CS100J Lab 03. Strings and wrapper classes Fall 2003

Name _____ Section time _____ Section instructor _____

An object of class `String` contains a list of characters. The values `"Java is fun."`, `"box"`, and `"a%8ju& !"` could each be stored in a `String`. Section 5.2 of the class text is a good reference on class `String`; refer to it often.

Index of a character in a string

A `String` object (manilla folder) associates a number, called its index, with each character in its list. 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:

```
s       J a v a   i s   f u n .

index   0 1 2 3 4 5 6 7 8 9 10 11
```

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

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

Open a web browser and look at the API specification for the class `String`. The URL for the Java API is http://java.sun.com/j2se/1.4.2/docs/api/ and in the left lower pane you should scroll down to class "String" and follow the link. Ask your TA or a consultant for help if you can't bring up the correct page. As you do the exercise below, **look at the API specification for the method you are using and read its description**.

Type the following expressions into the interactions pane and write down their values.

`s.indexOf("f")`	`s.indexOf("z")`
`s.indexOf("is")`	`s.indexOf("a",2)`
`s.indexOf("a")`	`s.indexOf(" ")`
`s.lastIndexOf("a")`	`s.lastIndexOf("s")`

Write down your answer to the following questions:

What value does method `indexOf` return if the input argument does not occur in the string?

What does method `indexOf` return if the input argument occurs more than once in the string?

Is method `indexOf` a `function` or a `procedure`?

Type the following expression into the interactions pane and note the result: `s.length`

Is `length` a method or a field?

Is the length of a string the same as the index of its last character?

If a string has length 10, what is the index of its first character? What is the index of its last character?

Suppose that `String` variable `sentence` has been given a value (it contains some sequence of characters). Write an expression in the interactions pane and then in the space below that would evaluate to `true` if `sentence` has more than one letter '`a`' in it and would evaluate to `false` otherwise. (Hint: look at the methods in the table above.)

We look at three more string methods. Keep your web browser handy and read the description of the following methods in the API for class String.

Type the following expressions in the interactions pane and write down their values.

`s`	`s.charAt(s.length - 1)`
`s.charAt(0)`	
`s.charAt(1)`	
`s.charAt(s.length)`	<- you will get an error message

`s.substring(6)`	`s.substring(0,4)`
`s.substring(7)`	`s.substring(0, s.length)`
`s.substring(s.indexOf("f"))`	`s.substring(0, s.length - 1)`
`s.substring(0,3)`	`s.substring(0)`

`String firstString= "mouse";`	`firstString.compareTo(secondString)`
`String secondString= "zebra";`	`firstString.compareTo(thirdString)`
`String thirdString= "mouse";`	`firstString.compareTo(fourthString)`
`String fourthString= "ant";`	`firstString == thirdString`

What is the value of a call `s1.compareTo(s2)` when string `s1` appears before argument `s2` in a dictionary? (positive or negative integer) What about when `s1` and `s2` contain the same sequence of characters?

Finally, what do you think functions `toLowerCase` and `toUpperCase do`? Type these expressions into the interaction pane and copy their values onto this page:

"aBcDe.,$".toLowerCase()

"aBcDe.,$".toUpperCase()

The 'wrapper classes' Integer, Boolean, and Character

Primitive types and class-types `String` are different, and one can't use one in a place that requires the other. It would be nice to be able to use values of primitive types as if they were in objects (manilla folders). Java provides "wrapper classes" for this purpose. A manilla folder of class `Integer` has one field, of type int, which obviously contains an integer. Class `Integer` is called a "wrapper class", because a folder of that class "wraps around" the int value, much like you wrap a sandwich in saran wrap. So, if you want to use the value 52 as an object, then use this expression:

```
new Integer(52)
```

which creates a manilla folder of class `Integer` and puts the int value 52 in it.

In Java, each primitive type has a corresponding wrapper class. They are discussed in Chapter 5 of the text.

Look at the API for wrapper classes `Integer`, `Boolean`, and `Character` as we use their methods. **Read the description of each method used below**.

Type the following into the interactions pane and write down the result. Write "error" if you get an error message.

`Integer i= new Integer(7);`	
`int j= i`	can't do this!
`int j= i.intValue()`	this will work
`j`	

`String num= "107"`	
`int k= num`	can't do this
`int k= (int) num`	or this
`int k= Integer.parseInt(num)`	but you can use a method from class Integer!

`Boolean b= new Boolean(true);`	
`boolean c= b`	can't do this
`boolean c= b.booleanValue()`	
`!b.booleanValue()`	

`Boolean b2 = new Boolean("true");`	`Boolean b4 = new Boolean("false");`
`Boolean b3= new Boolean("TrUe");`	`Boolean b5 = new Boolean("green cheese");`
`b2.booleanValue()`	`b4.booleanValue()`
`b3.booleanValue()`	`b5.booleanValue()`

`Character c= new Character("a");`	"a" is a string, not a char - get an error
`Character c= new Character('a');`	
`char c2= c;`	Error
`char c2= c.charValue();`	`c2`

Many classes have a method `compareTo`, and they all "work" the same way (that is, they all follow the same conventions of when to return a positive integer, negative integer, or zero). Type the results of the following, or "Error".

`Character c3= new Character('z');`	`Character c4= 'Z';`
`c.compareTo(c3)`	`c.compareTo('A')`
`c3.compareTo(c)`	`c.compareTo(new Character('A'))`
`c.compareTo(c)`	`"The character is: " + c3`