**What is the most important feature of Java?**

Java is a platform independent language.

**What do you mean by platform independence?**

Platform independence means that we can write and compile the java code in one platform (eg Windows) and can execute the class in any other supported platform eg (Linux,Solaris,etc).

**Why java is platform independent?**

The most unique feature of java is platform independent. In any programming language source code is compiled in to executable code. This cannot be run across all platforms. When javac compiles a java program it generates an executable file called .class file. class file contains byte codes. Byte codes are interpreted only by JVMs. Since these JVM’s are made available across all platforms, we can execute this byte code in any platform. Byte code generated in windows environment can also be executed in Linux environment. This makes java platform independent.

**What is a JVM?**

JVM is Java Virtual Machine which is a run time environment for the compiled java class files. JVM executes the class files produced by compilers.

**Are JVM's platform independent?**

JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.

**What is the difference between a JDK and a JVM?**

JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

**What is a pointer and does Java support pointers?**

Pointer is a reference handle to a memory location. Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.

**What is jar?**

Jar stands for java archive file. Jars are created by using Jar.exe tool. Jar files contains .class files, other resources used in our application and manifest file.Manifest file contains class name with main method.jar contains compressed .class files. Jvm finds these .class files without uncompressing this jar.

**What are the different kinds of memory in java?**

here are two kinds of memory used in Java. These are called stack memory and heap memory. Stack memory stores primitive types and the addresses of objects. The object values are stored in heap memory. An object reference on the stack is only an address that refers to the place in heap memory where that object is kept.

Test test1 = new Test();

Test test2 = new Test();

test2 = test1;

What you're actually doing when you write this is assigning the address of the test1 object to the test2 object. Assume that test1's memory address was 0x33d444 and that test2's address was 0x99f775. After performing the above assignment, test2 now holds this address in stack memory:0x99f775, which refers to the same object as test1. The test2 object on the heap still exists, but it cannot be accessed. That's because this reassignment overwrote the old address that test2 was keeping on the stack. This kind of reassignment makes two stack references to the same object on the heap.

It is useful to know that these two different kinds of memory exist in Java. Stack memory is the program's memory, and heap memory resides outside of the program.

As a Java programmer, you do not have to directly address memory allocation and recovery of memory space, which is a common headache for C++ programmers. When you need a new object, Java allocates the required memory. When you are done with an object, the memory is reclaimed for you automatically via Java's garbage collection facility.

Garbage collection runs as a thread in the background, looking for objects that no longer have a usable reference. When it finds them, it destroys them and reclaims the memory.