**Platform independency of java**

**Fig 1: For java language**

**Fig 2: For languages like c, c++**

Java language is called “machine independent because of the following reasons:

1. For java language, java complier (javac, a part of Java Development Kit (JDK)) is same for all machine or OS (windows OS, Linus dependent OS, android) and instead of generating machine instruction (i.e. steps to performed by the machine to execute the program) it generates an intermediate code named Byte code (The intermediate code named byte code comes with the extension .class. It is machine independent and does not depend upon the machine computer architecture). Then the roleplaying of Java Virtual Machine (JVM) (Note that, JVM is also within JDK. Basically everything relates with java is inside JDK: the java compiler, java debugger, other development tools and JRE (JVM)) comes into scenario. The java interpreter which is inside Java Virtual Machine transforms the intermediate “byte code” into machined code. Now, like other Virtual Machines, Java Virtual Machine is also a software application that simulates a computer, but hides the underlying Operating System and computer architecture. (Virtual machine itself is h/o dependent as any other software (s/o) application. But provides the processes which are running within it a machine independent environment environment). So, it also offers the programmers a platform independent environment for writing, compiling and executing the java code. That is why, java is called platform independent.
2. Another reason is that, for java language, the memory sizes of different variables do not depend upon java compiler as well as machine computer architecture. Those are machine independent.

(In languages like c, c++ variable sizes depend upon compile i.e. machine computer architecture)

Note: - We sometimes mention Hardware as h/o and software as s/o. (At least, in computer engineering level)