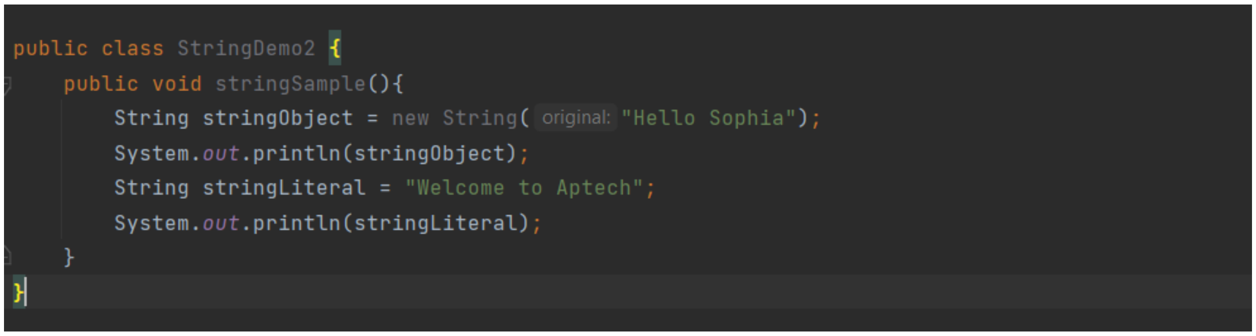
HOW JAVA STORES STRING

Strings are used to store a sequence of characters and are treated as objects in java.

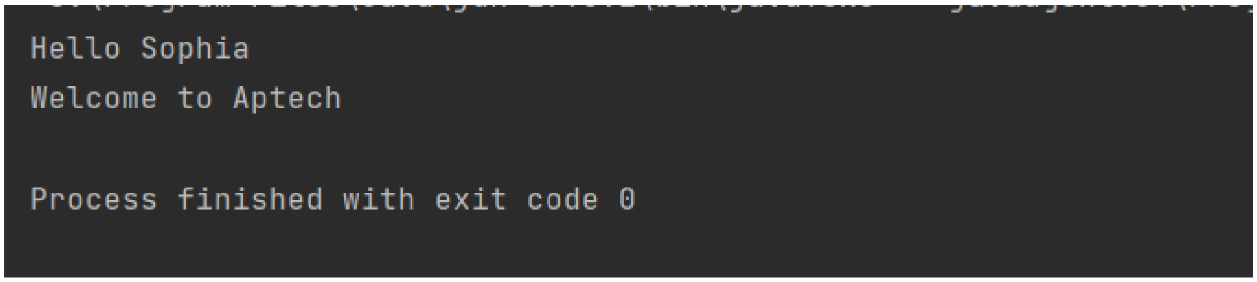
You can create a string by using the new keyword or by assigning value to the variable like primitive datatypes.

Example



```
public class StringDemo2 {
    public void stringSample(){
        String stringObject = new String( original: "Hello Sophia");
        System.out.println(stringObject);
        String stringLiteral = "Welcome to Aptech";
        System.out.println(stringLiteral);
    }
}
```

Output



```
Hello Sophia
Welcome to Aptech

Process finished with exit code 0
```

STORAGE OF STRINGS

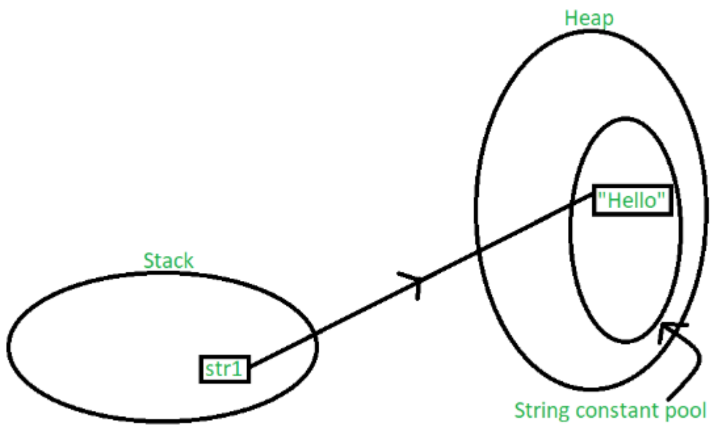
Strings are stored in the heap area in a separate memory location known as string constant pool.

A String Constant Pool is a separate place in the heap memory where the values of all the strings which are defined in the program are stored.

When we declare a string, an object of type String is created in the stack, while an instance with the value of the string is created in the heap. On standard assignment of a value to a string variable, the variable is allocated stack, while the value is stored in the heap in the string constant pool. For example, let’s assign some value to a string str1. In java, a string is defined and the value is assigned as:

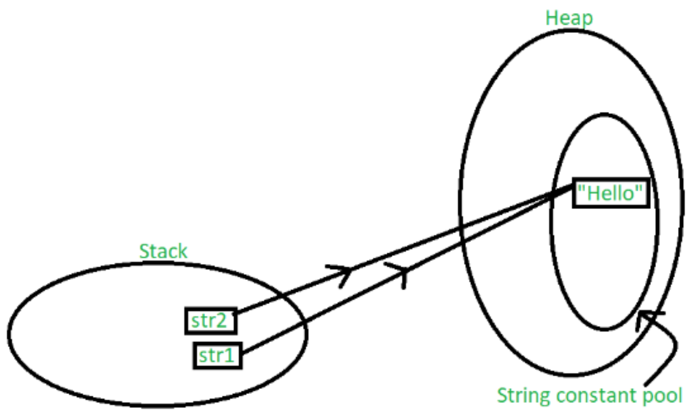
String str1 = “Hello”;

The following illustration explains the memory allocation for the above declaration:



In the above scenario, a string object is created in the stack, and the value “Hello” is created and stored in the heap. Since we have normally assigned the value, it is stored in the constant pool area of the heap. A pointer points towards the value stored in the heap from the object in the stack. Now, let’s take the same example with multiple string variables having the same value **as follows**:
String str1 = “Hello”;
String str2 = “Hello”;

The following illustration explains the memory allocation for the above declaration:



In this case, both the string objects get created in the stack, but another instance of the value “Hello” is not created in the heap. Instead, the previous instance of “Hello” is re-used. The *string constant pool* is a small cache that resides within the heap. Java stores all the *values* inside the string constant pool on direct allocation. This way, if a similar value needs to be accessed again, a new string object created in the stack can reference it directly with the help of a pointer.

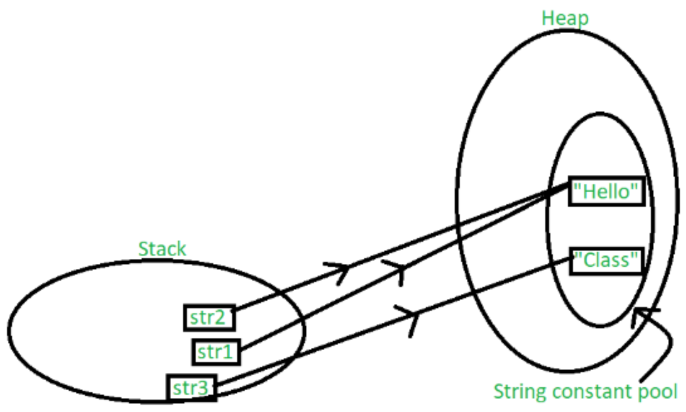
When a string object is assigned a different value, the new value will be registered in the string constant pool as a separate instance. Lets understand this with the following **example**:

String str1 = “Hello”;

String str2 = “Hello”;

String str3 = “Class”;

The following illustration explains the memory allocation for the above declaration:



One way to skip this memory allocation is to use the new keyword while creating a new string object. The ‘new’ keyword forces a new instance to always be created regardless of whether the same value was used previously or not. Using ‘new’ forces the instance to be created in the heap outside the string constant pool which is clear, since caching and re-using of instances isn’t allowed here.