Java Interview Questions:

**1.What is String**

string is an object that represents a sequence of characters.

**2.How to Create a string?**

There are two ways to create String object:

1. By string literal
2. By new keyword

**String literal:**

String s="welcome";

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool.

String literal to make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

**New Keyword**

String s=**new** String("Welcome");

will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

**3.Why String is immutable?**

* The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, you can use StringBuffer and StringBuilder classes.
* The String pool cannot be possible if String is not immutable in Java. A lot of heap space is saved by JRE. The same string variable can be referred to by more than one string variable in the pool. String interning can also not be possible if the String would not be immutable.
* If we don’t make the String immutable, it will pose a serious security threat to the application. For example, database usernames, passwords are passed as strings to receive database connections. The socket programming host and port descriptions are also passed as strings. The String is immutable, so its value cannot be changed. If the String doesn’t remain immutable, any hacker can cause a security issue in the application by changing the reference value.
* The String is safe for multithreading because of its immutableness. Different threads can access a single “String instance”. It removes the synchronization for thread safety because we make strings thread-safe implicitly.
* Immutability gives the security of loading the correct class by Classloader. For example, suppose we have an instance where we try to load java.sql.Connection class but the changes in the referenced value to the myhacked.Connection class does unwanted things to our database.

**String and Memory usage**

* it’s very common for String literals to occupy a large area of memory, which can even cause redundancy. So, in order to make Java more efficient, **the JVM sets aside a special area of memory called the “String constant pool”.**
* When the compiler sees a String literal, it looks for the String in the pool. If a match is found, the reference to the new literal is directed to the existing String and no new String object is created. The existing String simply has one more reference. Here comes the point of making String objects immutable:
* In the String constant pool, a String object is likely to have one or many references. If several references point to the same String without even knowing it, it would be bad if one of the references modified that String value. That’s why String objects are immutable.
* Well, now you could say, what if someone overrides the functionality of the String class? That’s the reason that **the String class is marked final** so that nobody can override the behavior of its methods.

**4.Java 8 Features**

Lambda Expressions

Functional Interfaces

default and static methods in Interfaces

Method Reference

Stream API

forEach() method

Optional Class

5.