CS1110 Lab 04. Writing functions Fall 2011 Name _____ NetId _____

The purpose of this lab is to give you practice with developing the bodies of methods. At the same time, this lab will give you practice with Strings. We begin with some information on Strings. We also introduce you to the equality comparison operator == and its counterpart, function equals. After this lab, study Section 5.2 of the text, beginning on page 175.

A String object (instance, or folder) associates a number with each character in its list of characters. The number is called the *index* or position of the character. Type the following line into the Interactions pane of Dr. Java:

String s= "Java is fun.";

String object s now contains the list of characters "Java is fun.". The index of each character is shown below:

```
    Index:  0  1  2  3  4  5  6  7  8  9 10 11
Character:  J  a  v  a     i  s     f  u  n  .
```

Note that the index of the first character is 0 (not 1) and that the period and each of the space characters between each of the words each have an index.

In the string "I will study every day.", what is the index of the character 'w'? Of the last space character? Write down your answers:

A list of some functions that appear in each String object is given at the end of this handout. Refer to it when doing this lab. Note also that if a String s contains only digits (not even blanks), then the function call

Integer.parseInt(s)

yields the integer represented by s. For example, Integer.parseInt("345") is 345.

Important point about Equality

Symbol == is used for equality testing. You know that 2+3 == 5 has the value true. Importantly, when x and y are of the same class-type, the test x == y is made on the names (on the tabs) of the object. Therefore, the following expression is always false because two objects, with different names, are created:

new C(args) == **new** C(args)

Evaluate the following expressions in the Interactions pane and write down their values. In the third one, for each occurrence of "ab", note that evaluation in the Interactions pane creates a new object of class String.

new String("ab") == new String("ab")	value:
new Integer(5) == new Integer(5)	value:
"ab" == "ab"	value:

Class Object, the superest class of them all, has a boolean function equals(Object), which in class Object is defined to work exactly like ==. Each class can override function equals, and the convention is to define equals to test for the equality of all the fields in two objects. For example, classes String and Integer override equals(Object). To see this, try the following in the interactions pane and write down their values:

(new String("ab")).equals("ab")	value:
(new Integer(5)).equals(new Integer(5))	value:
"ab".equals("ab")	value:

You need to understand the distinction between == and function equals. Read about equality of strings on page 179 and equality testing on page 118.

Writing methods that deal with strings

File Methods.java contains specifications for a bunch of functions for you to write. Download the file from the course webpage and put it in its own directory —always put separate projects in separate directories. The function bodies have "stub" return statements so that the class will compile. Write the bodies of as many of them as you can in this lab. You probably won't finish them. We hope that you will finish THREE of them during the lab —show them to your lab instructor at the end of the lab. How many of the others you do is up to you. The more you practice, the easier developing such programs will become.

The methods will, among other things, change a time in a String into a different format. The time comes in four formats:

24-hour-string:	"<hours>:<minutes>" <hours> is in 0..23 and <minutes> is in 0..59. Examples: "4:20" "13:0" "23:59" "0:0"
AM-PM-string	"<hours>:<minutes>AM" or "<hours>:<minutes>PM" <hours> is in 0..11 and <minutes> is in 0..59 Examples: "4:20AM" "1:0PM" "11:59PM" "0:0AM"
24-hour-verbose:	Example: "4 hours and 20 minutes". Example: "23 hours and 59 minutes" Note: exactly one blank between each of the pieces.
24-hour-correct:	Exactly like the 24-hour-verbose format except that it is grammatically correct. So, instead of "1 hours and 20 minutes" it reads "1 hour and 20 minutes" and instead of "0 hours and 1 minutes" it reads "0 hours and 1 minute".

What to do

First, download file Methods.java from the course website; it has all the methods in it already. The function bodies have "stub" return statements so that the class will compile.

Second, create a **JUnit** test class, as usual.

Third, for each function in the class Methods, in turn, do the following:

1. Write the body of the method. *Note that all methods have suitable javadoc specifications.*
2. Think about what test cases would be necessary for you to know that the method is correct. Create a testX method in class MethodTester and insert those test cases.
3. Test the method. When finished writing and testing (THREE of) the functions in class Method, show them to your lab instructor. Save both .java files that you created on a USB storage key or email them to yourself. If you do not have time to finish three in the allotted time, then show the completed lab to your instructor the next week.

Advice: Need help? Ask for it. Don't waste time! Some pondering is necessary, but don't overdo it.

Guidelines: Check the javadoc comments (click button javadoc) to make sure they are right. Keep your program indented properly, and don't let lines get too long. Everything should be readable.

s.length()	= length of s, that is, the number of characters in it. Can be 0. "abc".length() is 3
s.charAt(i)	= character at index i of String s, which we might write as s[i]. The result is of type char.
s.substring(b,e)	= the String s[b..e–1] —i.e. the chars s[b], s[b+1], ..., b[e–1] . "abc".substring(1,3) is "bc"
s.substring(b)	= the String s[b..], or s[b..s.length()–1] . "abc".substring(1) is "bc"
s.indexOf(s1)	= index of the first char of the FIRST occurrence of String s1 in s (–1 if s1 does not occur in s). "abbc".indexOf("b") is 1
s.indexOf(c)	= index of FIRST occurrence of char c in s (–1 if s1 does not occur in s). "abbc".indexOf('b') is 1
s.lastIndexOf(s1)	= index of first char of the LAST occurrence of String s1 in s (–1 if s1 does not occur in s) . "abbc".lastIndexOf("b") is 2
s.trim()	= a copy of s but with any preceding and ending whitespace removed. " abbc ".trim() is "abbc"
s.startsWith(s1)	= "s begins with String s1", i.e. = true if s begins with s1 and false otherwise.
s.endsWith(s1)	= "s ends with String s1". "abbc".endsWith("c") is true
s.equals(s1)	= true if s and s1 contain the same sequence of characters, i.e. the same strings.
s.compareTo(s1)	= negative, 0 , or positive, depending on whether s is less than, equal to, or greater than s1. The comparison is based on alphabetic ordering, as in the dictionary. "abbc".compareTo("a") is 3 "abbc".compareTo("abbcdb") is –2