## 10 Singleton Pattern Interview Questions in Java - Answered

Singleton design pattern is one of the most common patterns you will see in Java applications and it's also used heavily in core Java libraries. Questions from Singleton pattern is very common in Java interviews and good knowledge of how to implement Singleton pattern certainly help.This is also one of my favorite design pattern interview question and has lots of interesting follow-up to dig into details , this not only check the knowledge of design pattern but also check coding, multithreading aspect which is very important while working for a real life application. In this post have listed some of the most common question asked on Singleton pattern during a Java Interview. I have not provided the answers of these questions as they are easily available via google search but if you guys need I can try to modify this tutorial to include answers as well. As promised earlier and having received lot of request for providing answers of these question, I have decided to update this post along with answers. By the way if you are preparing for interview on Java technology than you can check my collection on Java interview questionsand multi-threading interview questions. There are lot of resources in Javarevisited which can help you in your interview preparation. On the other hand if you are more interested on design pattern tutorials than you can check my post on builder design pattern

# 10 Interview question on Singleton Pattern in Java

Here is my collection of interview questions based upon Singleton design pattern. They are collected from various Java interviews and highlights key aspects of pattern and where it is broken, if you know how to create thread-safe singletons and different ways to implement this pattern, and pros and cons of each approach. Questions starts with :

## What is Singleton class? Have you used Singleton before?

Singleton is a class which has only one instance in whole application and provides a`getInstance()` method to access the singleton instance. There are many classes in JDK which is implemented using Singleton pattern like `java.lang.Runtime` which provides`getRuntime()` method to get access of it and used to get free memory and total memory in Java.

## Which classes are candidates of Singleton? Which kind of class do you make Singleton in Java?

Here they check whether candidate has enough experience on usage of singleton or not. Does he is familiar of advantage/disadvantage or alternatives available for singleton in Java or not.

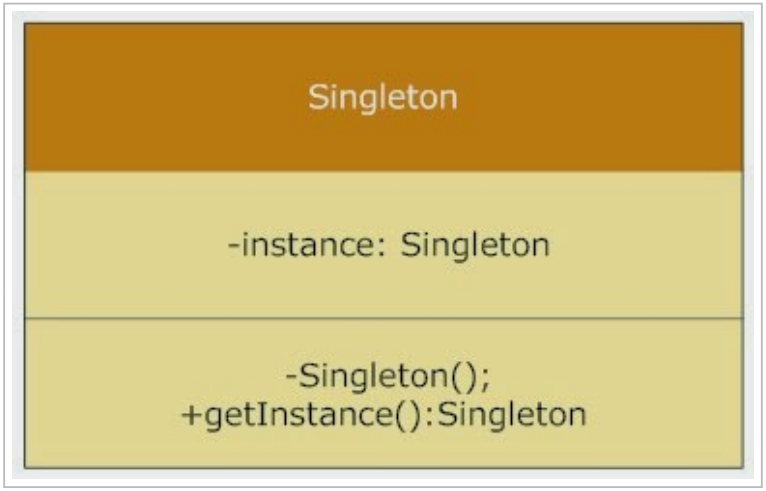
Answer : Any class which you want to be available to whole application and whole only one instance is viable is candidate of becoming Singleton. One example of this is `Runtime` class , since on whole java application only one runtime environment can be possible making`Runtime` Singleton is right decision. Another example is a utility classes like Popup in GUI application, if you want to show popup with message you can have one PopUp class on whole GUI application and anytime just get its instance, and call `show()` with message.

## Can you write code for getInstance() method of a Singleton class in Java?

Most of the java programmer fail here if they have mugged up the singleton code because

you can ask lots of follow-up question based upon the code they have written. I have seen many programmer write Singleton `getInstance()` method with double checked locking but they are not really familiar with the caveat associated with double checking of singleton prior to Java 5.

Answer : Until asked don't write code using double checked locking as it is more complex and chances of errors are more but if you have deep knowledge of double checked locking,volatile variable and lazy loading than this is your chance to shine. I have shared code examples of writing singleton classes using enum, using static factory and with double checked locking in my recent post Why Enum Singletons are better in Java, please see there.




## Is it better to make whole getInstance() method synchronized or just critical section is enough? Which one you will prefer?

This is really nice question and I mostly asked to just quickly check whether candidate is aware of performance trade off of unnecessary locking or not. Since locking only make sense when we need to create instance and rest of the time its just read only access so locking of critical section is always better option. read more about synchronization on How Synchronization works in Java

Answer : This is again related to double checked locking pattern, well synchronization is costly and when you apply this on whole method than call to `getInstance()` will be synchronized and contented. Since synchronization is only needed during initialization on singleton instance, to prevent creating another instance of Singleton, It's better to only synchronize critical section and not whole method. Singleton pattern is also closely related to factory design pattern where `getInstance()` serves as static factory method.

## What is lazy and early loading of Singleton and how will you implement it?

This is another great Singleton interview question in terms of understanding of concept of loading and cost associated with class loading in Java. Many of which I have interviewed not really familiar with this but its good to know concept.

Answer : As there are many ways to implement Singleton like using double checked locking or Singleton class with static final instance initialized during class loading. Former is called lazy loading because Singleton instance is created only when client calls `getInstance()` method while later is called early loading because Singleton instance is created when class is loaded into memory.

## Give me some examples of Singleton pattern from Java Development Kit?

This is open question to all, please share which classes are Singleton in JDK. Answer to this question is `java.lang.Runtime`

Answer : There are many classes in Java Development Kit which is written using singletonpattern, here are few of them:

1. Java.lang.Runtime with getRuntime() method
2. Java.awt.Toolkit with getDefaultToolkit()

3. Java.awt.Desktop with getDesktop()

## What is double checked locking in Singleton?

One of the most hyped question on Singleton pattern and really demands complete understanding to get it right because of Java Memory model caveat prior to Java 5. If a guy comes up with a solution of using volatile keyword with Singleton instance and explains it then it really shows it has in depth knowledge of Java memory model and he is constantly updating his Java knowledge.

Answer : Double checked locking is a technique to prevent creating another instance of Singleton when call to `getInstance()` method is made in multi-threading environment. In Double checked locking pattern as shown in below example, singleton instance is checked two times before initialization. See here to learn more about double-checked-locking in Java.

```
public static Singleton getInstance(){
     if(_INSTANCE == null){
         synchronized(Singleton.class){
         //double checked locking - because second check of Singleton insta
nce with lock
                 if(_INSTANCE == null){
                     _INSTANCE = new Singleton();
                 }
            }
         }
     return _INSTANCE;
}
```

Double checked locking should only be used when you have requirement for lazyinitialization otherwise use Enum to implement singleton or simple static final variable.

## How do you prevent for creating another instance of Singleton using clone() method?

This type of questions generally comes some time by asking how to break singleton or when Singleton is not Singleton in Java.

Answer : Preferred way is not to implement Cloneable interface as why should one wants to create `clone()` of Singleton and if you do just throw Exception from `clone()` method as “Can not create clone of Singleton class”.

## How do you prevent for creating another instance of Singleton using reflection?

Open to all. In my opinion throwing exception from constructor is an option.
Answer: This is similar to previous interview question. Since constructor of Singleton class is supposed to be private it prevents creating instance of Singleton from outside butReflection can access private fields and methods, which opens a threat of another instance. This can be avoided by throwing Exception from constructor as “Singleton already initialized”

## How do you prevent for creating another instance of Singleton during serialization?

Another great question which requires knowledge of Serialization in Java and how to use it for persisting Singleton classes. This is open to you all but in my opinion use of readResolve() method can sort this out for you.
Answer: You can prevent this by using `readResolve()` method, since during serialization`readObject()` is used to create instance and it return new instance every time but by using readResolve you can replace it with original Singleton instance. I have shared code on how to do it in my post Enum as Singleton in Java. This is also one of the reason I have said that use Enum to create Singleton because serialization of enum is taken

care by JVM and it provides guaranteed of that.

## When is Singleton not a Singleton in Java?

There is a very good article present in Sun's Java site which discusses various scenarios when a Singleton is not really remains Singleton and multiple instance of Singleton is possible. Here is the link of that articlehttp://java.sun.com/developer/technicalArticles/Programming/singletons/

Apart from these questions on Singleton pattern, some of my reader contribute few more questions, which I included here. Thank you guys for your contribution.

## Why you should avoid the singleton anti-pattern at all and replace it with DI?

Answer : Singleton Dependency Injection: every class that needs access to a singleton gets the object through its constructors or with a DI-container.

## Why Singleton is Anti pattern

With more and more classes calling `getInstance()` the code gets more and more tightly coupled, monolithic, not testable and hard to change and hard to reuse because of not configurable, hidden dependencies. Also, there would be no need for this clumsy double checked locking if you call `getInstance` less often (i.e. once).

## How many ways you can write Singleton Class in Java?

Answer : I know at least four ways to implement Singleton pattern in Java

1. Singleton by synchronizing `getInstance()` method
2. Singleton with public static final field initialized during class loading.

3. Singleton generated by static nested class, also referred as Singleton holder pattern.
4. From Java 5 on-wards using Enums

## How to write thread-safe Singleton in Java?

Answer : Thread safe Singleton usually refers to write thread safe code which creates one and only one instance of Singleton if called by multiple thread at same time. There are many ways to achieve this like by using double checked locking technique as shown above and by using Enum or Singleton initialized by class loader.

At last few more questions for your practice, contributed by Mansi, Thank you Mansi

14) Singleton vs Static Class?
15) When to choose Singleton over Static Class?
16) Can you replace Singleton with Static Class in Java?
17) Difference between Singleton and Static Class in java?
18) Advantage of Singleton over Static Class?

I have covered answers of couple of these questions in my post, Singleton vs Static Class in Java - Pros and Cons. If you like to read more Java interview questions you can have a look on some of my favorites below