Desk Check: StringPractice.java

The source code for the StringPractice.java program is shown below. You know the drill by now; look for the big picture, then the details. Highlighted code shows the String methods.

```
< 1>   public class StringPractice
< 2>   {
< 3>         public static void main(String[ ] args)
< 4>         {
< 5>            //determine the length of the String object called oldString
< 6>            String oldString = "Four score and seven years ago";
< 7>            System.out.println("Old string: " + oldString);
< 8>            int stringLength = oldString.length();
< 9>            System.out.println("Number of characters: " + stringLength);
<10>            System.out.println();
<11>
<12>            //replace characters within the String object
<13>            String replaceCharacters = oldString.replace("Four", "4");
<14>            replaceCharacters = replaceCharacters.replace("seven", "7");
<15>            System.out.println("Replacement characters: " + replaceCharacters);
<16>            System.out.println();
<17>
<18>            //split the String object and reassemble in reverse
<19>            int halfwayPoint = stringLength /2;
<20>            String firstHalf = oldString.substring(0, halfwayPoint);
<21>            String secondHalf = oldString.substring(halfwayPoint, stringLength);
<22>            String splitString = secondHalf + "-" + firstHalf;
<23>            System.out.println("Split string: " + splitString);
<24>            System.out.println();
<25>
<26>            //remove all the vowels from the string object.
<27>            String newString1 = oldString.replaceAll("[aeiou]", "");
<28>            System.out.println("New string: " + newString1);
<29>            System.out.println();
<30>
<31>             //move first word to the end of the string object
<32>            int positionOfSpace = oldString.indexOf(' ');
<33>            String substring1 = oldString.substring(0, positionOfSpace);
<34>            String substring2 = oldString.substring(positionOfSpace, stringLength);
<35>            String newString2 = substring2 + " " + substring1;
<36>            System.out.println("New substring: " + newString2);
<37>            System.out.println();
<38>
<39>         }//end of main method
<40>   }//end of class
```

The following detailed analysis of the program will help you conduct a desk check. Be sure you understand the detailed explanation of each line of code and the syntax of each statement. Two sections of code are highlighted and deserve your special attention because they use shortcut notation for the arithmetic/assignment operator and the increment operator as described in the eIMACS lesson.

Line(s)	Purpose of Statement
< 1>	Declares StringPractice to be the name of the class.
< 2>	Curly brace marks the beginning of the class (matches up with Line <40> which is the end of the class).
< 3>	The beginning of the main method.
< 4>	Curly brace marking the beginning of the main method (matches up with Line <39> which is the end of the class).
< 5>	Comment describing purpose of the next segment of code.
< 6>	Declares a String object and assigns a String literal to it.
< 7>	Concatentates a String literal with a String object and prints output to the screen.
< 8>	Declares an int variable, invokes the length() method on the oldString object and assigns the value to the stringLength variable.
< 9>	Concatentates a String literal with the value indicating the length of the String object.
<10>	Prints a blank line to improve appearance of screen output.
<11>	Whitespace to improve readability of the source code.
<12>	Comment describing the purpose of the next segment of code.
<13>	Declares replaceCharacters to be a String object. The replace() method is invoked on the oldString object to replace the word “Four” with the number “4”. The modified String object is assigned to replaceCharacters.
<14>	The replace() method is invoked on the replaceCharacters String object to replace the word “seven” with the number “7”.
<15>	Concatenates a String literal with the String object replaceCharacters and prints the modified output to the screen.
<16	Prints a blank line to improve appearance of screen output.
<17>	Whitespace to improve readability of the source code.
<18>	Comment describing the purpose of the next segment of code.
<19>	Declares halfwayPoint to be a variable of type int. The stringLength is divided by 2 and assigned to halfwayPoint.
<20>	Declares firstHalf to be a String object. The substring() method is invoked on the oldString object and the first half of the oldString object is assigned to the firstHalf String reference variable.
<21>	Declares secondHalf to be a String object. The substring() method is invoked on the oldString object and the second half of the oldString object is assigned to the secondHalf String reference variable.
<22>	Declares splitString to be a String object. Concatenates the secondHalf String object, a hypen, and the firstHalf String object together as a single String literal.
<23>	Concatenates a String literal with the String object splitString and displays the output to the screen.
<24>	Prints a blank line to improve appearance of screen output.
<25>	WhiteSpace to improve readability of the source code.

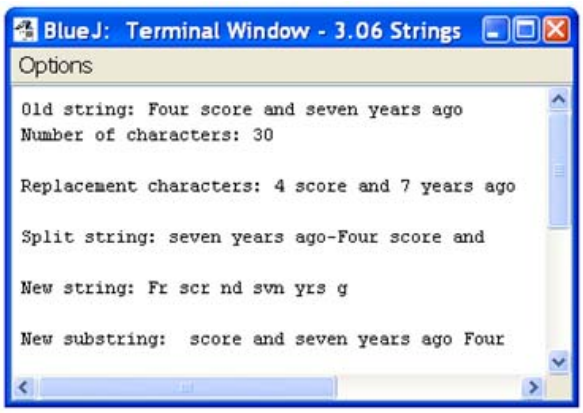
<26> Comment describing the puppose of the next segment of code.
<27> Declares newString1 to be String object. The replaceAll() method is invoked on the oldString object and all vowels are removed by replacing them with an empty pair of parentheses.
<28> Concatenates a String literal with the String object newString1 and prints the output to the screen.
<29> Prints a blank line to improve appearance of screen output.
<30> Whitespace to improve readability of the source code.

<31> Comment describing the purpose of the next segment of code.
<32> Declares positionOfSpace to be a variable of type int. Invokes the indexOf() method on the oldString object to determine the position of the first blank space in the oldString object and assigns that value to the positionOfSpace variable.
<33> Declares substring1 to be a String object. Invokes the substring() method on the oldString object to extract the first word (beginning up to space) fron the string of characters. Assigns these characters to the substring1 object.
<34> Declares substring2 to be a String object. Invokes the substring() method on the oldString object to extract the portion of the string from the first blank space to the end of the string of characters. Assigns these characters to the substring2 object.
<35> Declares newString2 to be a String object. Concatenates substring2, a blank space, and substring1 together and prints and assigns the revised string of characters to the newString2 object.
<36> Concatenates a String literal with the newString2 object and prints the output to the screen.
<37> Prints a blank line to improve appearance of screen output.
<38> Whitespace to improve the readability of the source code.

<39> Curly brace indicating the end of the main method (matches up with the curly brace on Line <4>).
<40> Curly brace indicating the end of the class (matches up with the curly brace on Line <2>).

Expected Output

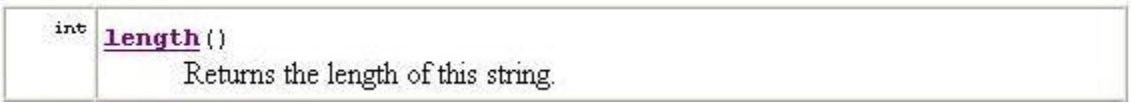
When you compile and run the program you should see the following output.



Try to match up the displayed results with the corresponding segments of code.

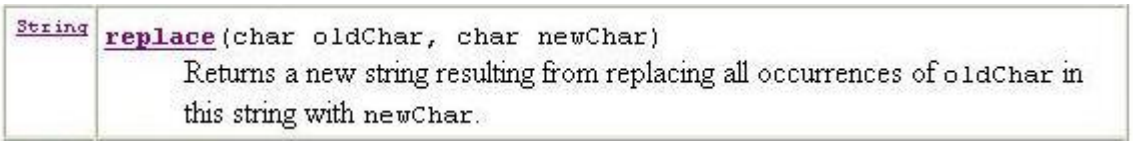
Code Analysis

Compare each of the following segments of code with the information in the corresponding Summary Table.



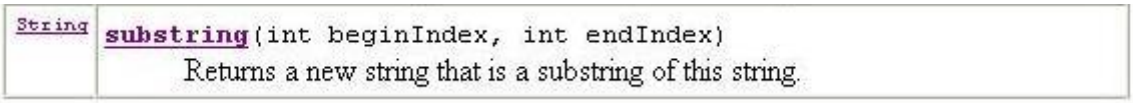
< 8> int stringLength = oldString.length();

- The length() method does not require any parameters (i.e. the parentheses are empty).
- It returns an integer value representing the length of the string which is assigned to the variables stringLenght.



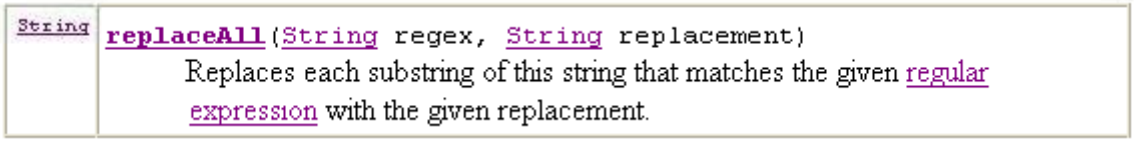
<13> String replaceCharacters = oldString.replace("Four", "4");
<14> replaceCharacters = replaceCharacters.replace("seven", "7");

- The replace() method takes two parameters.
- It returns a String with the old characters replaced by the new characters.



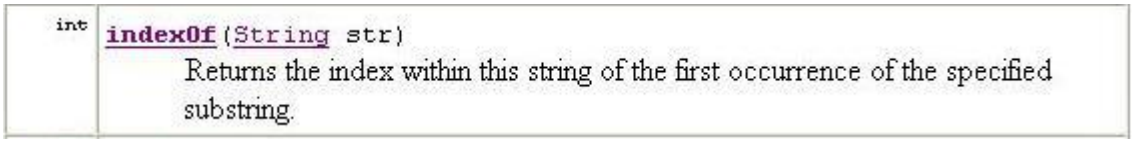
<20> String firstHalf = oldString.substring(0, halfwayPoint);
<21> String secondHalf = oldString.substring(halfwayPoint, stringLength);

- The substring() method takes two int parameters.
- It returns a new substring from the starting position to one less than the ending position.



<27> String newString1 = oldString.replaceAll("[aeiou]", "");

- The replaceAll() method takes two String parameters and replaces each character listed with the corresponding replacement character, which in this case was an empty space.
- It returns a new String with the replacements.
- This method is fairly advanced and will require some study of the Method Detail table to fully appreciate.




```
<32>        int positionOfSpace = oldString.indexOf(' ');
<33>        String substring1 = oldString.substring(0, positionOfSpace);
<34>        String substring2 = oldString.substring(positionOfSpace, stringLength);
```

- The indexOf() method takes a single String parameter representing a substring to be matched with characters of the target string.
- It returns an integer value representing the position in the string where the substring was found.

.

Modifications

As usual, the best way to extend your knowledge of programming is to play with code by making modifications and observing the results. However, be sure that when you make a modification you understand the reasons for any changes you observe. Some changes may not make sense in the context of the program, but they will still teach you something important.

1. Change the String literal assigned to the oldString object.
2. Change the parameters of each method to correspond with the new String object.
3. Study the Method Detail table for the replaceAll() method to see what else can be done.

Here's a challenge for you. Using only the methods covered in this Desk Check, how could you determine the number of vowels in the following String literal?

> To be, or not to be: that is the question. Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous fortune, or to take arms against a sea of troubles.

Be sure you complete the Desk Check before going on to the next lesson because you will need to apply the knowledge you gain here to the next assignment.