String

- String is a literal(data).It is group of character that is enclosed within the double quotes “ ”.
- In java we can store a String by creating instance of following classes.

1.java.lang.String

2.java.lang.StringBuffer

3.java.lang.StringBuilder

- In java whenever we create a String complier implicitly create an instance for java.lang.String in String pool area or String constant pool(scp)

There are two ways to create String object:

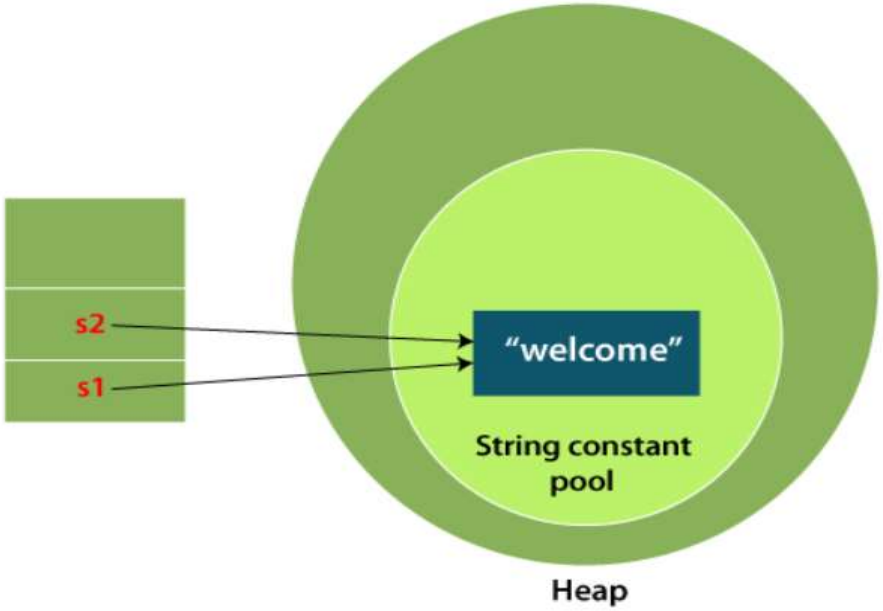
- By string literal
- By new keyword

- 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.

For example:

String s1="Welcome";

String s2="Welcome";//It doesn't create a new instance



Characteristics of String literal

- When a String is used in a java program an instance of java.lang.String class is created inside String constant pool.
- For the given String literal, if the instance of a string is already present, then the new instance is not created instead reference of existing instance is given.

By new keyword

```
String s=new String("Welcome");//creates two objects
```

- One object in String constant pool
- One object in Heap area.

String class

- String is a inbuilt class defined in java.lang package.
- It is final class
- It inherits java.lang.Object
- In a String class toString(),equals(),hashCode() methods of java.lang.Object class are Overridden.

Characteristics of String class

- Instance of String class is immutable in nature.(Once object created then the state cannot be modified).
- If we try to manipulate(modify) the state/data then new object is created and reference is given.

Disadvantage of java.lang.String class:

- Immutability, because for every modification separate object is created in a memory, it reduces the performance.

Note:

To Overcome the disadvantage of String class we can go for StringBuilder and StringBuffer class

Methods of String class

Return type	Method Name	Description
String	toUpperCase()	Converts the specified String to uppercase
String	toLowerCase()	Converts the specified String to LowerCase
String	concat(String s)	Joins the specified Strings
String	trim()	Remove the space present before and after the String
String	substring(int index)	Extract a characters from a String object start from specified index ends at the end of the String

Return type	Method name	Description
char	charAt()	Returns the character at the specified index (position)
int	length()	Returns the length of a specified string
char[]	toCharArray()	Converts this string to a new character array
boolean	equalsIgnoreCase()	Compares two strings, ignoring case considerations
int	indexOf()	Returns the position of the first found occurrence of specified characters in a string
boolean	isEmpty()	Checks whether a string is empty or not
int	lastIndexOf()	Returns the position of the last found occurrence of specified characters in a String

StringBuffer

- It is inbuilt class defined in java.lang package.
- It is a final class.
- It helps to create mutable instance of String
- StringBuffer does not have String Constant Pool.
- It inherits java.lang.Object class.
- In StringBuffer equals(),hashCode() methods of java.lang.Object class are **not Overridden.**

Characteristics of StringBuffer:

- It is mutable

Note:

String Constant Pool is not Applicable to StringBuffer.

Disadvantage of StringBuffer

- Multiple threads cannot execute the StringBuffer object simultaneously because all the methods are synchronised.
- So execution time is more, In order to overcome this problem we go for StringBuilder.

Note:

The characteristics of StringBuilder and StringBuffer are same.

Difference between StringBuilder and StringBuffer

String Buffer	StringBuilder
All the methods present in StringBuffer are Synchronised.	All the methods present in StringBuilder are **non-synchronised.**
At a time only one thread is allowed to access StringBuffer object. Hence it is thread safe.	At a time multiple threads are allowed to access StringBuilder object. Hence is **not** thread safe.
Threads are required to wait to operate a StringBuffer object. Hence relatively performance is low.	Threads are not required to wait to operate a StringBuilder Object. Hence relatively performance is high.
Less efficient than StringBuilder class	Efficiency is high compared to StringBuffer class
Introduced in 1.0v.	Introduced in 1.5v.