



String and StringBuffer

Agenda

1

String and StringBuffer

Objectives

At the end of this module, you will be able to:

- Implement String and StringBuffer class methods

String and StringBuffer



String

- String is a group of characters. They are objects of type String.
- Once a String object is created it cannot be changed. Strings are Immutable.
- To get changeable strings use the class called StringBuffer.
- String and StringBuffer classes are declared as final, so there cannot be subclasses of these classes.
- The default constructor creates an empty string.

```
String s = new String();
```

Creating Strings

- To Create a String in JAVA is

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
```

```
String str = new String(data);
```

- If data array in the above example is modified after the string object str is created, then str remains unchanged.
- Construct a string object by passing another string object.

```
String str2 = new String(str);
```

String class Methods

- The `length()` method returns the length of the string.

Eg: `System.out.println("Varun".length()); // prints 5`

- The `+` operator is used to concatenate two or more strings.

Eg: `String myName = "Varun";`

`String s = "My name is" + myName + ".";`

- For string concatenation the Java compiler converts an operand to a `String` whenever the other operand of the `+` is a `String` object.

String class Methods (Contd.).

Characters in a string can be retrieved in a number of ways

- `public char charAt(int index)`

Method returns the character at the specified index. An index ranges from 0 to `length() - 1`

- `char c;`

```
c = "abc".charAt(1); // c = "b"
```


String class Methods (Contd.).

- **equals() Method-** This method is used to compare the invoking String to the object specified. It will return true, if the argument is not null and it is String object which contains the same sequence of characters as the invoking String.

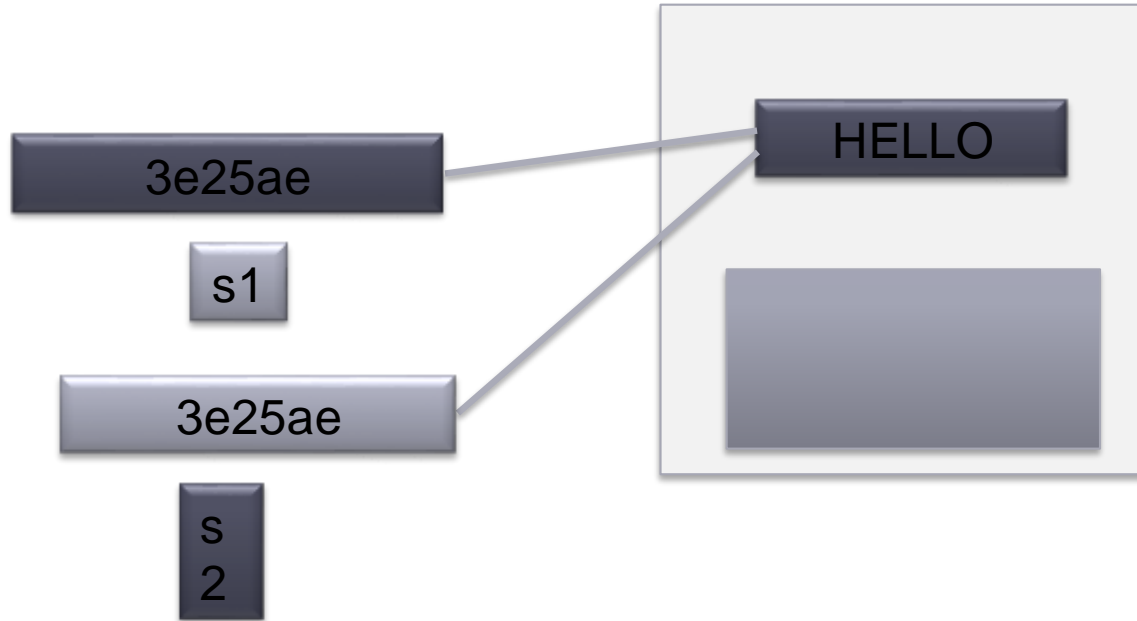
```
public boolean equals(Object anObject)
```

- **equalsIgnoreCase() Method-** 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 boolean equalsIgnoreCase(String anotherString)
```

Strings created using assignment operator

- `String s1 = "HELLO";`
- `String s2 = "HELLO";`



Comparing Strings using == operator

What is the output ?

```
public class StringTest{  
    public static void main(String[] args){  
        String s1="Hello";  
        String s2="Hello";  
        if(s1==s2)  
            System.out.println("String objects referenced are same");  
        else  
            System.out.println("String objects referenced are not same");  
    }  
}
```

Output: String objects referenced are same

Comparing Strings using equals method

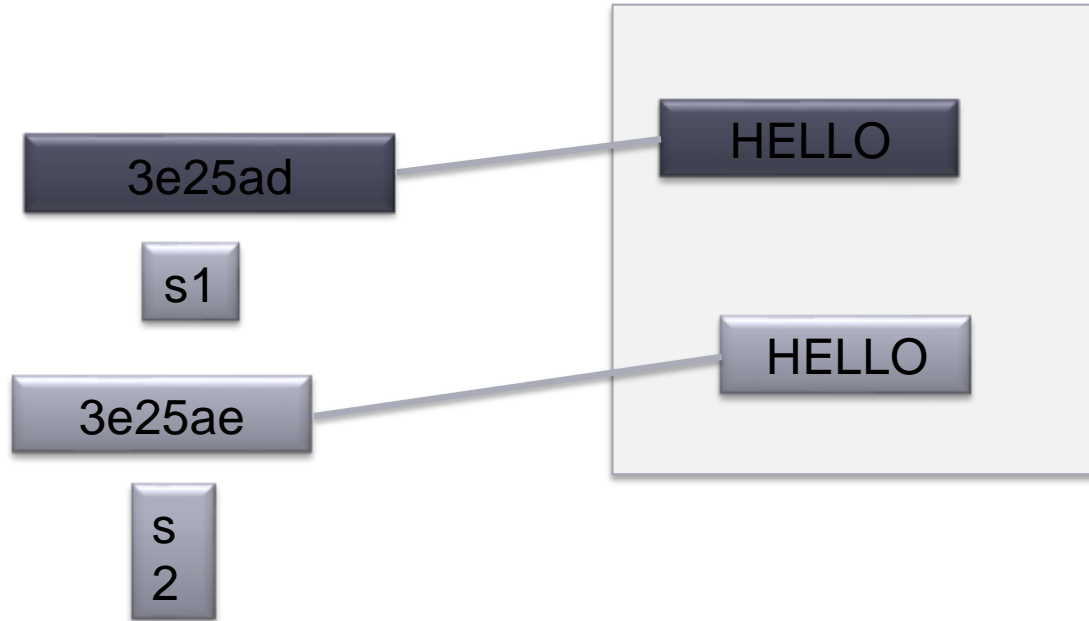
What is the output ?

```
public class StringTest{  
    public static void main(String[] args){  
        String s1="Hello";  
        String s2="Hello";  
        if(s1.equals(s2))  
            System.out.println("Strings are equal");  
        else  
            System.out.println("Strings are not equal");  
    }  
}
```

Output: Strings are equal

Strings created using new keyword

- `String s1 = new String("HELLO");`
- `String s2 = new String("HELLO");`



Comparing Strings using == operator

What is the output ?

```
public class StringTest{  
    public static void main(String[] args){  
        String s1= new String("Hello");  
        String s2= new String("Hello");  
        if(s1==s2)  
            System.out.println("String objects referenced are same ");  
        else  
            System.out.println("String objects referenced are not same");  
    }  
}
```

Output: String objects referenced are not same

Comparing Strings using equals method

What is the output ?

```
public class StringTest{  
    public static void main(String[] args){  
        String s1= new String("Hello");  
        String s2= new String("Hello");  
        if(s1.equals(s2))  
            System.out.println("Strings are equal");  
        else  
            System.out.println("Strings are not equal");  
    }  
}
```

Output: Strings are equal

String class Methods

startsWith() – Tests if this string starts with the specified prefix.

```
public boolean startsWith(String prefix)  
"January".startsWith("Jan"); // true
```

endsWith() - Tests if this string ends with the specified suffix.

```
public boolean endsWith(String suffix)  
"January".endsWith("ry"); // true
```


String class Methods (Contd.).

compareTo() - Compares two strings and to know which string is bigger or smaller

- We will get a negative integer, if this String object is less than the argument string
- We will get a positive integer if this String object is greater than the argument string.
- We will get a return value 0(zero), if these strings are equal.

```
public int compareTo(String anotherString)
public int compareToIgnoreCase(String str)
```

This method is similar to compareTo() method but this does not take the case of strings into consideration.

String class Methods (Contd.).

- **indexOf** – Searches for the first occurrence of a character or substring. Returns -1 if the character does not occur
- **public int indexOf(int ch)**- It searches for the character represented by ch within this string and returns the index of first occurrence of this character
- **public int indexOf(String str)** - It searches for the substring specified by str within this string and returns the index of first occurrence of this substring

```
String str = "How was your day today?";  
str.indexOf('t');  
str.indexOf("was");
```

String class Methods (Contd.).

- **public int indexOf(int ch, int fromIndex)**- It searches for the character represented by ch within this string and returns the index of first occurrence of this character starting from the position specified by fromIndex
- **public int indexOf(String str, int fromIndex)** - Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

```
String str = "How was your day today?";  
str.indexOf('a', 6);  
str.indexOf("was", 2);
```

String class Methods (Contd.).

- **lastIndexOf()** –It searches for the last occurrence of a particular character or substring
- **substring()** - This method returns a new string which is actually a substring of this string. It extracts characters starting from the specified index all the way till the end of the string

`public String substring(int beginIndex)`

Eg: "unhappy".substring(2) returns "happy"

`public String substring(int beginIndex, int endIndex)`

Eg: "smiles".substring(1, 5) returns "mile"

String class Methods (Contd.).

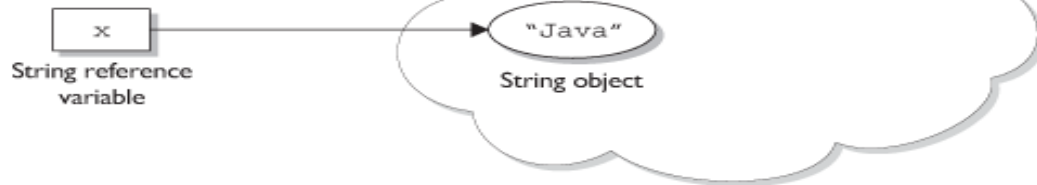
concat() - Concatenates the specified string to the end of this string

```
public String concat(String str)
```

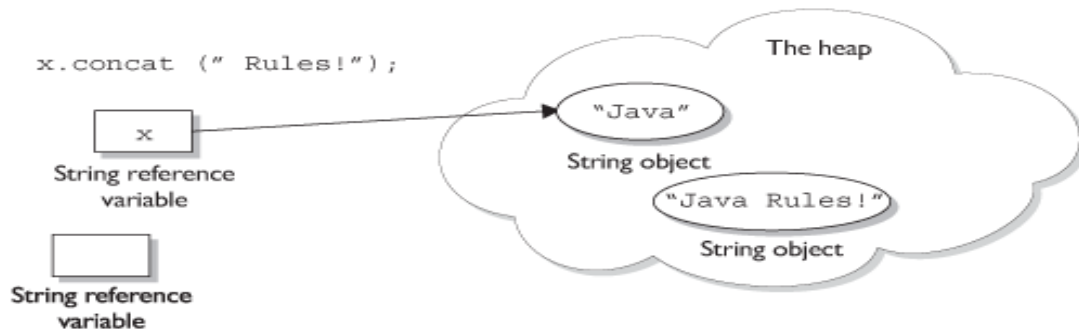
```
"to".concat("get").concat("her") returns "together"
```

Example

Step 1: `String x = "Java";`



Step 2: `x.concat (" Rules!");`



Notice that no reference variable is created to access the "Java Rules!" String.

String class Methods

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

```
public String replace(char oldChar, char newChar)
```

```
"wipra technalagies".replace('a', 'o') returns "wipro  
technologies"
```

String class Methods (Contd.).

- **trim()** - Returns a copy of the string, with leading and trailing whitespace omitted

```
public String trim()
```

```
String s = "  Hi Mom!  ".trim();  
S = "Hi Mom!"
```

- **valueOf()** – This method is used to convert a character array into String. The result is a String representation of argument passed as character array

```
public static String valueOf(char[] data)
```


String class Methods (Contd.).

Other forms are:

```
public static String valueOf(char c)
```

```
public static String valueOf(boolean b)
```

```
public static String valueOf(int i)
```

```
public static String valueOf(long l)
```

```
public static String valueOf(float f)
```

```
public static String valueOf(double d)
```

String class Methods (Contd.).

- **toLowerCase():** Method converts all of the characters in a String to lower case
- **toUpperCase():** Method converts all of the characters in a String to upper case

```
public String toLowerCase()
```

```
public String toUpperCase()
```

Eg: `"HELLO WORLD".toLowerCase();`
`"hello world".toUpperCase();`

StringBuffer

- StringBuffer class objects are mutable, so they can be modified
- The length and content of the StringBuffer sequence can be changed through certain method calls
- StringBuffer class defines three constructors:
 - *StringBuffer()*//empty object
 - *StringBuffer(int capacity)*//creates an empty object with a capacity for storing a string
 - *StringBuffer(String str)*//create StringBuffer object by using a string

String Buffer Operations

- StringBuffer has two main operations methods – *append* and *insert*
- Both these methods are overloaded so that they can accept any type of data

Here are few append methods:

```
StringBuffer append(String str)
```

```
StringBuffer append(int num)
```

- As the name suggests, the append method adds the specified characters at the end of the StringBuffer object

StringBuffer Operations (Contd.).

- The insert methods are used to insert characters at the specified index location

Here are few insert methods:

```
StringBuffer insert(int index, String str)
StringBuffer insert(int index, char ch)
```

- Index specifies at which point the string will be inserted into the invoking StringBuffer object

StringBuffer Operations (Contd.).

- **delete()** - This method is used to delete specified substring within the StringBuffer object

```
public StringBuffer delete(int start, int end)
```

StringBuffer Operations (Contd.)

- **replace()** - This method is used to replace part of this StringBuffer(substring) with another substring

```
public StringBuffer replace(int start, int end, String str)
```

- **substring()** - This method returns a new string which is actually a substring of this StringBuffer. It extracts characters starting from the specified index all the way till the end of the StringBuffer

```
public String substring(int start)
```

StringBuffer Operations (Contd.)

- **reverse()** - As the name suggests, the character sequence is reversed with this method

```
public StringBuffer reverse()
```

- **length()** – Used to find the length of the StringBuffer

```
public int length()
```


Example

```
class FullName{  
    public static void main(String[] args){  
        StringBuffer sb=new StringBuffer();  
        String surname="Mallapudi";  
        String lastname="vijaya";  
        sb.append(surname);  
        System.out.println("Name: "+sb);  
        int n=surname.length();  
        sb.insert(n,lastname);  
        System.out.println("Full name: "+sb);  
        System.out.println("In reverse = "+sb.reverse());  
    }  
}
```

StringBuffer Operations (Contd.).

- **capacity()** – We can find the capacity of the StringBuffer using this method

What is capacity ?

The capacity is the amount of storage available for the characters that have just been inserted

```
public int capacity()
```

- **charAt()** - Used to find the character at a particular index position

```
public char charAt(int index)
```

Example

```
public class stringBuffer{  
    public static void main(String[] args) throws Exception{  
        String str1="Hello";  
        try{  
            str1 += ", This program is a example of SringBuffer class and it's  
functions.";   
            //Create a object of StringBuffer class  
            StringBuffer strbuf1 = new StringBuffer();  
            System.out.print(strbuf1.length()+"\n");  
            strbuf1.append(str1);  
            System.out.println(strbuf1);  
            strbuf1.delete(0,str1.length());  
            //append()  
            strbuf1.append("Hello");  
            strbuf1.append("World");  
            System.out.println(strbuf1);  
            //insert()  
            strbuf1.insert(5,"_Java ");  
            System.out.println(strbuf1);  
        }  
    }  
}
```

Example(Contd.).

```
//reverse()  
  
strbuf1.reverse();  
  
    System.out.print("Reversed string : ");  
    System.out.println(strbuf1);  
    strbuf1.reverse();  
    System.out.println(strbuf1);  
    //setCharAt()  
    strbuf1.setCharAt(5, ' ');  
    System.out.println(strbuf1);  
    //charAt()  
    System.out.print("Character at 6th position : ");  
    System.out.println(strbuf1.charAt(6));  
    //substring()  
    System.out.print("Substring from position 3 to 6 : ");  
    System.out.println(strbuf1.substring(3,7));
```

Example(Contd.).

```
//deleteCharAt()  
  
    strbuf1.deleteCharAt(3);  
    System.out.println(strbuf1);  
  
    //capacity()  
    System.out.print("Capacity of StringBuffer object : ");  
    System.out.println(strbuf1.capacity());  
  
    //delete() and length()  
    strbuf1.delete(6, strbuf1.length());  
    System.out.println(strbuf1);  
  
}  
catch(StringIndexOutOfBoundsException e) {  
    System.out.println(e.getMessage());  
}  
  
}
```

Example(Contd.).

```
//deleteCharAt()  
  
    strbuf1.deleteCharAt(3);  
    System.out.println(strbuf1);  
    //capacity()  
    System.out.print("Capacity of StringBuffer object : ");  
    System.out.println(strbuf1.capacity());  
    //delete() and length()  
    strbuf1.delete(6, strbuf1.length());  
    System.out.println(strbuf1);  
}  
catch (StringIndexOutOfBoundsException e) {  
    System.out.println(e.getMessage());  
}  
}  
}
```

Quiz

What is the output ?

```
class StringExample {  
    public static void main(String[] args) {  
        String st = new String("Wipro Technologies");  
        StringBuffer sb = new StringBuffer("Wipro Technologies");  
        String result1 = st.substring(6,12);  
        String result2 = sb.substring(6);  
        String result3 = sb.substring(6,12);  
        System.out.println("Substring of String st : "+result1);  
        System.out.println("Substring of StringBuffer sb (with single argument): "+result2);  
        System.out.println("Substring of StringBuffer sb (with two arguments) : "+result3);  
    }  
}
```

Substring of String st : Techno
Substring of StringBuffer sb (with single argument): Technologies
Substring of StringBuffer sb (with two arguments) : Techno

Summary

In this module, we were able to:

- Implement String and StringBuffer class methods



Thank You