Working with Strings

Estimated time needed: 30 minutes

In this lab, you will learn how to work with Strings in Java.

You are currently viewing this lab in a Cloud based Integrated Development Environment (Cloud IDE). It is a fully-online integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Create a String object in more than one way
- Learn how to calculate the length of the String
- Access each character of the String
- Concatenate Strings
- Compare Strings
- Use other methods in the String class as required

Create String objects and calculate length

String is just an array of characters. In Java, String is one of the most basic and important object and it is a part of the `java.lang` package which is imported by default in all the programs. All the class in Java, implicitly have a toString method inherited from the `Object` class which gives the String representation of an Object.

1. Create a project directory by running the following command.

   ```
   mkdir my_string_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_string_proj/src
   mkdir -p my_string_proj/classes
   mkdir -p my_string_proj/test
   cd my_string_proj
   ```

3. Now create a file named `StringOps.java` inside the src directory.

```
touch /home/project/my_string_proj/src/StringOps.java
```

4. Click the following button to open the file for editing.

Open **StringOps.java** in IDE

5. Read each statement in the following program and understand how a `String` object can be created and how to measure the length of the object.

Paste the following content in `StringOps.java`.

```
public class StringOps {
    public static void main(String s[]) {
        String s1 = "Hello World";
        System.out.println(s1);
        String s2 = new String("Hello World");
        System.out.println(s2);
    }
}
```

`String s1 = "Hello World";` - create a String literal
`String s2 = "Hello World";` - creates a String Object

See if you are able to observe any difference in the two when you compile and run.

6. Compile the java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/StringOps.java
```

7. Set the `CLASSPATH` variable.

   ```
   export CLASSPATH=$CLASSPATH:/home/project/my_string_proj/classes
   ```

8. Compile the java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/StringOps.java
   ```

9. Run the program and observe the output.

   ```
   java StringOps
   ```

You will see the output as below:

```
Hello World
Hello World
```

As observed from the output, `String s1 = "Hello World";` and `String s2 = new String("Hello World");` may seem identical, but there's a subtle difference:

s1 is a String literal and s2 is a String object.

- String Literal (s1): When you write `String s1 = "Hello World";`, Java creates a String literal in the String constant pool. This pool is a memory area where Java stores String literals. If the same string literal is used again in the code, Java reuses the existing instance from the pool, instead of allocating memory for another one.
- String Object (s2): When you write `String s2 = new String("Hello World");`, Java creates a new String object on the heap memory. This object is not stored in the String constant pool. Every time you use `new String("Hello World")` a new object is created in the memory, though the content are the same.

10. Verify the difference between String literal and String object. Replace the code in StringOps.java with the following code.

```
public class StringOps {
    public static void main(String s[]) {
        String s1 = "Hello World";
        System.out.println(s1);
        String s2 = new String("Hello World");
        System.out.println(s2);
        String s3 = "Hello World";
        System.out.println("s1 and s2 comparison "+ (s1 == s2));
        System.out.println("s2 and s3 comparison "+ (s2 == s3));
        System.out.println("s1 and s3 comparison "+ (s1 == s3));
    }
}
```

`s1` and `s3` are String literals. The both refer to the same String in memory. `s2` is a String object. `==` operator returns `true` only if both operands point to the same object in the memory heap. Compile and run to observer the output.

11. Compile the java program.

```
javac -d classes src/StringOps.java
```

12. Run the program and observe the output.

```
java StringOps
```

You will see the output as below:

```
Hello World
Hello World
s1 and s2 comparison false
s2 and s3 comparison false
s1 and s3 comparison true
```

Find Length of the string and access characters

String is an array of characters. The length of the string is the same as the length of the array of characters that make up a String. String class provides a method to get the length of the String.

1. Click the following button to open `StringOps.java` for editing.

Open **StringOps.java** in IDE

2. Add the following code within the `StringOps` class in place of the existing code.

```
public class StringOps {
    public static void main(String s[]) {
        String s1 = "The quick brown fox jumped over the lazy dog";
        System.out.println(s1.length());
        char[] strAsArr = s1.toCharArray();
        System.out.println(strAsArr.length);
        System.out.println(strAsArr);
        System.out.println("The first char of the string is " + strAsArr[0]);
        System.out.println("The last char of the string is " + strAsArr[strAsArr.length-1]);
        System.out.println("The index of T is " + s1.indexOf('T'));
        System.out.println("The index of g is " + s1.indexOf('g'));
    }
}
```

`char[] strAsArr = s1.toCharArray();` - convert the String into a character array and stores in variable strAsArr.

The print statements prints the length of the String as is, the length of the array, the String as an array, the first character at index 0 and the last character at index (length-1).

`s1.indexOf('T')` returns the first index of `T` in the String stored in s1. If there is no such character, it returns `-1` indicating that there is no such character in the String.

Always remember the index positions start from 0. So if the length of the String is say `5`, the last index position is `4` (5-1).

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/StringOps.java
```

4. Run the program and observe the output.

```
java StringOps
```

You will see the output as the following:

```
44
44
The quick brown fox jumped over the lazy dog
The first char of the string is T
The last char of the string is g
The index of T is 0
The index of g is 43
```

String Comparison

Comparing two Strings with `==` will check if two Strings are referring to the same object and will not verify if the content of the objects is the same. Now explore ways to compare two Strings.

1. Click the following button to open the file `StringOps.java` for editing.

Open **StringOps.java** in IDE

2. Read each statement in the following program and understand how the String comparison is made. Paste the following content in `StringOps.java`.

```
public class StringOps {
    public static void main(String s[]) {
        String s1 = "Washington";
        String s2 = new String("Washington");
        String s3 = "WASHINGTON";
        System.out.println("Equality check s1 and s2 - "+s1.equals(s2));
        System.out.println("Equality check s1 and s3 - "+s1.equals(s3));
        System.out.println("Equality check s1 and s3 ignoring case - "+s1.equalsIgnoreCase(s3));
        System.out.println("s1 in lowercase - "+s1.toLowerCase());
        System.out.println("s3 in lowercase - "+s3.toLowerCase());
        System.out.println("s1 and s3 lowercase equality check - " +
                            s1.toLowerCase().equals((s3.toLowerCase())));
        System.out.println("s1 in uppercase - "+s1.toUpperCase());
        System.out.println("s3 in uppercase - "+s3.toUpperCase());
        String s4 = "50F1A";
        System.out.println("s4 in lowercase - "+s4.toLowerCase());
        String regexStr = "^W.*";
        System.out.println("s1 matches regex ^W.* - "+s1.matches(regexStr));
        System.out.println("s3 matches regex ^W.* - "+s3.matches(regexStr));
        String s5 = "     WASHINGTON          ";
        System.out.println("Equality check s3 and s5 - "+s3.equals(s5));
        s5 = s5.strip();
        System.out.println("Equality check after stripping s3 and s5 - "+s3.equals(s5));
    }
}
```

`.equals` - Returns true if the content of two Strings are exactly the same including the leading and trailing space.
`.equalsIgnoreCase` - Returns true ignoring the case of the two Strings. But the characters have to be in the same positions in both Strings and should not have any leading and trailing spaces.

`.toLowerCase()` - Returns a new String object with all the characters in the String object on which it is invoked, as lowercase. If there are any non-aplhabetic characters, those are kept as is.

`.toUpperCase()` - Returns a new String object with all the characters in the String object on which it is invoked, as uppercase. If there are any non-aplhabetic characters, those are kept as is.

`.strip()` - Removes the leading and trailing white spaces. This is extremely useful when dealing with data.

`.matches()` - Matches the String as regular expression. Regex or regular expression matches a pattern. `^W.*` - matches all Strings that start with uppercase W followed by any characeters or any length.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/StringOps.java
```

4. Now, run the Java program.

```
java StringOps
```

Your output would appear as below:

```
Equality check s1 and s2 - true
Equality check s1 and s3 - false
Equality check s1 and s3 ignoring case - true
s1 in lowercase - washington
s3 in lowercase - washington
s1 and s3 lowercase equality check - true
s1 in uppercase - WASHINGTON
s3 in uppercase - WASHINGTON
s4 in lowercase - 50f1a
s1 matches regex ^W.* - true
s3 matches regex ^W.* - true
Equality check s3 and s5 - false
Equality check after stripping s3 and s5 - true
```

Concatenation and substringing

Concatenation is putting two Strings together. Substring is extracting a part of a String from a given String.

1. Click the following button to open the file `StringOps.java` for editing.

Open **StringOps.java** in IDE

2. Read each statement in the following program and understand how the String concatenation and substringing is done. Paste the following content in `StringOps.java`.

```
public class StringOps {
    public static void main(String s[]) {
        String s1 = "Washington";
        String s2 = " DC";
        s1 = s1.concat(s2);
        System.out.println("s1 " + s1);
        s1 = s1.substring(0,10);
        System.out.println("s1 " + s1);
        System.out.println("s1.substring(7,10) " + s1.substring(7,10));
        System.out.println("s1.substring(7) " + s1.substring(7));
        }
    }
```

`.concat` - Java creates a new String object and copies the characters from the original Strings to the new object.
`.substring` - Java creates a new String object and copies the characters from the index position passed as first parameter to the index position passed as second parameter from the original Strings to the new object.

If there is not second parameter passed, it takes from the String in first index position to the end of the String.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/StringOps.java
   ```

4. Now run the java program.

   ```
   java StringOps
   ```

Your output would appear as below:

```
s1 Washington DC
s1 Washington
s1.substring(7,10) ton
s1.substring(7) ton
```

Practice Exercise

1. Create a two String literals "Maple Tree" and "Maple Tree". Check their equality using `==`.
2. Create a new String object "Maple Tree". Check their equality with the String literal using `==` and then check using `.equals` method.
3. Use `substring` to separate out "Maple" and "Tree" into two different objects.
4. Use `concat` to string the object back together as "Maple Tree".
5. Use toLowerCase to convert all the characters to lowercase.
6. Use toUpperCase to convert all the characters to lowercase.

   Hint: Use Scanner and Integer.parseInt to get the index number

▶ Click here for the sample code

Conclusion

In this lab, you learned how to create Strings and also invoke methods on the String objects.

Author(s)

Lavanya