More basic Java interview questions - frequently asked Java interview

Explain in brief transient modifiers and volatile modifiers.

When serializable interface is declared, the compiler knows that the object has to be handled so as so be able to serialize it. However, if you declare a variable in an object as transient, then it doesn't get serialized.

Volatile - Specifying a variable as volatile tells the JVM that any threads using that variable are not allowed to cache that value at all.

Volatile modifier tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of the program.

Define daemon threads and explain its purposes.

Threads that work in the background to support the runtime environment are called daemon threads.
Eg garbage collector threads.

When the only remaining threads in a process are daemon threads, the interpreter exits. This makes sense because when only daemon threads remain, there is no other thread for which a daemon thread can provide a service.

You cannot create a daemon method but you can use
public final void setDaemon(boolean isDaemon) method to turn it into one.

Can you explain JAVAdoc utility.

Javadoc utility enables you to keep the code and the documentation in sync easily.

The javadoc utility lets you put your comments right next to your code, inside your ".java" source files.

All you need to do after completing your code is to run the Javadoc utility to create your HTML documentation automatically.

StringBuilder class vs. StringBuffer class.

StringBuilder is unsynchronized whereas StringBuffer is synchronized. So when the application needs to be run only in a single thread then it is better to use StringBuilder.

StringBuilder is more efficient than StringBuffer.

Can you explain semaphore and monitors in java threading?

A semaphore is a flag variable used to check whether a resource is currently being used by another thread or process.

The drawback of semaphores is that there is no control or guarantee of proper usage.

A Monitor defines a lock and condition variables for managing concurrent access to shared data. The monitor uses the lock to ensure that only a single thread is active in the monitor code at any time.

A semaphore is a generalization of a monitor. A monitor allows only one thread to lock an object at once.

Describe synchronization in respect to multithreading.

Multithreading occurs asynchronously, meaning one thread executes independently of the other threads. In this way, threads don't depend on each other's execution. In contrast, processes that run synchronously depend on each other. That is, one process waits until the other process terminates before it can execute

What are Checked and UnChecked Exception?

The java.lang.Throwable class has two subclasses Error and Exception. There are two types of exceptions non runtime exceptions and runtime exceptions. Non runtime exceptions are called checked exceptions and the unchecked exceptions are runtime exceptions.

Runtime Exceptions occur when the code is not robust and non runtime exceptions occur due to the problems is environment, settings, etc.

In which scenario LazyInitilizationException will occur in Hibernate ? how to over come it?

when we are loading the data from the data base use load() method of Session.
And we are getting the data from bean after session has been closed . In such scenerio LazyInitilizationException will occur. i.e for Example

```
Session sess=factory.openSession();
Transaction trans=sess.beginTransaction();
SampleEntity se=(SampleEntity)sess.load(SampleEntity.class,11);
trans.commit(); sess,close();
System.out.println(se.getName());
```

To overcome this problem use initialize() of Hibernate i.e

```
Session sess=factory.openSession();
Transaction trans=sess.beginTransaction();
SampleEntity se=(SampleEntity)sess.load(SampleEntity.class,11);
```

Hibernate.initialize(se);
trans.commit(); sess,close();
System.out.println(se.getName());

Is it recommended to handle NumberFormatException?

NumberFormatException is Unchecked Exception. It is not recommended to handle Unchecked exception. So, we have to prevent it with out raising that Exception. we can prevent it by using regular expression i.e

```
public boolean isInteger(String str)
{
return str.matches("^-?[0-9]+(\\/[0-9]+)?$");
}
```

What is Singleton pattern? How can we implements this pattern in Java and Spring?

Singleton pattern allows us to create only one instantiation of a class to one object.
We can achieve this in Java by using private class Constructor i.e

```
Public Class Singleton{
private static Singleton sing;
private Singleton(){}
public static Singleton getObject()
{
sign=new Singleton();
}
}
```

In spring by default all beans are Singleton . To change we have to define a tag called scope for latest version and for old version we have to use tag called singleton
New Version:-

```
<bean id='sampleid" class="example.sample" scope="singleton">
```

Old Version:-

```
<bean id='sampleid" class="example.sample" singleton="true">
```