# Assignment – 11th (Strings in Java)

## 1. What is a String in Java?

**Ans: -** In Java, a String is an object that represents a sequence of characters. Strings are widely used in Java programs to represent text data, such as names, addresses, product descriptions, and more. Strings are immutable in Java, which means that once a String object is created, its value cannot be changed.

## 2. Types of String in Java are?

**Ans: -** There are two types of Strings in Java:

- **String literal: -** A String literal is created by enclosing a sequence of characters in double quotes. For example: **"Hello, World!"**. String literals are stored in a special memory area called the string pool, where the JVM keeps track of all the unique String literals used in the program. If two String literals have the same value, they will refer to the same object in the string pool.
- **String object: -** A String object is created by using the ***`new`*** operator and calling the ***`String`*** class constructor. For example: **`new String("Hello, World!")`**. String objects are stored in the heap memory, and every time a new String object is created, a new instance is created in the heap memory, even if the value of the object is the same as another String object.

  Here's an example that demonstrates the difference between String literals and String objects:

```
String str1 = "Hello";
String str2 = "Hello";
String str3 = new String( "Hello");

System.out.println(str1==str2);      // true
System.out.println(str1==str2);      // false
```

In this example, ***`str1`*** and ***`str2`*** are both String literals and refer to the same object in the string pool. On the other hand, `str3` is a String object created using the ***`new`*** keyword and refers to a different instance in the heap memory. The ***`==`*** operator compares the reference values of the two objects, and since ***`str1`*** and ***`str2`*** refer to the same object, the comparison returns ***`true`***, while the comparison between ***`str1`*** and ***`str3`*** returns ***`false`***.

## 3. In how many ways can you create string objects in Java?

**Ans: -** There are several ways to create String objects in Java:

- **Using string literals: -** You can create a String object by enclosing a sequence of characters in double quotes. For example: **`String str = "Hello, World!";`** .
- **Using the *`new`* Keyword: -** You can create a String object by using the new operator and calling the String class constructor. For example: **`String str = new String("Hello, World!");`**.
- **Using the *`StringBuilder class`*: -** You can create a String object by using the `StringBuilder` class and its **`toString()`** method. The **`StringBuilder`** class is used to create and manipulate strings efficiently.

```
StringBuilder sb = new StringBuilder( "Hello");
sb.append(", World!");
String str = sb.toString();
```

- **Using the *`StringBuffer`* class: -** You can create a String object by using the **`StringBuffer`** class and its **`toString()`** method. The **`StringBuffer`** class is similar to the **`StringBuilder`** class but is thread-safe, which means that it's safe to use in a multi-threaded environment.

```
StringBuilder sb = new StringBuilder( "Hello");
sb.append(", World!");
String str = sb.toString();
```

## 4. What is a string constant pool?

**Ans: -** The string constant pool is a special memory area in the Java Virtual Machine (JVM) where all unique String literals used in a Java program are stored. The JVM maintains a pool of String literals so that if two String literals have the same value, they will refer to the same object in the pool, thereby saving memory and improving performance. For example, consider the following code:

```
String str1 = "Hello";
String str2 = "Hello";
System.out.println(str1==str2);       // true
```

In this example, ***`str1`*** and ***`str2`*** are String literals with the same value, and they both refer to the same object in the string constant pool. This is why the comparison between ***`str1`*** and ***`str2`*** returns ***`true`***.

## 5. What do you mean by mutable and immutable objects?

**Ans: -** In computer programming, mutable and immutable objects refer to the characteristics of objects in terms of whether their state can be changed after they are created.

1. **Mutable Objects: -** A mutable object is an object whose state can be modified after it has been created. For example, a list in Python is a mutable object. You can add, remove, or modify elements in the list after it has been created.

2. **Immutable Objects: -** An immutable object is an object whose state cannot be changed after it has been created. For example, a tuple in Python is an immutable object. Once you have created a tuple, you cannot modify its elements. If you need to modify the elements of a tuple, you must create a new tuple with the desired elements.

In Java, the String class is an example of an immutable object, while the StringBuilder class is an example of a mutable object.

In general, mutable objects are more flexible than immutable objects, but they can also be more difficult to use correctly and can lead to bugs if not handled carefully. Immutable objects are simpler and safer to use, but they can be less flexible and may require more memory and processing time to create new instances when their state needs to change.

## 6. Where exactly is the string constant pool located in the memory?

**Ans: -** The string constant pool is located in the Java heap, which is the area of memory where objects in a Java program are stored. However, the string constant pool is a special area of the heap that is optimized for strings.

The JVM uses the string constant pool to store all unique String literals used in a Java program. When a String literal is created in a Java program, the JVM first checks the string constant pool to see if a String object with the same value already exists. If it does, the reference to the existing String object is returned, and a new String object is not created.

This optimization helps reduce the memory usage of a Java program and improve its performance, since multiple String literals with the same value will refer to the same object in the string constant pool.