CS 157 Lab 12 November 10, 2009
Boolean expressions and Introduction to Files

Create a new class called **Lab12**.

1. Write a method named `season` that takes two integers as parameters representing a month and day, and that returns a `String` indicating the season for that month and day. Assume that months are specified as an integer between 1 and 12 (1 for January, 2 for February, and so on) and that the day of the month is a number between 1 and 31.

   For this problem, winter occurs between 12/16 and 3/15. Spring is between 3/16 and 6/15. Summer is between 6/16 and 9/15. Fall is between 9/16 and 12/15.

   Add a JUnit test class for this method that contains enough test cases that you are confident that the method works in all cases.

2. The **Scanner** class contains methods that allow us to handle input and output from the console, and in addition, from files that contain text. There are always difficulties when using files. Java expects the program to handle difficulties (exceptions) one way or another. For example, what is supposed to happen if you are trying to use a file, and it's not there? Generally, there are two ways to handle exceptions. The first way, which is what we will do in this lab, is to pass the problem on to whoever called us. This is called "throwing" the exception. Eventually, if the exception is passed on to the operating system, the operating system simply terminates our program with an error message. The alternative is to specify somehow what is supposed to happen when the exception occurs (meaning that we program in how the problem is to be handled). Naturally this is a little more involved, and we'll talk about it another time. There is one annoying thing about the way we will handle exceptions in this lab: every method that involves a file operation that could generate an exception must have a statement in the heading of the method that indicates that it is throwing the exception (unless it "handles" the exception, which we will not be doing in this lab). In this case, the phrase is "**throws FileNotFoundException**".

   Open a new terminal window, change directory down to the folder where your Netbeans Java project is located. Then use the following Unix command to copy the file called **wordfile** from the home directory of **jcaristi** to the current directory with the name "wordfile.txt": **cp ~jcaristi/wordfile wordfile.txt**
   Note the "squiggle" in front of "jcaristi"! It is necessary.

3. Add a main method to the Lab12 class. In this main method, you need to declare and create a File variable. Do this as follows: `File words = new File("wordfile.txt");`
   Now you need to make sure the file you copied is actually in the right location so that your program can find it. The variable **words** is an object of the class **File**. One of its methods is the boolean **exists** method. Call that method and make sure it returns **true** before continuing.

4. You can remove the code that tests to see if the file exists. Now create a Scanner object as usual, except that instead of using the **System.in** object, use the **words** object. At this point, (after you add the import statement for the Scanner class) Netbeans will complain that you are not dealing with a possible exception that can come up in using the file. Until this point, it was not necessary for there to actually be a file called "wordfile.txt". But when you put the File variable **words** into the Scanner object that will be used to read from the file, if the file does not exist at the moment you try to use it, an exception will occur. You can click on the red exclamation point and have Netbeans add the appropriate "throws" statement, or you can add it yourself to the signature of the main method.

5. The file “wordfile” contains one very long line with a lot of words separated by spaces. Add code to your main method that determines how many words are in the file. To help in this task, notice that the Scanner class has a boolean method called **hasNext** that you can use to control a loop. The Scanner class also has a method called **next** that returns the next String using a space as a delimiter. (A delimiter is the character or characters used to separate the “words” or “tokens” of a file or String.) Fill in the blank below.
   The number of words in the file is _____.

6. You can remove the code that counts the number of words in the file. Add code to your main method that determines what the last word in the file is. Fill in the blank below.
   The last word in the file is _____.

7. Remove the code from step 6. Add code that determines the number of times the word “if” appears in the file. Make sure you include “IF” and any other possibilities. Fill in the blank below.
   The number of times “if” appears in the file is _____.

8. Remove the code from step 7. Add code that determines the longest word in the file. Fill in the blank.
   The longest word in the file is _____.