CS 157 Lab 6 October 13, 2009
Strings and Introduction to JUnit

1. Create a new class called Lab6 that contains a main method. We will be talking about String objects in this lab, so you may wish to refer to pages 137-143 of the text as we work through the lab. We will also be introducing an extremely popular testing framework called JUnit (which is built into Netbeans). JUnit is a way of testing any of the methods that you will use in a project, and allows the programmer to have tests that do not have to be part of the program itself. So the tests are kept separately from the program that actually does what it is supposed to do. We will begin with some tests that do not really test methods, but test features of String objects. This is just to show you how JUnit works, and also to give you some instruction on simple use of Strings.

2. Read the text's description of the length method of a String on pages 137 and 138. But instead of using `println` to make sure we understand this, we will use JUnit. In the Projects pane right click on Lab6.java, go to Tools, and you should find Create JUnit Tests. Choose that and accept all of the defaults on the window that pops up. Netbeans will start a new class for you called Lab6Test that is a JUnit testing class. It contains a lot of things that you don't have to pay attention to at this time. Go down to the method called `testMain`. Delete everything from the body of that method. In this method, create a String variable called `name` and store in it the String "Joe Blow". What do you think the length of that String is? You can test your guess by using an `assertTrue` statement. This is not part of Java, but it is part of JUnit. You can type: `assertTrue(name.length() == 123);` (replace 123 by what you think the length should be). Make sure Lab6 is selected in the Project pane. Then go to the Run menu and select Test Project. After a few moments, JUnit will start and eventually tell you whether your test passed or failed. If your test passed, change the number and make it fail. If your test failed, correct the length so you can see what things look like when the test passes. Add a line that asserts that a null String (a String that has no characters in it) has a length of zero. Add another line that asserts that the String "how are you?" has length 12. Make sure the tests all pass.

3. Read in the textbook about the `indexOf` method on page 139 and in the table on page 142. Add an `assertTrue` statement that tests the location of the blank in "Joe Blow". The `indexOf` method takes a String argument that represents a String you are searching for within a given String. So `name.indexOf(" ")` is how you get the index of the blank. Test also the location of the String "Joe" and test the location of the String "Blow".

4. Add similar tests to exercise your understanding of the `charAt` method as described on page 139.

5. Add similar tests to explore the `substring` method. This method is described on page 140. Note carefully the meaning of the two parameters of this method -- they are very confusing. This method returns a String object, so it will be a little different testing it. So, for example, if we apply the substring method to "Joe Blow" and try to extract the String "Blo", we will have to use `name.substring(4, 7)`. Then to test this, we need another method of the String class called `equals`. So here is the assertion:
   `assertTrue(name.substring(4, 7).equals("Blo"));`

   Add other assertions to make sure you can extract the String "Joe", the String "lo", and the String "Blow".

6. So far we have used JUnit to test features of Java. This is NOT a usual practice. Now we would like to test a new method that is to be written. Go back to the Lab6 class and add a static method called `padString`. This method should take two parameters. The first is a String and the second an integer representing a length. The method is to return a String that consists of the first argument with as many spaces added to the end of it to have a length equal to the second argument. So, for example, `padString("abc", 5);` would return the String `"abc  "`. Once you finish writing this method, you need to test it. Rather than adding things to the main method, we will use JUnit. Go back to the Lab6Test class, and inside that class add a public void method called `testPadString` that takes no arguments. Inside that method is where we will test the `padString` method from the Lab6 class. Now the problem is that `padString` is not a method in the Lab6Test class. But that's easy to solve: we just call it with the name of its class added on to the front, as in `Lab6.padString("abc", 5)`. Assert that what is returned is equal to the appropriate String. Add a couple more test cases. Make sure all the tests pass.
7. Go back to the Lab6 class and add a static method called `reverse` that takes a String argument and returns a String argument. This method must return a String that contains the same characters as the argument, but in reverse order. For example, `Lab6.reverse("abc")` would return the String "cba". When you finish writing the method, return to Lab6Test and create a new method called `testReverse`, add appropriate tests and make sure they all pass.

When you finish the lab, put the two files Lab6.java and Lab6Test.java on the shared drive.