CSCI 145 Week 3
Lab 2

Goals for this Lab

1. Increase familiarity with the String class.
2. Practice with method calling and the return statement.
3. Increase familiarity with variable scoping.
4. Understand fencepost loops and while loops.
5. Use Scanner lookahead.

Task Description

Exercise 1: Robust Input

In addition to the methods listed in the book, the `String` class has a number of methods. One that I find useful is `trim`. Here's a description of the `trim` method on `String`s:

`public String trim()`

Return a copy of the string with leading and trailing whitespace omitted. (Whitespace is any character, such as the space character, that produces white space when output.) This method can be used to trim whitespace from the beginning and end of the string.

Examples:

Method call	**Returns**
`"abc".trim()`	`"abc"`
`"  abc  ".trim()`	`"abc"`
`"  ab  c ".trim()`	`"ab  c"`

Consider the following program (This program is included in `lab2.zip`.):

```
import java.util.Scanner;
class yesorno {
   public static void main(String[] args) {
      Scanner console = new Scanner(System.in);
      System.out.print("What is your answer? ");
      String answer = console.nextLine();

      // Changes start here.

      if (answer == "yes")
         System.out.println("You answered yes.");
      else if (answer == "no")
         System.out.println("You answered no.");
      else
         System.out.println(answer + " isn't yes or no.");
      // Changes end here.
   }
}
```

This program has two problems:

1. It doesn't work as written. You will always get the last message no matter what is input.
2. If you enter `Yes` (capital Y) or `   Yes   ` (extra spaces before and after), the program will not recognize that as a yes.

The exercise:

1. Open the code in jGrasp and test it.
2. Fix problem #1 so that the proper messages are printed when `yes` or `no` are entered.
3. Using the `trim` and the `toUpperCase` and/or `toLowerCase` methods, fix problem #2. (This can be done as a one line fix.)
4. Test your code. When you are satisfied that it works correctly, save your corrected code.

Note: You can only change (add, delete, or modify) code occurring between the "Changes start here" and "Changes end here" comments.

Exercise 2: Get the sign of a number

Consider the following program:

```
public static int sign(int n) {
   if (n > 0)
      return 1;
   else if (n == 0)
      return 0;
   else if (n < 0)
      return -1;
}
```

This program can be found in `lab2.zip` as `returntest.java`.

Open program in jGrasp and compile it. There is a problem with the program. Identify the problem, fix it. Save the file. In a separate text file (`returntest.txt`), explain what went wrong and how you fixed it.

Here is an alternate version of the same program. Try to compile this version in jGrasp.

```
public static int sign(int n) {
   int answer;
   if (n > 0)
      answer = 1;
   else if (n == 0)
      answer = 0;
   else if (n < 0)
      answer = -1;
   return answer;
}
```

In the same text file you just used, state what error message you got for this program and explain why you got that error message.

Exercise 3: Variable scoping

Consider the following program (see `lab2.zip` for the source):

```
public class scopetest {

   static int num1 = 40;                          // #1
   static int num2 = 41;                          // #2
   static int num3 = 0;                           // #3

   public static void main(String[] args) {
      int num3 = 100;                             // #4
      add(10, 11);                                // #5
      add(num1, num2);                            // #6
      add(num2, num3);                            // #7
      System.out.println(num3);                   // #8
   }

   public static void add(int num1, int num2) {   // #9
      int inputNum = num1;                        // #10
      num2 = num2 + inputNum;                     // #11
      num3 += 1;                                  // #12
      System.out.println(num2);                   // #13
   }                                              // #14
}
```

Open the program in jGrasp and run it. In a text file (`scopetest.txt`) answer the following questions about this program. (The line numbers are there to help you refer to specific lines of the program when answering the questions.)

1. The program output consists of four numbers. Explain the output. That is, for each of the four numbers, where did that number come from?
2. What are the scopes of the num3 declared at line 4 and `inputNum` declared at line 10?
3. Suppose that the output of the `println` call at line 8 (the last of the four numbers output) is supposed to be the value of `num3` declared at line 3, how would you change the program to make that happen?

Comment: From the perspective of good style, this program is a mess. There are multiple declarations of variables with the same names. That's a real easy way to confuse yourself and others. In general, don't do that.

Exercise 4: Fence Posts

The text book in Chapter 5, pages 323-325, discusses “Fencepost Algorithms”. Consider the following program (`fencepost.java` in `lab2.zip`):

```
import java.util.Scanner;

public class fencepost {
   public static void main(String[] args) {
      System.out.print("Enter a line of text: ");
```

```
            Scanner console = new Scanner(System.in);
            String input = console.nextLine();

            System.out.print("You entered the words: ");
            Scanner linescanner = new Scanner(input);
            while (linescanner.hasNext()) {
               System.out.print(linescanner.next());
               System.out.print(", ");
            }
            System.out.println();
         }
      }
```

This program is will output the words that were entered separated by commas. However, when you run it, there is an extra comma at the end of the output. This is a classic fencepost algorithm problem. Using the technique described in the book, modify this program to output the words without the extra command at the end.

When you are satisfied with your program, save it. Note: If no words were entered, your program should output the phrase ("You entered the words: "), rather than failing with an exception. Note: the `next()` method in `Scanner` will fail with a `NoSuchElementException` if there are no words left.

References

Some useful references for this assignment:

- `Scanner` class: textbook, p166-171
- `String` class: textbook, p160-166
- Methods, parameters, return values: textbook, p137-159
- Fencepost loops: textbook, p323-325
- `while` loops: textbook p312-316
- Scanner lookahead: textbook, p345-346

Turn in your program on Canvas

Create a zip file with your work. It should contain the following five files: `yesorno.java`, `returntest.java`, `returntest.txt`, `scopetest.txt`, `fencepost.java`. Look for the Assignment link on the Canvas page for Lab 2 and click that to turn in the zip file you just created.

Ask your TA for help if you need it. Your grade for lab 2 depends upon your file being turned in properly. Among other things, this means you should turn in a zip file with exactly five files. You should not include `.class` files. It should be a `.zip` and not a `.7z`, `.tgz`, etc..

Due Date

Your program is due by 11:59pm, Sunday, January 24th.