**Why Java take 2 byte of memory for store character ?**

Java support more than 18 international languages so java take 2 byte for characters, because for 18 international language 1 byte of memory is not sufficient for storing all characters and symbols present in 18 languages. Java supports Unicode but c support ascii code. In ascii code only English language are present, so for storing all English latter and symbols 1 byte is sufficient.

### What is java and javac ?

Java and javac are tools or application programs or exe files developed by sun micro system and supply as a part of jdk 1.5/1.6/1.7/1.8 in bin folder. Java tool are used for rur the java program and javac tool are used for compile the java program.

## Why Using naming Conversion

Different Java programmers can have different styles and approaches to write program. By using standard Java naming conventions they make their code easier to read for themselves and for other programmers. Readability of Java code is important because it means less time is spent trying to figure out what the code does, and leaving more time to fix or modify it.

1. Every package name should exist a lower case latter.

## Example

package student; // creating package

import java.lang; // import package

2. First letter of every word of class name or interface name should exists in upper case.

## Example

class StudentDetails

{

.....

.....

}

interface FacultyDetail

{

.....

.....

}

3. Every constant value should exists in upper case latter. It is containing more than one word than it should be separated with underscore (-).

## Example

class Student

{

final String COLLEGE\_NAME="abcd";

....

....

}

**Note:** if any variable is preceded by final keyword is known as constant value.

## Example

class Student

{

Final String Student\_name="abcd";

}

While declaring variable name, method, object reference the first letter of first word should be exits in lower case but from the second words onward the first letter should exists in upper case.

## Example

class Student

{

String StudentName="xyz";

void instantStudentDetails();

{

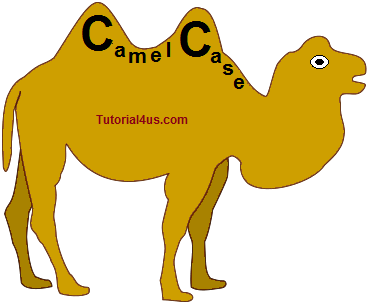
....

....

}

Student final

## CamelCase in java naming conventions



Java follows camelcase syntax for naming the class, interface, method and variable.   
According to CamelCase if name is combined with two words, second word will start with uppercase letter always.

General Example studentName, customerAccount. In term of java programming e.g. actionPerformed(), firstName, ActionEvent, ActionListener etc.

**StringBuider vs StringBuffer**  
  
1) StringBuilder is non synchronized version of StringBuffer class. Methods in StringBuilder e.g. all overloaded version of append() method is not synchronized.  
  
2) StringBuilder is definitely faster than StringBuffer because of no overhead of acquiring and releasing locks associated with synchronized methods.  
  
3) StringBuffer is thread-safe and StringBuilder is not. we can not share Instances of StringBuilder class between multiple threads. If such synchronization is required then it is better to use StringBuffer class.  
  
4) StringBuffer is old class, its there in JDK from very first release, while StringBuilder is relatively newer class, introduced much later in release of JDK 1.5  
  
5) Another interesting fact to know about both of this class is that, when we do String concatenation using + operator, Java internally convert that call to corresponding StringBuilder append() method class. For example "one" + "two" + "three" will be converted to new StringBuilder().append("one").append("two").append("three"). Only problem is that it initialize StringBuilder with default capacity, which means expensive array copy operation, when StringBuilder get resized.

### Difference between PATH and CLASSPATH in Java

1)Main difference between PATH and CLASSPATH is that  PATH is an environment variable which is used to locate JDK binaries like "java" or "javac" command used to run java program and compile java source file. On the other hand CLASSPATH environment variable is used by System or Application ClassLoader to locate and load compile Java bytecodes stored in .class file.  
  
2) In order to set PATH in Java you need to include JDK\_HOME/bin directory in PATH environment variable while in order to set CLASSPATH in Java you need to include all those directory where you have put either your .class file or JAR file which is required by your Java application.  
  
3) Another significant difference between PATH and CLASSPATH is that PATH can not be overridden by any Java settings but CLASSPATH can be overridden by providing command line option -classpath or -cp to both "java" and "javac" commands or by using Class-Path attribute in Manifest file inside JAR archive.  
  
4) PATH environment variable is used by operating system to find any binary or command typed in shell, this is true for both Windows and Linux environment while CLASSPATH is only used by Java ClassLoaders to load class files.

## Difference between Serializable and Externalizable in Java

1) One of the obvious difference between Serializable and Externalizable is that Serializable is a [marker interface](http://javarevisited.blogspot.in/2012/01/what-is-marker-interfaces-in-java-and.html) i.e. does not contain any method but Externalizable interface contains two methods writeExternal() and readExternal().

# [Difference between FileInputStream and FileReader in Java | InputStream vs Reader](http://javarevisited.blogspot.sg/2014/04/difference-between-fileinputstream-and-filereader-in-java.html)

[InputStream](http://javarevisited.blogspot.sg/2012/08/convert-inputstream-to-string-java-example-tutorial.html) is used to read binary data, while Reader is used to read text data,

### What is volatile variable in Java - When to use

🡪volatile variable in Java is a special variable which is used to signal threads, compiler that this particular variables values is going to be updated by multiple thread inside Java application. By making a variable volatile using volatile keyword in Java,

**Important point related to volatile keyword in Java**  
   
1) Volatile keyword can only be applied to variable, it can not be applied to class or method. using volatile keyword along with class and method is compiler error.  
  
2) volatile is also refereed as modifier in Java.

**When to use Volatile variable in Java**  
  
1) Any variable which is shared between multiple threads should be made variable, in order to ensure that all thread must see latest value of volatile variable.  
  
2) A signal to compiler and JIT to ensure that compiler does not change ordering or volatile variable and moves them out of synchronized context.  
  
3) You want to save cost of synchronization as volatile variables are less expensive than synchronization.