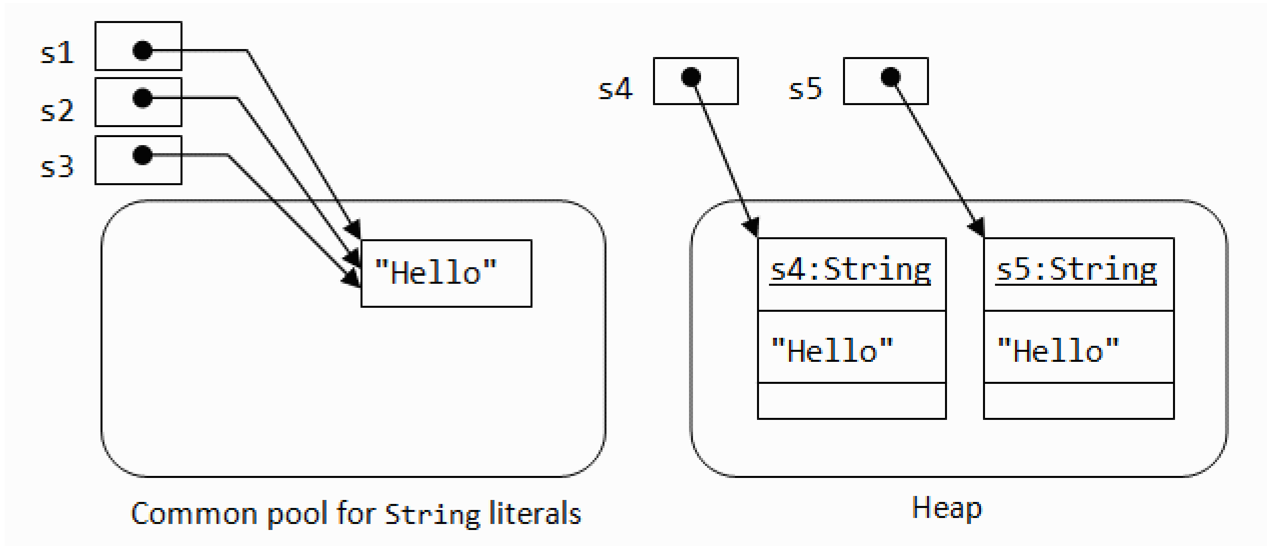
Strings

```
String s1="hello"
String s2="hello"
String s4=new String("hello");
String s5=new String("hello");
```



- the String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.
- Strings are Immutable; their values cannot be changed after they are created
- String str = "abc"; is equivalent to:

  String str = new String(data);

String library functions

- public int length() returns length of the String.

- public char **charAt**(int index) Returns the char value at the specified index. An index ranges from 0 to length() - The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

- public boolean **equals**(Object anObject) Compares this string to the specified object. The result is true if and only if the argument is not null and is a String object that represents the same sequence of characters as this object.

- public boolean **equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations. Two strings are considered equal ignoring case if they are of the same length, and corresponding characters in the two strings are equal ignoring case.

- public int **compareTo**(String anotherString) Compares two strings lexicographically. The comparison is based on the Unicode value of each character in the strings. The character sequence represented by this String object is compared lexicographically to the character sequence represented by the argument string. The result is a negative integer if this String object lexicographically precedes the argument string. The result is a positive integer if this String object lexicographically follows the argument string. The result is zero if the strings are equal;

- public int **indexOf**(int ch) Returns the index within this string of the first occurrence of the specified character. If a character with value ch occurs in the character sequence represented by this String object, then the index (in Unicode code units) of the first such occurrence is returned.

- public int **indexOf**(int ch, int fromIndex) Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

- public int **lastIndexOf**(int ch) Returns the index within this string of the last occurrence of the specified character

- public int **indexOf**(String str) Returns the index within this string of the first occurrence of the specified substring
- public int **indexOf**(String str,
int fromIndex)
- public int **lastIndexOf**(String str)
- public int **lastIndexOf**(String str,
int fromIndex)

- public String **substring**(int beginIndex) Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

Examples:

```
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)
```

- public String **substring**(int beginIndex, int endIndex) Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.
- Examples:
- "hamburger".substring(4, 8) returns "urge"
  "smiles".substring(1, 5) returns "mile"

- public String **toLowerCase**() Converts all of the characters in this String to lower case

- public String **toUpperCase**() Converts all of the characters in this String to upper case

- **valueOf**
- public static String **valueOf**(int i): converts the passed argument type to String. (same for other data types)

- **endsWith**
- public boolean **endsWith**(String suffix)Tests if this string ends with the specified suffix.

- **getChars**
- public void **getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin)Copies characters from this string into the destination character array.

- **replace**
- public String **replace**(char oldChar, char newChar)Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

- **startsWith**
- public boolean **startsWith**(String prefix)
- Tests if this string starts with the specified prefix.

- **valueOf**
- public static String **valueOf**(boolean b)
- Returns the string representation of the boolean argument.
- (same for all data types)

StringBuffer

- A string buffer is like a String, but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.
- Eg: StringBuffer sb=new StringBuffer(str)
- Where str is a string literal

Methods of StringBuffer class

- append(<data>)
- charAt(int index)
- delete(int begin,int end)
- deleteCharAt(int index)
- Insert(int index,<datatype> data)
- reverse()
- setCharAt(int index)
- Capacity

(refer documentation)

StringBuilder

- A mutable sequence of characters. This class provides an API compatible with StringBuffer, but with no guarantee of synchronization. This class is designed for use as a drop-in replacement for StringBuffer in places where the string buffer was being used by a single thread (as is generally the case). Where possible, it is recommended that this class be used in preference to StringBuffer as it will be faster under most implementations.

- BrainDrill