JAVA - A HIGH LEVEL OVERVIEW

What is Java(core attributes):

1. General purpose: can be used to develop wide variety of applications
2. OO: helps to model real world scenarios
3. Platform independent: Write once and Run anywhere
4. Concurrent: multi threading
5. Very fast
6. Easy to use: Some features of c/c++ that made them complex have not been picked up by java like memory management.
7. Secure

How Java came into picture:

1. In 1991 Sun microsystems was acquired by oracle.
2. A specific team called green team, was focussed on networking between heterogeneous devices. This was under green project
3. So communication between TV and VCR, all will have the same software installed and they should be able to communicate
4. Goals for this included: consume less memory, platform independence, security, multi threading
5. C++ was initially considered but it did not suffice for requirements considering platform independence
6. WWW also targetted the same thing i.e.communication between devices.
7. For this Java applets were created

Programming language types:

1. High level language
    a. User can easily interpret and read it
2. Low level language
    a. Provide little or no abstraction towards machine instructions
    b. Allow to manipulate h/w elements like memory, register etc
    c. *Assembly languages specific to various CPU architectures, such as x86 Assembly and ARM Assembly,*
    d. *platform-specific and less portable.*

Compilation:

1. The compiler is platform dependent
2. It converts the source code to machine code, and machine code is then consumed by CPU to execute.
3. This is done using fetch-and-execute lifecycle where the machine code is put into memory and fetched by CPU instruction by instruction.
4. This provides faster execution but problem is platform dependency and every compiler converts into machine code which can be read by specific machine only

Interpretation:

1. The interpreter is platform independent

2. It converts the source code to interpreted code which is then converted to machine code
3. This provides slower execution as it executed instruction by instruction(also solving all issues) but is platform independent

Java virtual machine:
1. JVM is the interpreter here. The compiler converts source code into java byte code. Which can then be interpreted by JVM and converted to machine code.
2. Java is an interpreted language.
3. Called virtual machine because it is an abstract computing machine. Like CPU it also consumes instructions from bytecode and executed them. It also manipulates memory.
4. There are multiple compilers created for other languages like groovy, which convert source code to Java bytecode, which can then be executed using jvm. This helps to attain platform independency in those languages.
5. It has following responsibilities:
    a. Automatic memory management
    b. Loading and interpreting java byte code
    c. Security
6. JVM high level Architecture:
    a. Whenever a java program is run, a jvm instances is created and both instance and bytecode are loaded into memory. One instance one java program execution.
    b. The JVM gets some memory allocated by the OS.
    c. JVM loads the bytecode using Class loader, then verifies for corruption using bytecode verifier
    d. This is then executed by execution engine
    e. Garbage collector is used for memory management
    f. Security is ensured by security manager.

JIT Compiler
1. Bytecode is specifically designed for JVM which leads for fdaster execution
2. JIT compiler identifies frequently executed bytecode- hot spots which is converted to machine code and is cached
3. This then can be used from cache to execute whenever next time it is required.
4. Called dynamic compilation.
5. This makes java as fast as c/c++

Platforms of Java:
1. Java SE
    a. Standalone applications for desktops and servers
2. Java EE
    a. Large scale applications for servers
    b. Build on java SE
3. Java micro editions J ME
    a. For resource constrained devices
    b. Uses subset of java SE