Activity 3: Classes and Objects

Reading:

Chapter 3.1 - 3.6	pages 114-137
Chapter 3.8 - 3.11	pages 141-154

Terminology:

new operator	return	DecimalFormat class
class	alias	wrapper class
object	garbage collection	autoboxing
reference variable	String class	unboxing
instantiation	immutable	component
null	Java API	container
method	package	frame, JFrame class
static method	import declaration	panel, JPanel class
constructor	Random class	label, JLabel class
dot (**.**) operator	Math class	ImageIcon class
parameter	NumberFormat class	

COMP 1210 Coding Standard Additions

- When using the import declaration, import classes from the same package individually (i.e. do not use the * notation).
- When making calculations with π, use Math.PI rather than the literal value 3.141. You should always try to use constants when available.
- From now on, use nextLine to get numerical input from the user and convert it to a number using methods from the corresponding wrapper class.

Due: Wednesday, Septempber 5, 2012 by the end of the lab.
(Note that you should attempt to do most of this activity prior to your lab on Wednesday.)

Goals:

By the end of this activity you should be able to do the following:

- ➢ Understand how to instantiate an object using the **new** operator
- ➢ Use the Java standard library (the Java API) to look up a class
- ➢ Use the String, and Scanner class

Program Description:

You will create a program that exchanges letters in a String to encode a message.

In-lab Directions:

MessageConverter.java

- This program will read in a message from the user and then require the user to enter a number. The following will happen based on the number entered by user:
  - o If the user enters a 1, the message will be printed as it was entered.
  - o If the user enters a 2, the message will be printed in lower case.
  - o If the user enters a 3, the message will be printed in upper case.
  - o If the user enters a 4, the program will replace all vowels with underscores.
  - o Any other number should generate an appropriate message.

- Create a class called **MessageConverter** that includes a main method. Also add an import for java.util.Scanner.

- Declare the following variables:
  - **message**: String object that holds user input
    ```
    String message;
    ```
  - **result**: The message that has been converted (String). **Initialize** the message to an empty String ("").
    ```
    String result = "";
    ```
  - **outputType**: An int representing the type of output the user selects.
  - **userInput**: Scanner object for getting user input:
    ```
    Scanner userInput = new Scanner(System.in);
    ```

- Get user input using the nextLine() method of the Scanner class:
  ```
  System.out.print("Type in a message and press enter:\n\t> ");
  message = userInput.nextLine();
  ```
- Now get the user's input for the output type. Make sure that you understand how the code below works.

```
System.out.print("\nOutput types:"
   + "\n\t1: As is"
   + "\n\t2: lower case"
   + "\n\t3: UPPER CASE"
   + "\n\t4: v_w_ls r_pl_c_d"
   + "\nEnter your choice: ");
```

$$\texttt{outputType = }\underbrace{\texttt{Integer.parseInt(}\underbrace{\texttt{userInput.nextLine()}}_{\text{Get user input as a String}}\texttt{)}}_{\substack{\text{Use the static parseInt method of the Integer class to} \\ \text{convert the string to an int value.}}}\texttt{;}$$

- Set up an if statement that will print out the appropriate message based on the user's selected output type. Note that you can write code for alternate conditions in the if statement using else if after the if condition.

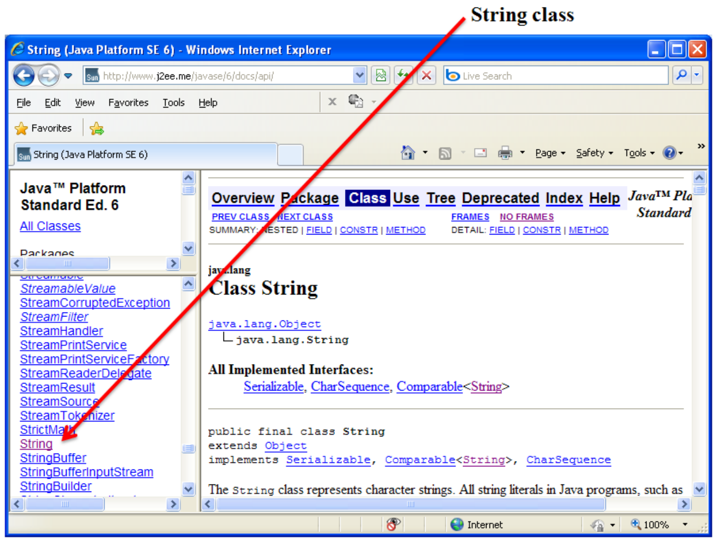
```
if (outputType == 1) { // as is

}
else if(outputType == 2) { // lower case

}
else if(outputType == 3) { // upper case

}
else if(outputType == 4) { // replace vowels

}
else { // invalid input

}
```

Fill in the if statement in the following steps.

- The first condition just sets the result equal to the original message.

```
if (outputType == 1) { // as is
   result = message;
}
```

- Because message is an instance of the String class, we can use the methods that come with the String class to modify input and return variations of the string it holds. In order to look at the methods that String has available, go to the Java API at http://docs.oracle.com/javase/6/docs/api/ and find the String class at the bottom left of the page:


String class
String (Java Platform SE 6) - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites
Page Safety Tools
Internet 100%

Java™ Platform Standard Ed. 6
All Classes
Packages

StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder

Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.lang
Class String

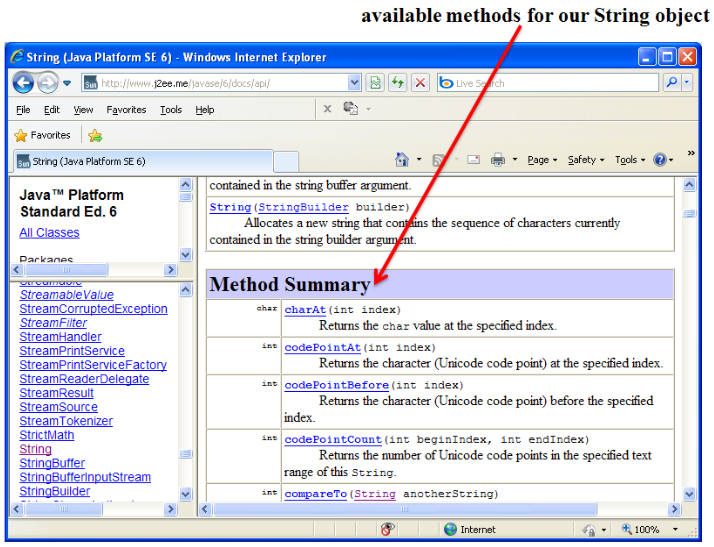
```
java.lang.Object
  └ java.lang.String
```

All Implemented Interfaces:
Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The String class represents character strings. All string literals in Java programs, such as

- In the main window on the right side, scroll down to method summary to see the methods that our String object **message** can use:


available methods for our String object
String (Java Platform SE 6) - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites
Page Safety Tools
Internet 100%

Java™ Platform Standard Ed. 6
All Classes
Packages

StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder

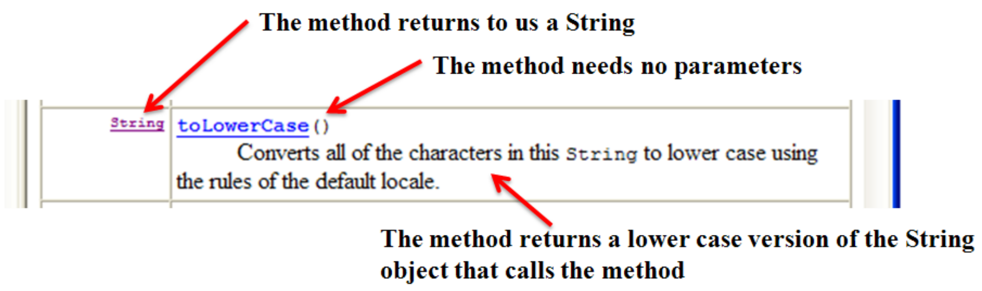
contained in the string buffer argument.

String(StringBuilder builder)
Allocates a new string that contains the sequence of characters currently contained in the string builder argument.

Method Summary

char	charAt(int index) Returns the char value at the specified index.
int	codePointAt(int index) Returns the character (Unicode code point) at the specified index.
int	codePointBefore(int index) Returns the character (Unicode code point) before the specified index.
int	codePointCount(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this String.
int	compareTo(String anotherString)

- Find a method called toLowerCase, and take a look at the method summary.
  - o What *parameters* do we need to give the method?
  - o What does the method do?
  - o What does the method *return*?


The method returns to us a String
The method needs no parameters
The method returns a lower case version of the String object that calls the method

String toLowerCase()
Converts all of the characters in this String to lower case using the rules of the default locale.

<u>The **toLowerCase()** method does not modify the actual String object that calls it</u>. Instead it returns a lower case version of message while **message** stays the same.

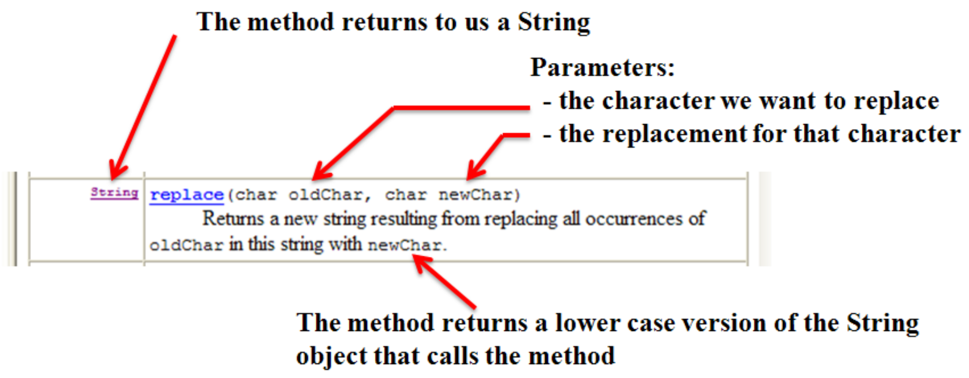
- Based on this knowledge, you can now use the toLowerCase method of our **message** object to change the message to lower case. Place the following code in second block of the if statement:

```
else if(outputType == 2) { // lower case
   result = message.toLowerCase();
}
```

- Look at the API description for the toUpperCase method and place the following line in the second block of the if statement:

```
else if(outputType == 3) { // upper case
   result = message.toUpperCase();
}
```

- Now write code that will shorten the message when appropriate. In order to change certain characters, we need to use the replace method of String to change certain characters of our string. Take a look at the replace method in the Java API:


The method returns to us a String
Parameters:
- the character we want to replace
- the replacement for that character
The method returns a lower case version of the String object that calls the method

String replace(char oldChar, char newChar)
Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

- First, use the method to change all of the a's to underscores. The following line of code replaces the characters of input and then stores the new version of input.

```
else if(outputType == 4) { // replace vowels
result = message.replace("a", "_");

}
```

- Now complete the translation by converting the remaining characters:

```
else if(outputType == 4) { // replace vowels
   result = message.replace("a", "_");
   result = result.replace("e", "_");
   result = result.replace("i", "_");
   result = result.replace("o", "_");
   result = result.replace("u", "_");
}
```

- Finally, the result will be an error message if :

```
else { // invalid input
   result = "Error: Invalid choice input.";
}
```

- Print out the result after the else block:

```
System.out.println("\n" + result);
```

- Test your program under each of the conditions:
  Original ("as is"):

```
Type in a message and press enter:
        > This is a test

Output types:
        1: As is
        2: lower case
        3: UPPER CASE
        4: v_w_ls r_pl_c_d
Enter your choice: 1

This is a test
```

  Lower case:

```
Type in a message and press enter:
        > This is a test.

Output types:
        1: As is
        2: lower case
        3: UPPER CASE
        4: v_w_ls r_pl_c_d
Enter your choice: 2

this is a test.
```

Upper case:

```
Type in a message and press enter:
        > This is a test.

Output types:
        1: As is
        2: lower case
        3: UPPER CASE
        4: v_w_ls r_pl_c_d
Enter your choice: 3

THIS IS A TEST.
```

Vowels replaced:

```
Type in a message and press enter:
        > This is a test

Output types:
        1: As is
        2: lower case
        3: UPPER CASE
        4: v_w_ls r_pl_c_d
Enter your choice: 4

Th_s _s _ t_st
```

Invalid input:

```
Type in a message and press enter:
        > This is a test

Output types:
        1: As is
        2: lower case
        3: UPPER CASE
        4: v_w_ls r_pl_c_d
Enter your choice: 5

Error: Invalid choice input.
```

- **Try the input "A message" for the message and choose to replace all vowels with underscores. What do you notice about the output? Why do you think that one of the vowels was not replaced? How would you go about correcting the issue?**