JAVA Interview Questions:

1:What is a ClassLoader?

Java Classloader is the program that belongs to JRE (Java Runtime Environment). The task of ClassLoader is to load the required classes and interfaces to the JVM when required.

Example- To get input from the console, we require the scanner class. And the Scanner class is loaded by the ClassLoader

2: How would you differentiate between a String, StringBuffer, and a StringBuilder?

Storage area: In string, the String pool serves as the storage area. For StringBuilder and StringBuffer, heap memory is the storage area.

Mutability: A String is immutable, whereas both the StringBuilder and StringBuffer are mutable.

Efficiency: It is quite slow to work with a String. However, StringBuilder is the fastest in performing operations. The speed of a StringBuffer is more than a String and less than a StringBuilder. (For example appending a character is fastest in StringBuilder and very slow in String because a new memory is required for the new String with appended character.)

Thread-safe: In the case of a threaded environment, StringBuilder and StringBuffer are used whereas a String is not used. However, StringBuilder is suitable for an environment with a single thread, and a StringBuffer is suitable for multiple threads.

3: Using relevant properties highlight the differences between interfaces and abstract classes.

Availability of methods: Only abstract methods are available in interfaces, whereas non-abstract methods can be present along with abstract methods in abstract classes.

Variable types: Static and final variables can only be declared in the case of interfaces, whereas abstract classes can also have non-static and non-final variables.

Inheritance: Multiple inheritances are facilitated by interfaces, whereas abstract classes do not promote multiple inheritances.

Data member accessibility: By default, the class data members of interfaces are of the public- type. Conversely, the class members for an abstract class can be protected or private also.

Implementation: With the help of an abstract class, the implementation of an interface is easily possible. However, the converse is not true;

Abstract class example:

```
public abstract class Athlete {

public abstract void walk();
```

}

Interface example:

public interface Walkable {

void walk();

}

4: What differences exist between HashMap and Hashtable?

Answer

There are several differences between HashMap and Hashtable in Java:

Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.

Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.

One of HashMap's subclasses is LinkedHashMap, so in the event that you'd want predictable iteration order (which is insertion order by default), you could easily swap out the HashMap for a LinkedHashMap. This wouldn't be as easy if you were using Hashtable.

5: What is the difference between throw and throws?

Answer

The throw keyword is used to explicitly raise a exception within the program. On the contrary, the throws clause is used to indicate those exceptions that are not handled by a method. Each method must explicitly specify which exceptions does not handle, so the callers of that method can guard against possible exceptions. Finally, multiple exceptions are separated by a comma.

6: Why would we want to use multiple threads in our application? Why not just stick with the main thread?

There are two main reasons for this:

First of all, we sometimes want to perform a task that's going to take a long time. For example, we might want to query a database, or we might want to fetch data from somewhere on the Internet. We could do this on the main thread, but the code within each main thread executes in a linear fashion. The main thread won't be able to do anything else while it's waiting for the data.

The second way of putting this is that the execution of the main thread will be suspended. It has to wait for the data to be returned before it can execute the next line of code. To the user, this could appear as if our application has died or is frozen, especially when we're dealing with a UI application.

Therefore, instead of tying up the main thread, we can create multiple threads and execute the long-running task on those threads. This would free up the main thread, so that it can continue executing. This process is called Multithreading. It can report progress or accept user input while the long-running task continues to execute in the background.

Adv JAVA Question:

1: What is @RequestMapping annotation in Spring Boot?

The @RequestMapping annotation is used to provide routing information. It tells to the Spring that any HTTP request should map to the corresponding method. We need to import org.springframework.web.annotation package in our file.

2: What is Spring IOC Container?

At the core of the Spring Framework, lies the Spring container. The container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.

3: **What do you mean by Dependency Injection?**

In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don’t connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

4: **How many bean scopes are supported by Spring?**

The Spring Framework supports five scopes. They are:

- **Singleton:** This provides scope for the bean definition to single instance per Spring IoC container.
- **Prototype:** This provides scope for a single bean definition to have any number of object instances.
- **Request:** This provides scope for a bean definition to an HTTP-request.
- **Session:** This provides scope for a bean definition to an HTTP-session.
- **Global-session:** This provides scope for a bean definition to an Global HTTP-session.

5: **What do you understand by auto wiring and name the different modes of it?**

The Spring container is able to autowire relationships between the collaborating beans. That is, it is possible to let Spring resolve collaborators for your bean automatically by inspecting the contents of the BeanFactory.
Different modes of bean auto-wiring are:

a. **no:** This is default setting which means no autowiring. Explicit bean reference should be used for wiring.

b. **byName:** It injects the object dependency according to name of the bean. It matches and wires its properties with the beans defined by the same names in the XML file.

c. **byType:** It injects the object dependency according to type. It matches and wires a property if its type matches with exactly one of the beans name in XML file.

d. **constructor:** It injects the dependency by calling the constructor of the class. It has a large number of parameters.

e. **autodetect:** First the container tries to wire using autowire by *constructor*, if it can't then it tries to autowire by *byType*.