B1. Can String be referred as a datatype?

No. String is not a data type in Java. The primitive data types in Java are byte, short, int, long, float, double, boolean, char as explained in [Primitive Data Types](http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html) from the Official Java Docs.

Some people think String is a data type since it offers direct assignment of a value. For example, the following is valid:

**B2. What is toString () method?**

The method is used to get a String object representing the value of the Number Object.

Implementing toString method in java is done by overriding the Object’s toString method. The java toString() method is used when we need a string representation of an object. It is defined in Object class. This method can be overridden to customize the String representation of the Object. Below is a program showing the use of the Object’s Default toString java method.

**B3. All the String objects created using String literals are stored in String pool?**

As the name suggests, **String Pool in java** is a pool of Strings stored in [Java Heap Memory](https://www.journaldev.com/4098/java-heap-space-vs-stack-memory). We know that String is special class in java and we can create String object using new operator as well as providing values in double quotes.

String Pool is possible only because [String is immutable in Java](https://www.journaldev.com/802/string-immutable-final-java) and its implementation of [String interning](https://en.wikipedia.org/wiki/String_interning) concept. String pool is also example of [Flyweight design pattern](https://www.journaldev.com/1562/flyweight-design-pattern-java).

String pool helps in saving a lot of space for Java Runtime although it takes more time to create the String.

When we use double quotes to create a String, it first looks for String with the same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference.

However using new operator, we force String class to create a new String object in heap space. We can use intern() method to put it into the pool or refer to another String object from the string pool having the same value.

**B4. Difference between StringBuffer and StringBuilder.**

StringBuffer and StringBuider are mutable classes. StringBuffer is thread safe and synchronized whereas StringBuilder is not, thats why StringBuilder is more faster than StringBuffer. String concat + operator internally uses StringBuffer or StringBuilder class.

**B5. What is Immutable?**

Immutable simply means unmodifiable or unchangeable. Once string object is created its data or state can't be changed but a new string object is created.

**B6. Is String is Immutable?**

Yes, Effectively, StringBuffer is mutable, its String representation isn't. In Java, all strings are immutable. When you are trying to modify a String , what you are really doing is creating a new one. However, when you use a StringBuilder , you are actually modifying the contents, instead of creating a new one

**B7. How do you create a String object?**

String str = “Hell”;

String str = new String(“Hello”);

**B8. What is the difference between creating String object using new and String literals?**

At high level both are String object, but main difference comes from the point that new() operator always creates a new String object. Also when you create String using literal they are interned. This will be much more clear when you [compare two String objects](http://javarevisited.blogspot.sg/2012/03/how-to-compare-two-string-in-java.html) created using String literal and new operator.

Similarly when you compare a String literal with an String object created using new() operator using == operator, it will return false,

**B9. What is Java String Pool?**

As the name suggest, String pool is a pool, or else a set, of [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) objects located in a special place in [Java Heap](https://examples.javacodegeeks.com/java-basics/java-heap-space-everything-you-need-to-know/). Java creators introduced this String pool construct as an optimization on the way String objects are allocated and stored.

String is one of the most used types in Java, and it is a costly one when it comes to memory space. For example, [a 4 character long String requires 56 bytes of memory](https://www.cs.virginia.edu/kim/publicity/pldi09tutorials/memory-efficient-java-tutorial.pdf).

Which shows that only 14% percent of the allocated memory is the actual data, the 4 characters. So, a lot of overhead there. Naturally, an optimization should be implemented on how String objects are going to be stored in the heap. This is why the String pool was created. It is a simple implementation of the [Flyweight](http://www.oodesign.com/flyweight-pattern.html) pattern, which in essence, says this : when a lot of data is common among several objects, it is better to just share the same instance of that data than creating several different “copies” of it. Applying that to Strings, its better to share the same String object than creating multiple instances of String objects with the same value.

**B10. Immutable objects are thread-safe?**

So, Immutable objects are always thread-safe, but their references may not be. To make their references thread-safe, we may need to access them from synchronized blocks/methods. An immutable object is an object that is no longer modified once it has been constructed

**B11. Overriding toString () example.**

class Student{

int id;

String name;

String address;

Student(int id, String name, String address){

this.id=id;

this.name=name;

this.address=address;

}

public static void main(String args[]){

Student s1=new Student(100,”Joe”,”success”);

Student s2=new Student(50,”Jeff”,”fail”);

System.out.println(s1);//compiler writes here s1.toString()

System.out.println(s2);//compiler writes here s2.toString()

}

}

**B12. What is String interpolation in Java?**

Strings that contain references to variables or expressions are called interpolated strings. When you use interpolated strings, the embedded expressions and variables are evaluated and the result is inserted into the string. The act of inserting data into a string is called string interpolation.

**B13. Name the interfaces that Java String class implements.**

The CharSequence interface is used to represent the sequence of characters. String, StringBuffer and StringBuilder classes implement it.

**B14. How do I compare strings in Java?**

Using String.equals() :In Java, string equals() method compares the two given strings based on the data/content of the string. If all the contents of both the strings are same then it returns true. If all characters do not match, then it returns false.

**B15. Can we have case null in string switch case?**

The answer is simply that if you use a switch with a reference type (such as a boxed primitive type), the run-time error will occur if the expression is null because unboxing it would throw the NPE. so case null (which is illegal) could never be executed anyway ;)

**B16. What is the default implementation of equals method in Object class?**

In java equals() method is used to compare equality of two Objects. The equality can be compared in two ways: Shallow comparison: The default implementation of equals method is defined in Java.lang.Object class which simply checks if two Object references (say x and y) refer to the same Object.