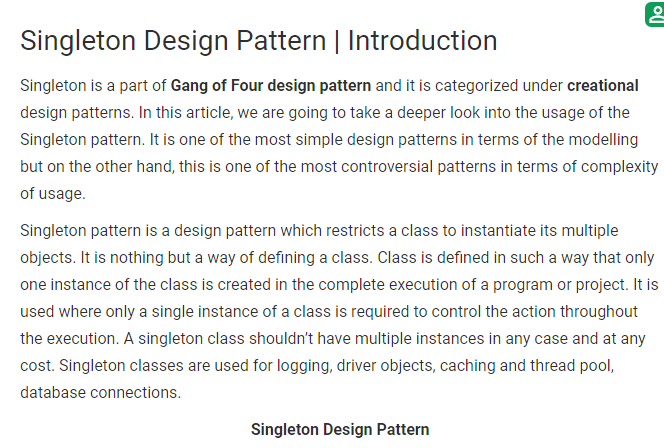
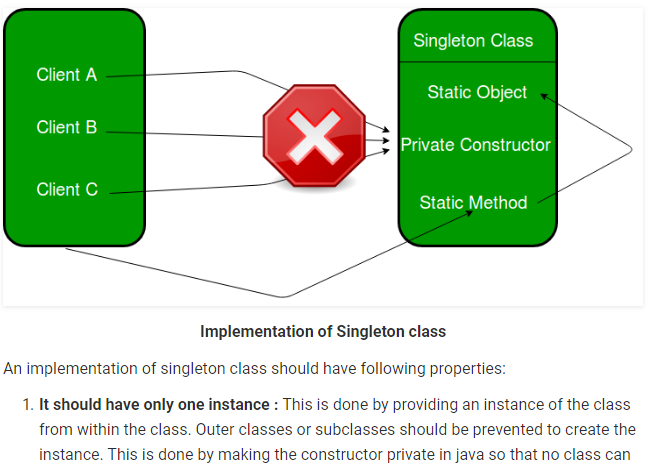
Java interview questions



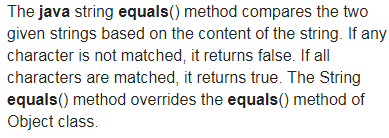


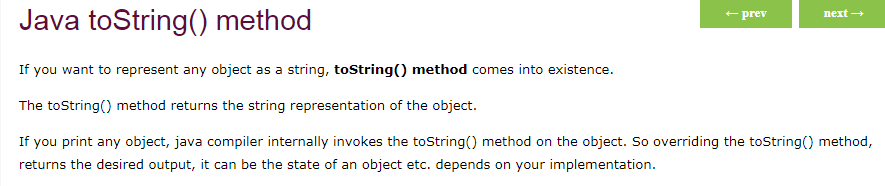
1. Why the main method is static in java?

This is neccesary because **main**() is called by the JVM before any objects are made. Since it is **static** it can be directly invoked via the class. Similarly, we use **static**sometime for user defined **methods** so that we need not to make objects.

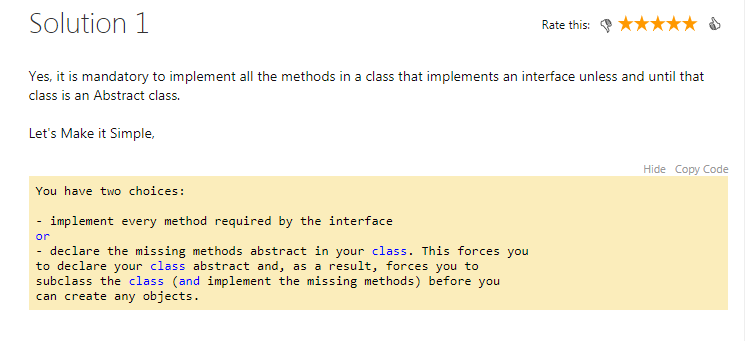
1. **Comparable vs Comparator**. **Comparable** interface can be used to provide single way of sorting whereas**Comparator** interface is used to provide different ways of sorting. For using **Comparable**, Class needs to implement it whereas for using **Comparator** we don't need to make any change in the class.

When we make a collection element comparable(by having it implement Comparable), we get only one chance to implement the compareTo() method. The solution is using [Comparator.](https://www.geeksforgeeks.org/comparator-interface-java/)





[is compulsory to implement all the methods in interface?](https://www.codeproject.com/Questions/195703/is-compulsory-to-implement-all-the-methods-in-inte)



Path vs classpath

path is a mediator between developer and operating system to inform binary file path where as Classpath is a mediator between developer and compiler to inform the library file path those are used in our source code

The path points to the location of the jre i.e. the java binary files such as the jvm and necessary libraries. The classpath points to the classes you developed so that the jvm can find them and load them when you run your product.

So essentially you need the path to find java so it can then find your classes and run them from the classpath

1

path : it is location of bin files(binary executable files) example- java.exe,javac.exe

classPath: it is location of your .class file(which is created after compile your java source file)

Jar vs runnable jar

The runnable jar contains a MANIFEST.MF file, which defines the Main class to be executed when the jar is run.

Non-runnable jars are just libraries of classes, that can be added to the classpath so that code is reused (it also contains the manifest file, but no main class there)

* Bootstrap Class Loader  
  Bootstrap class loader loads java’s core classes like java.lang, java.util etc. These are classes that are part of java runtime environment. Bootstrap class loader is native implementation and so they may differ across different JVMs.
* Extensions Class Loader  
  JAVA\_HOME/jre/lib/ext contains jar packages that are extensions of standard core java classes. Extensions class loader loads classes from this ext folder. Using the system environment propery java.ext.dirs you can add ‘ext’ folders and jar files to be loaded using extensions class loader.
* System Class Loader  
  Java classes that are available in the java classpath are loaded using System class loader.