Guided LAB 303.3.1 - Java String Methods

Objective

In this lab, you will explore and demonstrate the built-in methods in the String class. These methods help you manipulate the String data type.

By the end of this lab learners will be able to:

- Describe the Strings methods.
- Use Java Strings Methods.

Instructions

1. length() method

The length() method returns the length of the string, or it returns the count of the total number of characters present in the string.

Example:

```
Public class lengthDemo {
    public static void main(String[] args) {
        String str1 = "Java";
        String str2 = "";

        System.out.println(str1.length());  // 4
        System.out.println(str2.length());  // 0
        System.out.println("Java".length());  // 4
        System.out.println("Java\n".length()); // 5
        System.out.println("Learn Java".length()); // 10
    }
}
```

Output

4
0
4
5
10

2. IsEmpty() method:

This method checks whether the String contains anything or not. If the Java String is empty, it returns true or false.

Example:

```
public class IsEmptyExample{
public static void main(String args[]){
String s1="";
String s2="hello";
System.out.println(s1.isEmpty());      // true
System.out.println(s2.isEmpty());      // false
}}
```

Output:

True
False

3. Trim() method:

The Java string **trim()** method removes the leading and trailing spaces. It checks the Unicode value of the space character (‘\u0020’) before and after the string. If it exists, then remove the spaces and return the omitted string.

Example:

```
public class StringTrimExample{
public static void main(String args[]){
```

```
String s1="  hello   ";
System.out.println(s1+"how are you");        // without trim()
System.out.println(s1.trim()+"how are you"); // with trim()
 }
}
```

In the above code, the first print statement will print ***"hello how are you"*** while the second statement will print ***"hello how are you"*** using the trim() function.

4. toLowerCase() method:

The **toLowerCase()** method converts all the characters of the String to lowercase. For example:

```
public class StringLowerExample{
 public static void main(String args[]){
    String s1="HELLO HOW Are You?";
    String s1lower=s1.toLowerCase();
    System.out.println(s1lower);}
}
```

The above code will return **"hello how are you."**

5. Java String toUpper() method:

The **toUpperCase()** method converts all of the String characters to uppercase. For example:

```
public class StringUpperExample{
  public static void main(String args[]){
    String s1="hello how are you";
    String s1upper=s1.toUpperCase();
    System.out.println(s1upper);
}}
```

The above code will return **"HELLO HOW ARE YOU."**

6. concat() method

You can also use the **+** operator to concatenate two or more strings. But the Java String class provides the **concat()** method. This method combines a specific string at the end of another string, and ultimately returns a combined string. For example:

```
public class concatDemo {
   public static void main(String[] args) {
      //------By using concat method----
       String str1 = "Learn ";
       String str2 = "Java";
       // concatenate str1 and str2
       System.out.println(str1.concat(str2)); // "Learn Java"

       // concatenate str2 and str11
       System.out.println(str2.concat(str1)); // "JavaLearn "
     //--- By using + operator---
       String s3 =  "hello";
       String s4 = "Learners";
     //  String s5 = s3.concat(s4); or
          String s5 = s3 + s4;
     //both of the above statement will give you the same result

      // Three strings are concatenated
       String message = "Welcome " + "to " + "Java";

     // String Chapter is concatenated with number 2
       String s = "Chapter" + 2; // s becomes Chapter2

   // String Supplement is concatenated with character B
       String s1 = "Supplement" + 'B'; // s1 becomes SupplementB
   }
}
```

7. split() method

The split() method divides the string at the specified regex and returns an array of substrings. The syntax of the string split() method is:

string.split(String regex, int limit)

- regex - the string is divided at this regex (can be strings)
- limit (optional) - controls the number of resulting substrings

If the limit parameter is not passed, split() returns all possible substrings.

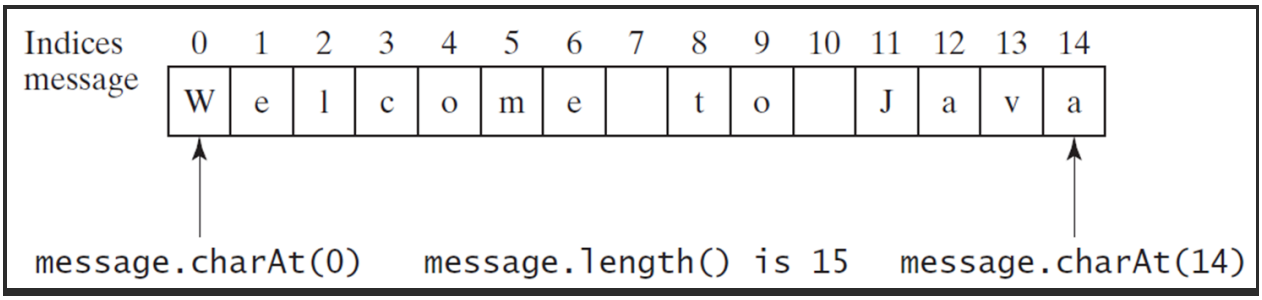
```
// importing Arrays to convert array to string
// used for printing arrays
import java.util.Arrays;

public class Main {
  public static void main(String[] args) {
    String vowels = "a::b::c::d:e";

    // splitting the string at "::"
    // storing the result in an array of strings
    String[] result = vowels.split("::");

    // converting array to string and printing it
    System.out.println("result = " + Arrays.toString(result));
  }
}
```

Output

result = [a, b, c, d:e]

Here, we split the string at :: Since the limit parameter is not passed, the returned array contains all the substrings.

8. charAt() method: *Getting Characters From a String*



Example:

```
public class ConcatDemo {
   public static void main(String[] args) {
   String message = "Welcome to Java";
   System.out.println("The first character in the message is
" + message.charAt(0));
 }
}
```

9. compareTo() method:

The Java String compareTo() method compares the given string with the current string. It is a method of 'Comparable' interface that is implemented by the String class. It either returns a positive number, a negative number, or 0. For example:

```
public class CompareToExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="hemlo";
String s4="flag";
System.out.println(s1.compareTo(s2)); // 0 because both are equal
```

```
System.out.println(s1.compareTo(s3)); //-1 because "l" is only one time lower than "m"
System.out.println(s1.compareTo(s4)); // 2 because "h" is 2 times greater than "f"
}}
```

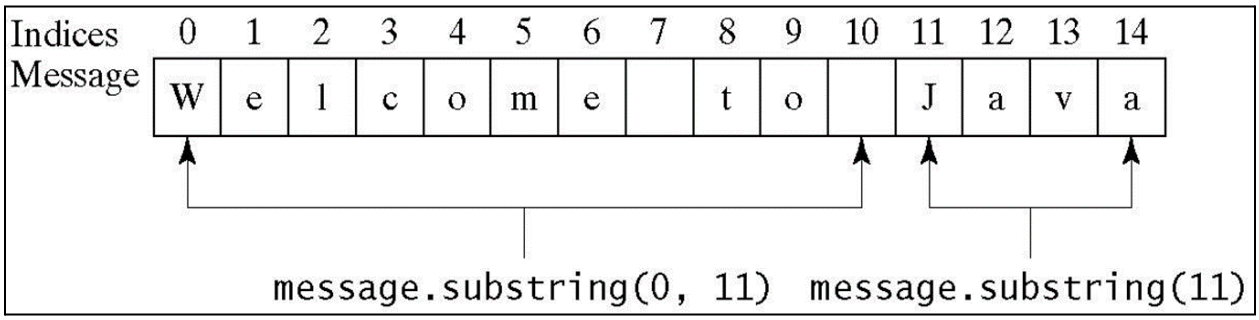
The above program shows the comparison between the various Strings. Notice:

```
if s1 > s2, it returns a positive number.
if s1 < s2, it returns a negative number.
if s1 == s2, it returns 0.
```

10. Substrings() method:

The substring() method extracts a substring from the string and returns it. The syntax of the substring() method is

```
stringObj.substring(int startIndex, int endIndex)
```



Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`, as shown in Figure 9.6. Note that the character at `endIndex` is not part of the substring.

Example:

```
public class substringDemo{
  public static void main(String[] args) {
    String str1 = "java is fun";

    // extract substring from index 0 to 3
    System.out.println(str1.substring(0, 4));

  }
}
```

// Output:

java

11. indexOf() method:

The **indexOf()** method returns the index of the first occurrence of the specified character/substring within the string.

Example:

```
public class findString {
   public static void main(String[] args) {
       String str1 = "Java is fun";
       int result;

       // getting index of character 's'
       result = str1.indexOf('s');

       System.out.println(result); // 6

       // getting index of character 'J'
       result = str1.lastIndexOf('J');
       System.out.println(result); // 0
```

```
        // the last occurrence of 'a' is returned
        result = str1.lastIndexOf('a');
        System.out.println(result); // 3

        // character not in the string
        result = str1.lastIndexOf('j');
        System.out.println(result); // -1

        // getting the last occurrence of "ava"
        result = str1.lastIndexOf("ava");
        System.out.println(result); // 1

        // substring not in the string
        result = str1.lastIndexOf("java");
        System.out.println(result); // -1
    }
}
```

Output:
6
0
3
-1
1
-1

12. contains() method:

The **contains()** method checks whether the specified string (sequence of characters) is present in the string or not.

```
Public class ContainExample {
  public static void main(String[] args) {
    String str1 = "Learn Java";
```

```
    Boolean result;

    // check if str1 contains "Java"
    result = str1.contains("Java");
    System.out.println(result);  // true

    // check if str1 contains "Python"
    result = str1.contains("Python");
    System.out.println(result);  // false

    // check if str1 contains ""
    result = str1.contains("");

    System.out.println(result);  // true
  }
}
```

Output:
true
false
true

13. replace() method

The replace() method replaces each matching occurrence of the old character/text in the string with the new character/text.

The syntax of the replace() method is either:

```
stringobj.replace(char oldChar, char newChar)
```

or

```
stringobj.replace(CharSequence oldText, CharSequence newText)
```

Here, stringobj is an object of the String class.

```
Public class ReplaceDemoMain {
  public static void main(String[] args) {
    String str1 = "abc cba";

    // all occurrences of 'a' is replaced with 'z'
    System.out.println(str1.replace('a', 'z'));

    // all occurences of 'L' is replaced with 'J'
    System.out.println("Lava".replace('L', 'J'));
    // character not in the string
    System.out.println("Hello".replace('4', 'J'));
// all occurrences of "C++" is replaced with "Java"
    System.out.println(str1.replace("C++", "Java"));

    // all occurences of "aa" is replaced with "zz"
    System.out.println("aa bb aa zz".replace("aa", "zz"));

    // substring not in the string
    System.out.println("Java".replace("C++", "C"));
  }
}
```

Output

```
zbc cbz
Java
Hello
abc cba
zz bb zz zz
Java
```

Note: If the substring to be replaced is not in the string, replace() returns the original string.

14. Java String replaceAll()

- The ***replaceAll()*** method replaces each substring that matches the regex of the string with the specified text.
- The syntax of the `replaceAll()` method is:

```
string.replaceAll(String regex, String replacement)
```

- **regex** - a regex (can be a typical string) that is to be replaced
- **replacement** - matching substrings are replaced with this string

Create a class named **`DemoReplaceAll`** and add the code below to it.

```
public  class DemoReplaceAll {
  public static void main(String[] args) {
    String str1 = "Java123is456fun";

    // regex for sequence of digits
    String regex = "\\d+";

    // replace all occurrences of numeric
    // digits by a space
    System.out.println(str1.replaceAll(regex, " "));
  }
}
```

Output: Java is fun

In the above example, **`"\\d+"`** is a regular expression that matches one or more digits.

To learn more about regexes, **visit *Java Regex*.**

15. Java String compares

We can compare String in Java based on content and reference.

It is used in **authentication** (by equals() method), **sorting** (by compareTo() method), and **reference matching** (by == operator) etc.

There are three ways to compare String in Java:

- By Using equals() Method.
- By Using **==** Operator.
- By compareTo() Method.

16. By Using equals() Method

The String class's equals() method compares the original content of the string. It compares the values of strings for equality.

Create a class named **TestStringComparison** and add the code below to it.

```
public class TestStringComparison {
   public static void main(String args[]){
       String s1="PerScholas";
       String s2="PerScholas";
       String s3=new String("PerScholas");
       String s4="Teksystem";
       System.out.println(s1.equals(s2));//true
       System.out.println(s1.equals(s3));//true
       System.out.println(s1.equals(s4));//false
   }
}
```

Output
true
true
false

In the above code, two strings are compared to using the **equals()** method of String class. The result is printed as Boolean values, **true** or **false**.

17. By Using == operator

The == operator compares references, not values.

Create a class named Teststringcomparison2.java and write the code below to it.

```
public class Teststringcomparison2 {
   public static void main(String args[]){
        String s1="Perscholas";
        String s2="Perscholas";
        String s3=new String("Perscholas");
        System.out.println(s1==s2);//true (because both refer to same
instance)
        System.out.println(s1==s3);//false(because s3 refers to instance
created in nonpool)
   }

}
```

Output:

true
false

The above code demonstrates the use of == operator used for comparing two String objects.

18. By Using compareTo() method,

The String class's compareTo() method compares values lexicographically. It returns an integer value that describes if the first string is less than, equal to, or greater than the second string.

// return 0 if this string is the same as another;

// <0 if lexicographically less than another; or >0

Suppose s1 and s2 are two String objects. If:
s1 == s2 : The method returns 0.
s1 > s2 : The method returns a positive value.
s1 < s2 : The method returns a negative value.

Create a class named Teststringcomparison4.java and write the code below to it.

```
public class Teststringcomparison3 {
   public static void main(String args[]){
       String s1="Perscholas";
       String s2="Perscholas";
       String s3="Perschola";
       String s4="PerscholasX";
       System.out.println(s1.compareTo(s2)); //0
       System.out.println(s1.compareTo(s3)); // 1(because s1>s3)
       System.out.println(s1.compareTo(s4)); // -1(because s1<s4 )
   }
}
```

Output;

0

1

-1

Submission Instructions:

Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: GLAB - 303.3.1 - Java String methods **Points: 100** **Assignment Group: Module 303: Java SE Review (Not Graded)** **Display Grade As: Non-graded (This assignment does not count toward the final grade.). Complete/Incomplete** **Do not count this assignment towards the final grade: Checked** **Submission Types: File Uploads** **Everything else is the default.**