Strings in Java

Assignment-11

1.What is a String in Java ?

Ans: In Java, a string is a sequence of characters that can be manipulated and stored as an object. A string is created by enclosing a sequence of characters in double quotes.

For example:

String greeting = "Hello, World!";

Strings in Java are objects of the **java.lang.String** class, which is a final class and cannot be subclassed. The **String** class

provides a variety of methods for manipulating and examining strings, such as **length(), charAt(), substring(), toUpperCase(), toLowerCase(), etc.**

It is important to note that once a string is created, it cannot be modified. Any operations that appear to modify a string actually result in the creation of a new string object.

2.Types of String in Java Are ?

1. Ans: Immutable vs Mutable: Strings in Java are immutable, meaning that once a string is created, it cannot be modified

.

Any operations that appear to modify a string actually result in the creation of a new string object. Mutable strings can be

represented using the **StringBuilder** or **StringBuffer** classes.

2. Literal vs Object: A string literal is created by enclosing a sequence of characters in double quotes, while a string object is

created using the **new** operator followed by a call to the constructor of the **String** class. String literals are stored in the string

pool, while string objects are stored in the heap memory.

3. Thread-Safe vs Non-Thread-Safe: The **StringBuilder** class is non-thread-safe, meaning that it cannot be safely used in a

4. multi-threaded environment. The **StringBuffer** class, on the other hand, is thread-safe and can be used in a multi-threaded environment.

Overall, the classification of strings in Java is based on the purpose and use-case of the strings, as well as the requirement for thread-safety, immutability, and memory management.

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

Ans: There are several ways to create string objects in Java:

1. String literal: A string literal is created by enclosing a sequence of characters in double quotes. For example: **String**

greeting = "Hello, World!";. String literals are stored in the string pool, which is a special area in the heap memory used to

store strings.

2. Using the **new** operator: A string object can be created using the **new** operator, followed by a call to the constructor of the

String class. For example: **String greeting = new String("Hello, World!");**. This method creates a new string object in

the heap memory, separate from the string pool.

3. Concatenating strings: A new string object can be created by concatenating two or more strings using the **+** operator. For

example: **String greeting = "Hello, " + "World!";**.

4. Converting from primitive data types: A string object can be created from a primitive data type, such as **int**, **float**, or

char, using the **valueOf()** method. For example: **String intString = String.valueOf(42);**.

5. From a character array: A string object can be created from a character array using the **String** class's constructor that

takes a **char[]** as an argument. For example: **char[] helloArray = {'H', 'e', 'l', 'l', 'o'}; String helloString = new**

String(helloArray);.

These are the main ways to create string objects in Java. The method used depends on the requirement and the source of the string data.

4.What is a String Constant pool ?

Ans: The string constant pool is a special area of memory in the Java heap that is used to store string literals. When a string literal

is created in Java, the JVM checks the string constant pool to see if an identical string already exists. If it does, the JVM reuses

the existing string in the pool, rather than creating a new string object in the heap memory. This helps to reduce memory usage

and improve performance, as multiple references to the same string literal will point to the same object in the pool.

For example, consider the following code:

String str1 = "Hello";

String str2 = "Hello";

In this example, both **str1** and **str2** reference the same string object in the string constant pool, even though two separate string variables are created.

It's worth noting that string objects created using the **new** operator are not stored in the string constant pool. Instead, they are

stored in the heap memory as separate objects, even if they have the same value as a string in the constant pool. For example:

String str1 = "Hello";

String str2 = new String("Hello");

In this example, **str1** references the string object in the string constant pool, while **str2** references a separate string object in the heap memory.

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

Ans: In computer programming, mutable and immutable objects refer to the behavior of objects with respect to changes to their state or value.

An immutable object is an object whose state cannot be changed after it is created. Once an immutable object is created, any

attempt to modify its state will result in a new object being created with the modified state. For example, in Java, the **String** class

represents an immutable object, as once a string object is created, it cannot be modified. Any operation that appears to modify a

string actually results in a new string object being created.

On the other hand, a mutable object is an object whose state can be changed after it is created. For example, in Java, the

StringBuilder class represents a mutable object, as it can be used to dynamically build and modify a string.

n general, immutable objects are safer to use in multi-threaded environments, as they cannot be corrupted by concurrent access,

while mutable objects can lead to problems with thread safety if not used properly. Immutable objects also have advantages in

terms of memory usage and performance, as they can be easily shared and reused, while mutable objects often require more

memory and can be slower to use.

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

Ans: The string constant pool is located in the heap memory of the Java virtual machine (JVM). The heap is a region of

memory used to store objects and other data that is dynamically allocated at runtime. The string constant pool is a

special area within the heap that is used to store string literals

When a string literal is created in Java, the JVM checks the string constant pool to see if an identical string already

exists. If it does, the JVM reuses the existing string in the pool, rather than creating a new string object in the heap

memory. This helps to reduce memory usage and improve performance, as multiple references to the same string

literal will point to the same object in the pool.

It's important to note that the JVM is responsible for managing the string constant pool, and the exact location of the

pool within the heap may vary from one JVM implementation to another. However, the concept of the string constant

pool as a special area within the heap used to store string literals is a fundamental part of the Java language and is

implemented consistently across all JVM implementations.