Java SE 8 Programming Language

Lab Guides

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 06/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	01/Oct/2018	Add the new labs	Create new	DieuNT1	VinhNV
2	01/Jun/2019	Update template	Fsoft template	DieuNT1	VinhNV

Contents

CODE:	JPL.M.L201
TYPE:	MEDIUM
LOC:	
DURATION:	60 MINUTES

Unit 7: Manipulate and format data in a program

Knowledge Summary

When working with text data, Java provides you with three classes including String, StringBuffer and StringBuilder. When working with big data, you should use StringBuffer or StringBuilder to optimize the efficiency. Basically, three classes have many similarities.

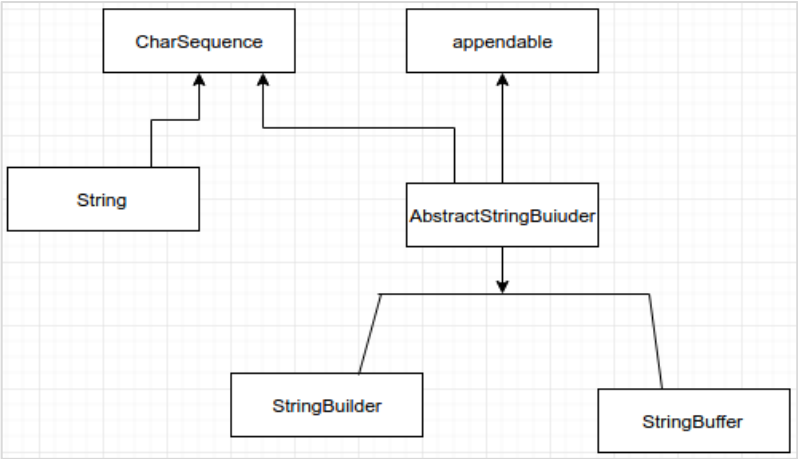
- String is immutable. It does not allow the existence of subclass.
- StringBuffer, StringBuilder are mutable.
- StringBuilder and StringBuffer are alike, except for the using situation related to Multi Thread.

To handle the text with many threads, you should use StringBuffer in order to prevent the conflict among threads.

To handle the text with one thread, you should use StringBuilder.

As for the handling speed, StringBuilder is the best, following is StringBuffer and String is the worst.

Hierarchical Inheritance in String



Factor/Class	String	StringBuilder	StringBuffer
Mutability	Immutable	Mutable	Mutable
Thread Safety	Not thread safe	Thread safe	Not thread safe
Performance	Very high	Moderate	Very high

Lab Guide 1: Manipulate data with String

Objectives:

- This lab guide helps trainees know how to perform some operations with String in Java.

Problem Descriptions:

Create a Java project named **JPL.M.L201** in Eclipse.

Create package **fa.training.stringdemo** that contains 3 classes:

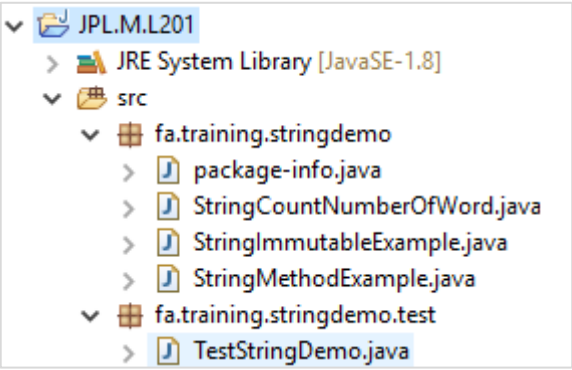
- StringImmutableExample class
- StringMethodExample class
- StringCountNumberOfWord class

Create package **fa.training.stringdemo.test** that contains a class included the main method to run the program

- TestStringDemo class

Guidelines:

Step 1: Create project struture like this:



Step 2: Create **StringImmutableExample** class

```
1. package fa.training.stringdemo;
2.
3. /**
4.  * String immutableness demonstration
5.  *
6.  * @author hoabt2
7.  *
8.  */
9. public class StringImmutableExample {
10.
11.
12. /**
13.  * Show an demonstration about immutable characteristic of String in Java.
14.  *
15.  */
16. public void demonstrateStringImmutable() {
17.         System.out.println("demonstrateStringImmutable() !!!");
18.         // "Java" String created in pool and reference assigned to s1
19.         String s1 = "Java";
20.
21.         // s2 is also having the same reference to "Java" in the pool
22.         String s2 = s1;
23.
24.
```

```
25.         // proof that s1 and s2 have same reference
26.         System.out.println(s1 == s2);
27.
28.         s1 = "Python";
29.         // s1 value got changed above, so how String is immutable?
30.
31.         // well, in above case a new String "Python" got created in the pool
32.         // s1 is now referring to the new String in the pool
33.         // BUT, the original String "Java" is still unchanged and
34.         // remains in the pool
35.         // s2 is still referring to the original String "Java" in the pool
36.
37.         // proof that s1 and s2 have different reference
38.         System.out.println(s1 == s2);
39.
40.         System.out.println(s2);
41.         // prints "Java" supporting the fact that
42.         // original String value is unchanged, hence String is immutable
43. }
44. }
45.
```

Step 3: Create **StringMethodExample** class

```
1.  package fa.training.stringdemo;
2.
3.  /**
4.   * @author hoabt2
5.   *
6.   */
7.  public class StringMethodExample {
8.
9.          /**
10.          * Examples of using some basic methods of String
11.          */
12.         public void demonstrateStringMethod() {
13.                 String targetString = "Java is fun to learn";
14.                 String s1 = "JAVA";
15.                 String s2 = "Java";
16.                 String s3 = "  Hello Java  ";
17.
18.                 System.out.println("demonstrateStringMethod() !!!");
19.
20.                 System.out.println("Char at index 2(third position): " +
21.                                         targetString.charAt(2));
22.                 System.out.println("After Concat: " +
23.                                     targetString.concat("-Enjoy-"));
24.                 System.out.println("Checking equals ignoring case: " +
25.                                         s2.equalsIgnoreCase(s1));
26.                 System.out.println("Checking equals with case: " +
27.                                         s2.equals(s1));
28.                 System.out.println("Checking Length: " +
29.                                         targetString.length());
30.                 System.out.println("Replace function: " +
31.                                         targetString.replace("fun", "easy"));
32.                 System.out.println("SubString of targetString: " +
33.                                         targetString.substring(8));
34.                 System.out.println("SubString of targetString: " +
35.                                         targetString.substring(8, 12));
36.                 System.out.println("Converting to lower case: " +
37.                                         targetString.toLowerCase());
```

```
38.             System.out.println("Converting to upper case: " +
39.                                         targetString.toUpperCase());
40.             System.out.println("Triming string: " + s3.trim());
41.             System.out.println("searching s1 in targetString: " +
42.                                         targetString.contains(s1));
43.             System.out.println("searching s2 in targetString: " +
44.                                         targetString.contains(s2));
45.
46.             char[] charArray = s2.toCharArray();
47.             System.out.println("Size of char array: " + charArray.length);
48.             System.out.println("Printing last element of array: " +
49.                                                         charArray[3]);
50.
51.         }
52. }
```

Step 4: Create **StringCountNumberOfWord** class

```
1.  package fa.training.stringdemo;
2.
3.  /**
4.   * Demonstrate to count number of words in a String.
5.   *
6.   * @author hoabt2
7.   *
8.   */
9.  public class StringCountNumberOfWord {
10.
11.
12. public void demonstrateCountWord() {
13.         System.out.println("demonstrateCountWord() !!!");
14.
15.         countNumberOfWords("My name is Admin");
16.         countNumberOfWords("I love Java Programming");
17.         countNumberOfWords("This is not properly formatted data");
18. }
19.
20. /**
21.  * Count number of words in a String.
22.  *
23.  * @param str the input String
24.  */
25. private static void countNumberOfWords(String str) {
26.         String trimmedLine = str.trim();
27.         int count = trimmedLine.isEmpty() ? 0 :
28.                                 trimmedLine.split("\\s+").length;
29.         System.out.println(count);
30. }
31.
32. }
33.
```

Step 5: Create **TestStringDemo** class

```
1.  package fa.training.stringdemo.test;
2.
3.  import fa.training.stringdemo.StringCountNumberOfWord;
4.  import fa.training.stringdemo.StringImmutableExample;
5.  import fa.training.stringdemo.StringMethodExample;
6.
```

```
7.  /**
8.   * @author hoabt2
9.   *
10.  */
11. public class TestStringDemo {
12.
13. /**
14.  * @param args
15.  */
16. public static void main(String[] args) {
17.         StringImmutableExample stringImmutable = new StringImmutableExample();
18.         StringMethodExample stringMethod = new StringMethodExample();
19.         StringCountNumberOfWord countWord = new StringCountNumberOfWord();
20.         stringImmutable.demonstrateStringImmutable();
21.         stringMethod.demonstrateStringMethod();
22.         countWord.demonstrateCountWord();
23. }
24.
25. }
26.
```

Step 6:: Run TestStringDemo to see the output

You can call coresponding methods separatedly in order to see the result clearly.

```
Problems  @ Javadoc  Declaration  Console
<terminated> TestStringDemo [Java Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (Jun 14, 2019, 9:40:54 AM)
demonstrateStringImmutable() !!!
true
false
Java
demonstrateStringMethod() !!!
Char at index 2(third position): v
After Concat: Java is fun to learn-Enjoy-
Checking equals ignoring case: true
Checking equals with case: false
Checking Length: 20
Replace function: Java is easy to learn
SubString of targetString: fun to learn
SubString of targetString: fun
Converting to lower case: java is fun to learn
Converting to upper case: JAVA IS FUN TO LEARN
Triming string: Hello Java
searching s1 in targetString: false
searching s2 in targetString: true
Size of char array: 4
Printing last element of array: a
demonstrateCountWord() !!!
4
4
6
```

Lab Guide 2: Manipulate data with StringBuilder, StringBuffer

Objective:

- This lab guide helps trainees know how to use StringBuilder, StringBuffer.

Problem Description:

Use **JPL.M.L201** project created in Lab Guide 1.

Create package **fa.training.stringbuilderdemo** that contains:

- StringBuilderExample class

Create package **fa.training.stringbufferdemo** that conttains:

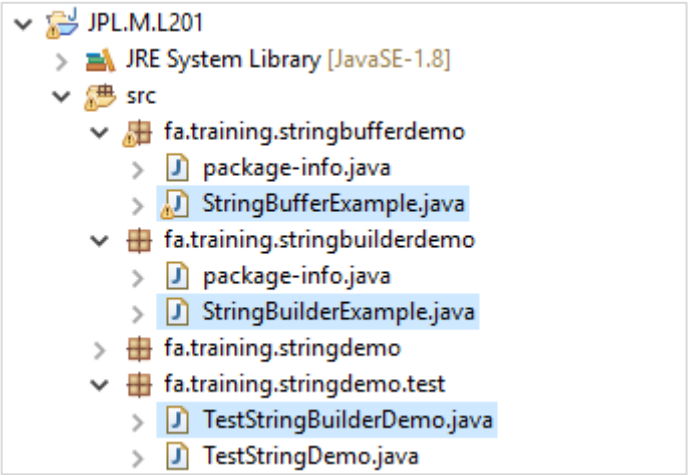
- StringBufferExample class

Create package **fa.training.stringdemo.test** that contains a class included the main method to run the program

- TestStringBuilderDemo class

Guidelines:

Step 1: Update the project structure:



Step 2: Create **StringBuilderExample** class

```
1. package fa.training.stringbuilderdemo;
2.
3. /**
4.  * Examples of StringBuilder
5.  *
6.  * @author hoabt2
7.  *
8.  */
9. public class StringBuilderExample {
10.
11./**
12. * Demonstrate how to use StringBuilder
13. *
14. */
15.public void demonstrateStringBuilder() {
16.       System.out.println("demonstrateStringBuilder() !!!");
17.       StringBuilder sb1 = new StringBuilder("Hello Java World");
18.       sb1.delete(4, 8);
19.       System.out.println("Delete method demo: " + sb1);
20.       StringBuilder sb2 = new StringBuilder("Hello Java World");
```

```
21.         sb2.insert(4, "abc");
22.         System.out.println("Inser Operation: " + sb2);
23.         StringBuilder sb3 = new StringBuilder("fsoft.fpt.com");
24.         sb3.replace(1, 4, "Amit");
25.         System.out.println("Replace Operation: " + sb3);
26.         StringBuilder sb4 = new StringBuilder("ABCDE");
27.         System.out.println("Reverse of ABCDE: " + sb4.reverse());
28.         StringBuilder sb5 = new StringBuilder("ABCDEF");
29.         sb5.setCharAt(3, 'x');
30.         System.out.println("Replacing char at index 3: " + sb5);
31. }
32.
33. }
34.
```

Step 3: Create **StringBufferExample** class

```
1. package fa.training.stringbufferdemo;
2.
3. /**'
4.  * Example of using StringBuffer
5.  *
6.  * @author hoabt2
7.  *
8.  */
9. public class StringBufferExample {
10.
11. /**
12.  * Demonstrate how to use StringBuffer
13.  *
14.  */
15. public void demonstrateStringBuffer() {
16.         System.out.println("demonstrateStringBuffer() !!!");
17.
18.         StringBuffer buffer = new StringBuffer();
19.
20.          // Append the string representation of the argument to
21.          // the end of the buffer.
22.          // In this example we use a string,
23.          // but the method also accepts int, float,
24.          // double, boolean, char (or char[]), as well as objects.
25.          buffer.append("Hello World!");
26.          System.out.println(buffer.toString());
27.
28.          // Delete the specified substring by providing the start and the end
29.          // of the sequence.
30.          buffer.delete(5, 11);
31.          System.out.println(buffer.toString());
32.
33.          // Delete just one char by providing its position.
34.          buffer.deleteCharAt(5);
35.          System.out.println(buffer.toString());
36.
37.          // Insert a string in a specified place inside the buffer.
38.          buffer.insert(0, "World ");
39.          System.out.println(buffer.toString());
40.
41.          // Get the index that the specified substring starts at.
42.          System.out.println("Index of Hello: " + buffer.indexOf("Hello"));
43.          System.out.println(); // Empty line
44.
45.
46.          // You can also instantiate a new StringBuffer and provide
```

```
47.         // the initial String in the constructor.
48.         StringBuffer newBuffer = new StringBuffer("
49.                                 This is a Hello World string. Hello!");
50.
51.         // You can use lastIndexOf(String) to get the last time
52.         // that a specified
53.         // substring appears in the StringBuffer.
54.         System.out.println("Index of Hello: " + newBuffer.indexOf("Hello"));
55.         System.out.println("Last index of Hello: " +
56.                                     newBuffer.lastIndexOf("Hello"));
57.
58.         // You can also replace a specific sub-sequence of
59.         // the StringBuffer with another string.
60.         // The size does not need to be the same, as shown here.
61.         newBuffer.replace(0, 4, "That here");
62.         System.out.println(newBuffer.toString());
63.
64.         // You can replace a single char using this method here. We want to
65.         // replace the last character of the string,
66.         // so instead of counting the length,
67.         // we will use the provided length() method,
68.        // and replace the char in the last index.
69.         newBuffer.setCharAt(newBuffer.length() - 1, '?');
70.         System.out.println(newBuffer.toString());
71.
72.         // You can reverse the StringBuffer as well!
73.         newBuffer.reverse();
74.         System.out.println(newBuffer.toString());
75.         comparePerformance();
76.
77.}
78.
79./**
80. * Compare performance between String and StringBuffer
81. *
82. */
83.   private static void comparePerformance() {
84.         long startTime;
85.         String str = "";
86.         StringBuffer buffer = new StringBuffer();
87.
88.         // Using String
89.         startTime = System.currentTimeMillis();
90.         for (int i = 0; i < 10000; i++) {
91.             str += "extra";
92.         }
93.         System.out.println("Time using String: "
94.                 + (System.currentTimeMillis() - startTime) + " ms.");
95.
96.         // Using StringBuffer
97.         startTime = System.currentTimeMillis();
98.         for (int i = 0; i < 10000; i++) {
99.             buffer.append("extra");
100.            }
101.            System.out.println("Time using StringBuffer: "
102.                    + (System.currentTimeMillis() - startTime) + " ms.");
103.         }
104.     }
105.
```

Step 4: Create **TestStringBuilderDemo** class

```
1. package fa.training.stringdemo.test;
2.
3. import fa.training.stringbufferdemo.StringBufferExample;
4. import fa.training.stringbuilderdemo.StringBuilderExample;
5.
6. /**
7.  * @author hoabt2
8.  *
9.  */
10.public class TestStringBuilderDemo {
11.
12./**
13. * @param args
14. */
15.public static void main(String[] args) {
16.       StringBuilderExample stringBuilder = new StringBuilderExample();
17.       StringBufferExample stringBuffer = new StringBufferExample();
18.       stringBuilder.demonstrateStringBuilder();
19.       stringBuffer.demonstrateStringBuffer();
20.}
21.
22.}
```

Step 5: Run TestStringBuilderDemo to see the result

You can call coresponding methods separatedly in order to see the result clearly.

Result:

```
Problems  @ Javadoc  Declaration  Console
<terminated> TestStringBuilderDemo [Java Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (Jun 14, 2019, 10:16:36 AM)
demonstrateStringBuffer() !!!
Hello World!
Hello!
Hello
World Hello
Index of Hello: 6

Index of Hello: 10
Last index of Hello: 30
That here is a Hello World string. Hello!
That here is a Hello World string. Hello?
?olleH .gnirts dlroW olleH a si ereh tahT
Time using String: 249 ms.
Time using StringBuffer: 4 ms.
```

-- THE END --