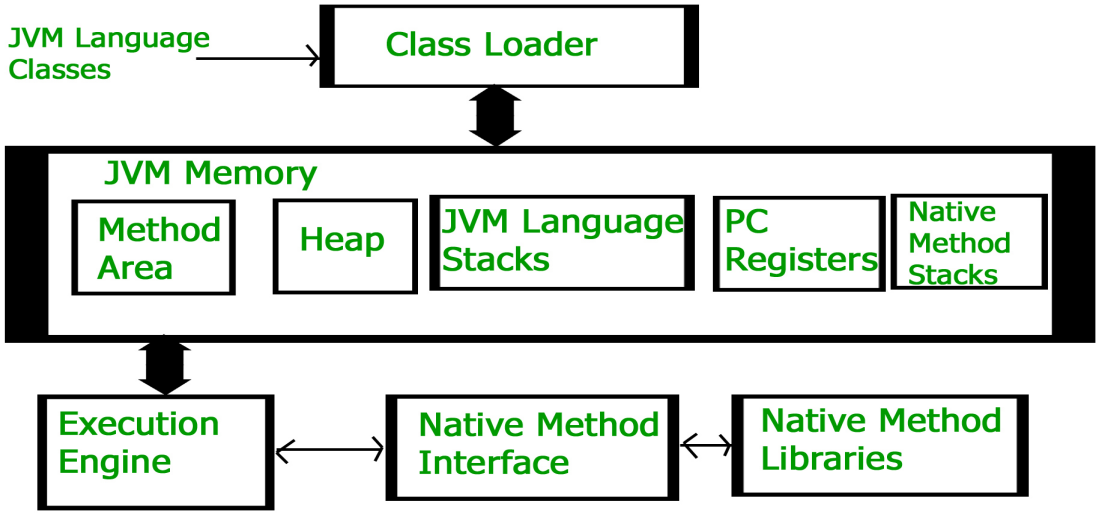
Q 23.Why JVM is called as "virtual machine"?

ANS :

JVM is called a virtual machine because there is no real hardware which interprets the byte code.If you have done any assembly programming for any microprocessor/microcontroller you will able to understand this. A microprocessor has builtin instruction set to interpret the assemly code. Similarly the JVM is similar to a microprocessor in the sense it has its own instruction set but it implemented in software. That is why it is called a virtual machine.

Q24.What are the main components of JVM? Explain them. Or Explain JVM Architecture.

ANS :



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. Bootstrap ClassLoader: This is the first classloader which is the super class of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.
2. Extension ClassLoader: This is the child classloader of Bootstrap and parent classloader of System classloader. It loades the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. System/Application ClassLoader: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to

current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

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 the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1. A virtual processor
2. Interpreter: Read bytecode stream then execute the instructions.
3. Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

Q 25.What is the difference betweenJava compiler (javac) and JIT ?

ANS:

Java compiler compiles source files (.java) to bytecode files (.class). Sun gives a free java compiler which is invoked with the 'javac' command.

A java interpreter is usually referred to as the Java Virtual Machine (or JVM). It reads and executes the bytecodes in the .class files. Sun also supplies a free version of the JVM which is invoked with the 'java' command.

Where we get confusing is when people talk about a Just-In-Time compiler (or JIT compiler). This is actually part of a JVM. Its purpose is to take the generic (i.e. cross-platform) bytecodes and compile them into more machine-specific instructions (allowing the program to run significantly faster). Even though it is referred to as a JIT 'compiler', it is part of the Virtual Machine.

Now, this is what every JVM is facing: should it further translate those byte codes (using that JIT compiler) or should it interpret (or execute) these things itself? JIT compilation is just what it says: just in time compilation. Facing the choice of interpreting those byte codes itself (being slow) or compiling it into real machine code (by activating the JIT compiler) first and then let the real processor do the job, so the real machine code could get executed (or interpreted) by the real CPU is quite a choice to make.

Q 26.Is Empty .java file name avalid source file name?

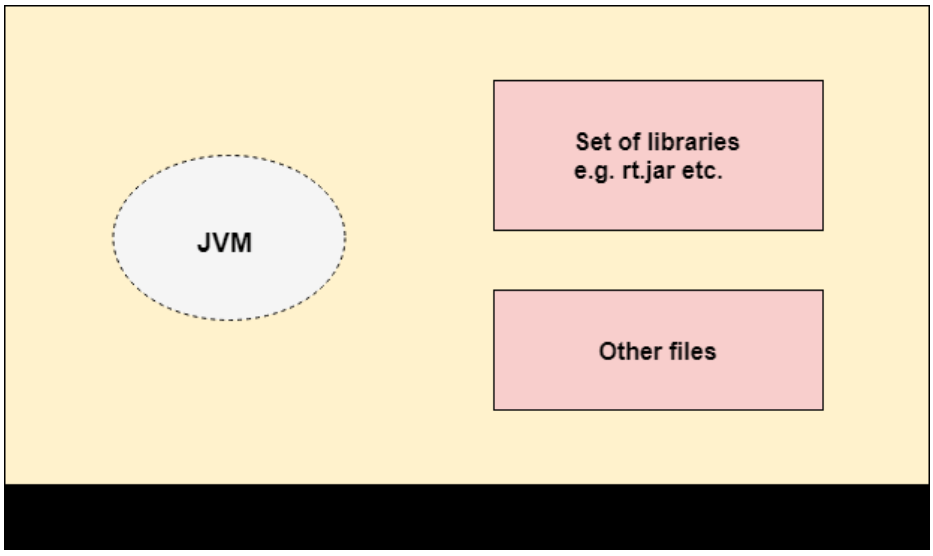
ANS :Yes. An empty *.java* file is a perfectly valid source file.

Q27.Is JRE different for different Platforms ?

ANS :

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 which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



Q 28 .Difference between C++ and java in terms of object creation.

ANS :

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.

Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.

Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

Q 29:Who invokes main() function ?
ANS :
To call some function before main() method in C++,

1. Create a class
2. Create a function in this class to be called.
3. Create the constructor of this class and call the above method in this constructor
4. Now declare an object of this class as a global variable.
5. Global variables are usually declared outside of all of the functions and blocks, at the top of the program. They can be accessed from any portion of the program.

Eg :

```
// C++ program to call some function
// before main() function

#include <iostream>
using namespace std;

// Class
class GFG {

public:
    // Constructor of the class
    GFG()
    {

        // Call the other function
        func();
```

```
    }

    // Function to get executed before main()
    void func()
    {
        cout << "Inside the other function"
             << endl;
    }
};

// Global variable to declare
// the object of class GFG
GFG obj;

// Driver code
int main()
{
    cout << "Inside main method" << endl;
    return 0;
}
```

Output:
Inside the other function
Inside main method

Q 30.What is.class file known as ?
ANS :
A Java class file is a file containing Java bytecode and having .class extension that can be executed by JVM. A Java class file is created by a Java compiler from *.java* files as a result of successful compilation. As we know that a single Java programming language source file (*or we can say .java file*) may contain one class or more than one class. So if a *.java* file has more than one class then each class will compile into a separate class files.

For Example: Save this below code as Test.java on your system.

```
// Compiling this Java program would
// result in multiple class files.

class Sample
{

}

// Class Declaration
class Student
{

}
// Class Declaration
class Test
{
      public static void main(String[] args)
      {
         System.out.println("Class File Structure");
      }
}
```

For Compiling:

```
javac Test.java
```

After compilation there will be 3 class files in corresponding folder named as:

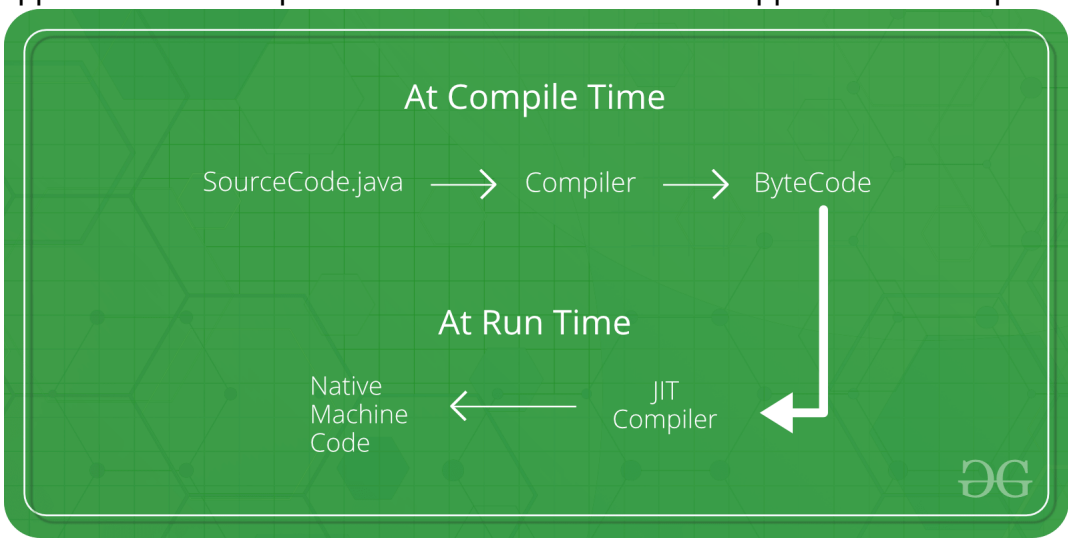
- Sample.class
- Student.class
- Test.class

A single class file structure contains attributes that describe a class file.

Q 31 .Can we define more than one public class in a java source code ? what is the rule of public class and file name . ?
ANS :
No, while defining multiple classes in a single Java file you need to make sure that only one class among them is public. If you have more than one public classes a single file a compile-time error will be generated.

Q 32 .What is JIT compiler?
ANS :
The Just-In-Time (JIT) compiler is a an essential part of the JRE i.e. Java Runtime Environment, that is responsible for performance optimization of java based applications

at run time. Compiler is one of the key aspects in deciding performance of an application for both parties i.e. the end user and the application developer.



WORKING of JIT

Q 33.How many types of memory areas are allocated by JVM?
ANS :
The memory in the JVM divided into 5 different parts:

1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

Let's see about them in brief:

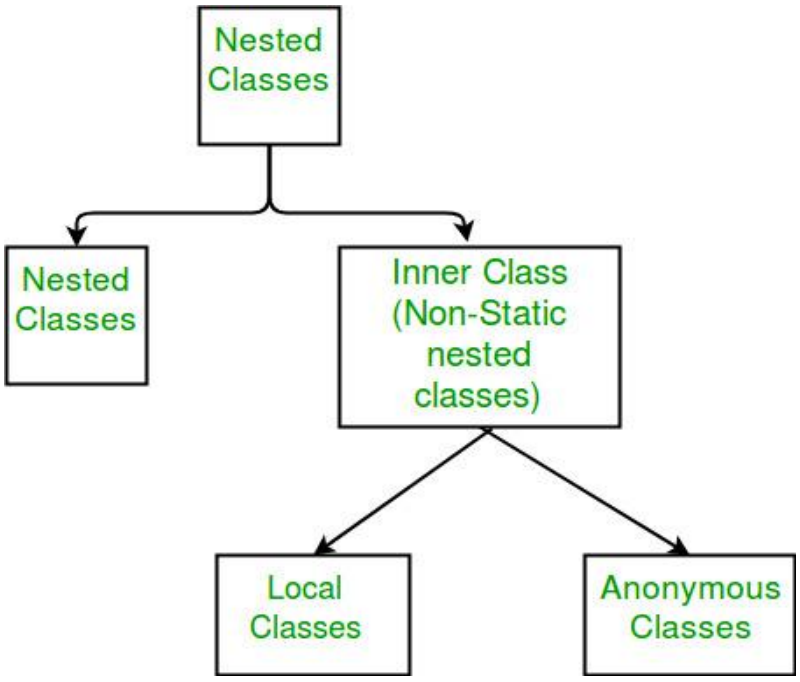
1. Class Loader:It is a subsystem of JVM which is used to load class files.It is mainly responsible for three activities.
    - Loading
    - Linking
    - Initialization
2. Class(Method) Area: It stores class level data of every class such as the runtime constant pool, field and method data, the code for methods.
3. Heap: It is used to allocate memory to objects at run time
4. Stack:
    - Each thread has a private JVM stack, created at the same time as thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.
    - Java Stack stores frames and a new frame is created each time at every invocation of the method.
      A frame is destroyed when its method invocation completes
5. Program Counter Register: Each JVM thread which carries out the task of a specific method has a program counter register associated with it. The

non-native method has a PC which stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

6. Native method Stacks: Also called as C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when its created And it can be of a fixed or dynamic nature.

Q 34. What is the rule for local member in java.
ANS :

1. The scope of local inner class is restricted to the *block* they are defined in.
2. Local inner class cannot be instantiated from outside the *block* where it is created in.
3. Till JDK 7,Local inner class can access only final local variable of the enclosing block. However From JDK 8, it is possible to access the non-final local variable of enclosing block in local inner class.
4. A local class has access to the members of its enclosing class.
5. Local inner classes can extend an abstract class or can also implement an interface.



Declaring a Local Inner class: A local inner class can be declared within a block. This block can be either a method body, initialization block, for loop or even an if statement.

Accessing Members: A local inner class has access to fields of the class enclosing it as well as the fields of the block that it is defined within. These classes, however, can access the variables or parameters of the block that encloses it only if they are declared

as final or are effectively final. A variable whose value is not changed once initialized is called as effectively final variable. A local inner class defined inside a method body, have access to it's parameters.

Q 35 . What are the various access specifiers in Java?
ANS :
There are four types of Java access modifiers:

1. Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Q 36.What is native code?
ANS :
The native keyword is applied to a method to indicate that the method is implemented in native code using JNI (Java Native Interface). native is a modifier applicable only for methods and we can't apply it anywhere else. The methods which are implemented in C, C++ are called as native methods or foreign methods.

The main objectives of native keyword are:

- To improve performance of the system.
- To achieve machine level/memory level communication.
- To use already existing legacy non-java code.

Pseudo code to use native keyword in java:

```
Class Native
{
   Static
   {
      System.LoadLibrary(“Native library path”);
   }
   Public native void m();
}
Class Test
{
   Public static void main(String[] args)
   {
      Native n = new Native();
      n.m();
   }
}
```

Important points about native keyword:

- For native methods implementation is already available in old languages like C, C++ and we are not responsible to provide implementation. Hence native method declaration should end with ; (semi-colon).
- We can’t declare native method as abstract.
- We can’t declare native method as strictfp because there is no guarantee that old languages (C, C++) follow IEEE 754 standard. Hence native strictfp combination is illegal combination for methods.
- The main advantage of native keyword is improvement in performance but the main disadvantage of native keyword is that it breaks platform independent nature of java.

Declaring Native Methods: In this section we explain how to declare a native method in Java and how to generate the corresponding C/C++ function prototype.

Q 38:Why there is no sizeof operator in java ?
ANS :
There is no sizeof operator in Java. We generally don’t need size of objects. Because java data type have already pre-defined size so we does not need of sizeof.

Q 39.What kinds of programs u can develop using Java ?
ANS :
Technology is constantly going through an evolution and so are the languages that are used to develop them. Java is one of the popular programming languages having n number of applications.

- Mobile Applications
- Desktop GUI Applications

- Web-based Applications
- Enterprise Applications
- Scientific Applications
- Gaming Applications
- Big Data technologies
- Business Applications
- Distributed Applications
- Cloud-based Applications

Q 40.U have reference type as a member of class. What is the default value it gets?
ANS :
1. Reference variable is used to point object/values.

2. Classes, interfaces, arrays, enumerations, and, annotations are reference types in Java. Reference variables hold the objects/values of reference types in Java.

3. Reference variable can also store null value. By default, if no object is passed to a reference variable then it will store a null value.

4. You can access object members using a reference variable using dot syntax.

By default value will be NULL.

Q 41.What is the job done by classloader ?
ANS :
Java ClassLoader is an abstract class. It belongs to a java.lang package. It loads classes from different resources. Java ClassLoader is used to load the classes at run time. In other words, JVM performs the linking process at runtime. Classes are loaded into the JVM according to need. If a loaded class depends on another class, that class is loaded as well. When we request to load a class, it delegates the class to its parent. In this way, uniqueness is maintained in the runtime environment. It is essential to execute a Java program.

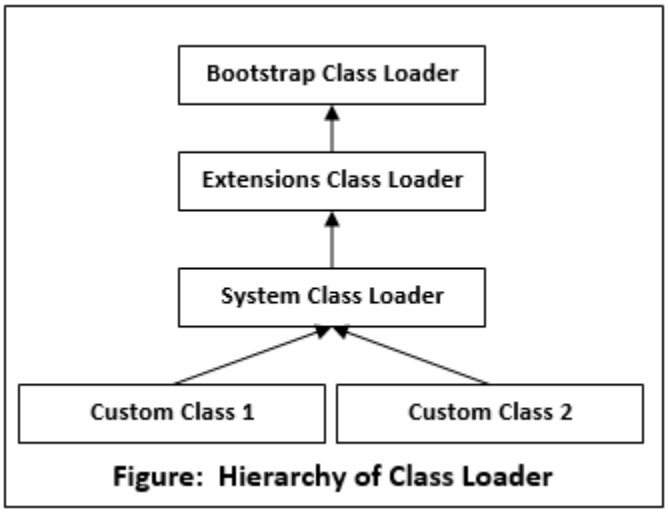


Figure: Hierarchy of Class Loader

Q 42.Explain the hierarchy of classloaders in java.
ANS :
Bootstrap Class Loader: It loads standard JDK class files from rt.jar and other core classes. It is a parent of all class loaders. It doesn't have any parent. When we call String.class.getClassLoader() it returns null, and any code based on it throws NullPointerException. It is also called Primordial ClassLoader. It loads class files from jre/lib/rt.jar. For example, java.lang package class.

Extensions Class Loader: It delegates class loading request to its parent. If the loading of a class is unsuccessful, it loads classes from jre/lib/ext directory or any other directory as java.ext.dirs. It is implemented by sun.misc.Launcher$ExtClassLoader in JVM.

System Class Loader: It loads application specific classes from the CLASSPATH environment variable. It can be set while invoking program using -cp or classpath command line options. It is a child of Extension ClassLoader. It is implemented by sun.misc.Launcher$AppClassLoader class. All Java ClassLoader implements java.lang.ClassLoader.

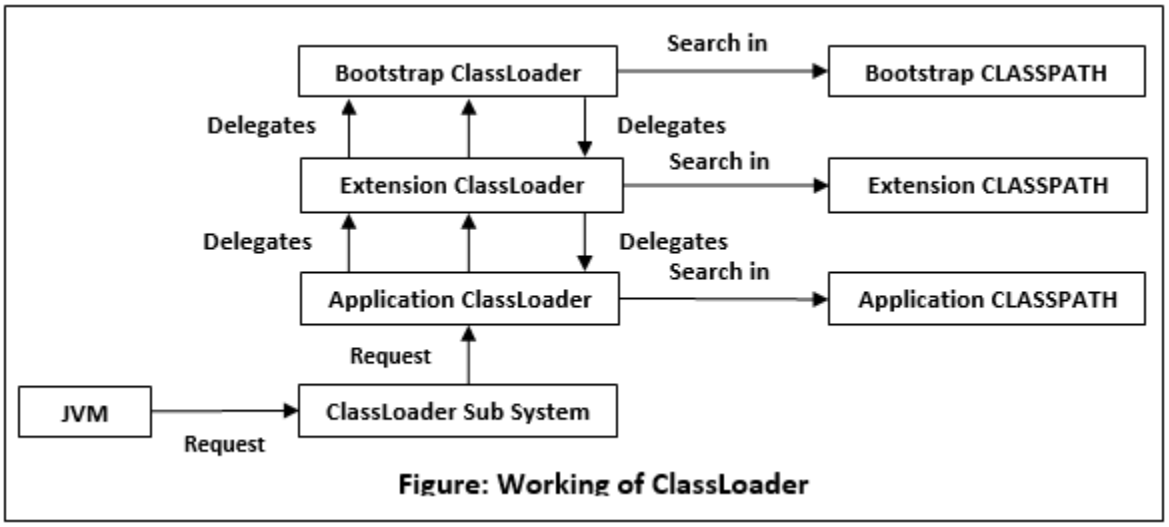


Figure: Working of ClassLoader

Q 43.What is the role played by Bytecode Verifier ?

ANS :

The bytecode verifier traverses the bytecodes, constructs the type state information, and verifies the types of the parameters to all the bytecode instructions.

While all this checking appears excruciatingly detailed, by the time the bytecode verifier has done its work,now the Java interpreter can proceed by knowing that the code will run securely. Knowing these properties makes the Java interpreter much faster, because it doesn't have to check anything. There are no operand type checks and no

stack overflow checks. The interpreter can thus function at full speed without compromising reliability.

Q 46.When parseInt() method can be used?

ANS :

While operating upon strings, there are times when we need to convert a number represented as a string into an integer type. The method generally used to convert String to Integer in Java is parseInt().

How to use parseInt() method in Java?

There are two variants of this method:

public static int parseInt(String s) throws NumberFormatException

- This function parses the string argument as a signed

  decimal integer.

public static int parseInt(String s, int radix) throws NumberFormatException

- This function parses the string argument as a signed

  integer in the radix specified by the second argument.

In simple words, both the methods convert the string into its integer equivalent. The only difference being is that of the parameter radix. The first method can be considered as an equivalent of the second method with radix = 10 (Decimal).

- parseInt(String s) − This returns an integer (decimal only).
- parseInt(int i) − This returns an integer, given a string representation of decimal, binary, octal, or hexadecimal (radix equals 10, 2, 8, or 16 respectively) numbers as input.

Q 47.What is finalized() method ?

ANS:

The java.lang.Object.finalize() is called by the garbage collector on an object when garbage collection determines that there are no more references to the object. A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.

Declaration :Following is the declaration for java.lang.Object.finalize() method

protected void finalize()

Parameters : NA

Return Value : This method does not return a value.

Exception : Throwable − the Exception raised by this method

Q 50.What are the expressions allowed in switch block of java ?

ANS :

The switch statement is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Basically, the expression can be byte, short, char, and int primitive data types. Beginning with JDK7, it also works with enumerated types (Enums in java), the String class and Wrapper classes.

Syntax of Switch-case :

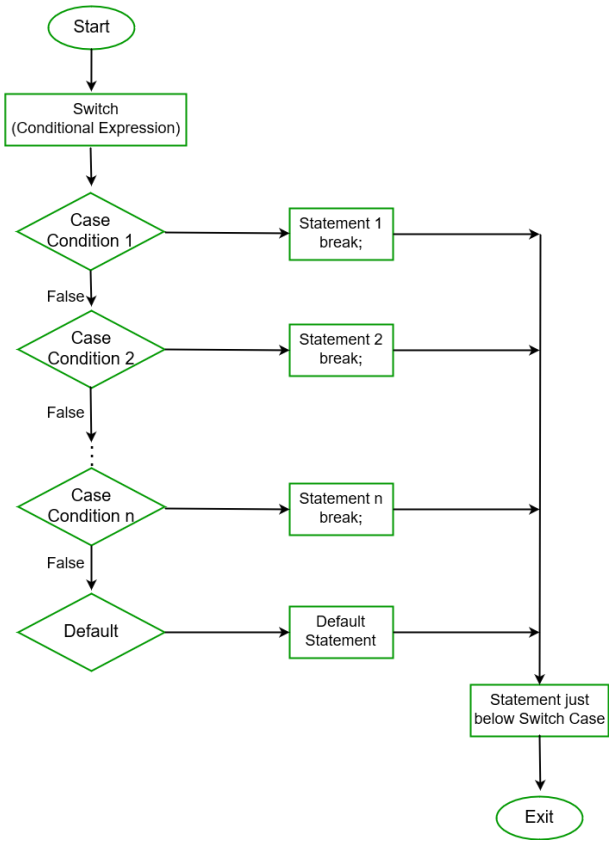
```
// switch statement
switch(expression)
{
   // case statements
   // values must be of same type of expression
   case value1 :
      // Statements
      break; // break is optional

   case value2 :
      // Statements
      break; // break is optional

   // We can have any number of case statements
```

```
    // below is default statement, used when none of the cases is true.
    // No break is needed in the default case.
    default :
      // Statements
}
```

Flow Diagram of Switch-case :



Some Important rules for switch statements :

- Duplicate case values are not allowed.
- The value for a case must be of the same data type as the variable in the switch.
- The value for a case must be a constant or a literal. Variables are not allowed.

- The break statement is used inside the switch to terminate a statement sequence.
- The break statement is optional. If omitted, execution will continue on into the next case.
- The default statement is optional and can appear anywhere inside the switch block. In case, if it is not at the end, then a break statement must be kept after the default statement to omit the execution of the next case statement.