Assignment 1: Logic

CSCI 110 Section 1

DUE at 11:59 pm on Monday August 29, 2016

Taking user input

During the first week, you learned about variables (of type integer, boolean, float, and String) and printing. For this assignment you'll write two programs that take user input (like we did in the first lab). The user input parts of this assignment are done for you, but this is background information so you know how it works.

The Scanner class can be used to ask the user for information. For example, here's a simple program that reads in an integer and doubles it. Save it as Doubler.java and compile and run it to see how it works.

```
import java.util.Scanner;
class Doubler {
    /**
     * Takes an integer and returns that integer times two.
     */
    public int doubleNumber(int n) {
        // this is where the magic happens
        return n * 2;
    }

    public static void main(String[] args) {
        Scanner reader = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = reader.nextInt();
        System.out.println("Here's the number times two: " + doubleNumber(n));
    }
}
```

You can search "java scanner" on the web to find the official documentation for the Scanner class. It's a lot to read but if you scroll down, you'll see a table called "Method summary" containing all of the things that a Scanner can do. We use the nextInt() method in the example above to read an integer that the user types in. There are also methods called nextFloat(), nextBoolean(), and next() to read in floats, booleans, and Strings.

Installing JUnit

You need the JDK if you don't already have it.

In this class, we'll use JUnit, a testing framework for Java. The hardest part of this assignment is probably installing JUnit and getting the CLASSPATH variable set up correctly.

- instructions to install JUnit on Windows (JUNIT_HOME means create a new folder to put the JUnit jar file in. See this documentation to change your CLASSPATH)
- instructions to install JUnit on Mac (.bash_profile is a hidden file in /home/your_username. You can edit it by typing "nano .bash_profile", hitting CTRL-O to save then CTRL-X to exit)

Essentially you're downloading the junit JAR file and then telling your Java installation where it is by adding to the CLASSPATH.

Testing your code

Starter code is provided for this assignment. For each question, there's a file in which you'll write code (Greeter.java, Cheater.java) and a file with unit tests (GreeterTest.java, CheaterTest.java). Besides testing by hand, you'll use unit tests to verify that your code works correctly.

Open your terminal (Terminal app for Mac or Command Prompt for Windows) and navigate to the directory where you unzipped the starter code using the "cd" (change directory) command. From there let's go into the greeter/ directory, which contains the starter code for part 1.

To test your code for the first question, run the following commands (the '$' symbol indicates a command line prompt. The commands to type in come after the '$'):

```
$ javac GreeterTest.java
$ java org.junit.runner.JUnitCore GreeterTest
```

If you've installed JUnit, the code should compile. Since you haven't completed the homework, you should see the test cases fail like this:

```
JUnit version 4.12
.E.E.E.E
Time: 0.008
There were 4 failures:
1) firstPersonOlderByDays(GreeterTest)
java.lang.AssertionError: expected:<1> but was:<0>
<<<...truncated by sheng...>>>
FAILURES!!!
Tests run: 4,  Failures: 4
```

A lot of stuff gets printed. The parts you'll want to pay attention to are **how many failures there were** and **what parts failed**. For instance, look at the first failure:

```
1) firstPersonOlderByDays(GreeterTest)
java.lang.AssertionError: expected:<1> but was:<0>
              at org.junit.Assert.fail(Assert.java:88)
              at org.junit.Assert.failNotEquals(Assert.java:834)
              at org.junit.Assert.assertEquals(Assert.java:645)
              at org.junit.Assert.assertEquals(Assert.java:631)
```

```
                at GreeterTest.firstPersonOlderByDays(GreeterTest.java:26)
...
```

The error message is "expected:<1> but was:<0>". And if you look five lines down, you see "at GreeterTest.firstPersonOlderByDays(GreeterTest.java:26)". This means you should look at GreeterTest.java line 26 to investigate why the test failed.

When all of the tests pass, it looks like this:

```
JUnit version 4.12
....
Time: 0.003

OK (4 tests)
```

Your grade for each part will be the number of test cases you pass divided by the total number of test cases, times the number of points. I have given you **some but not all** of the test cases I will use for grading.

You can also test your code by hand by compiling the implementation file. For instance:

```
$ javac Greeter.java
$ java Greeter
```

This will run the Greeter program directly. You can enter values in to see what it does:

```
Enter your name: Dwight Schrute
What year were you born? 1968
What month were you born? 1
What day were you born? 20
Enter another person's name: Simone Biles
What year was he/she born? 1997
What month was he/she born? 3
What day was he/she born? 14
---
Dwight Schrute and Simone Biles are the same age!
```

In this example the program is incorrect.

Part 1: Friend greeter [50 points]

Greeter.java currently contains a program that asks the user for the their birthday (year, month, day) and a friend's birthday. Then it congratulates whoever is older, or prints that the two people are the same age.

Your job is fill in the "whoIsOlder" method so that the program correctly determines who is older.

Part 2: Math cheater [50 points]

Cheater.java contains a program that reads in the three sides of a triangle. Then,

1. If it's equilateral (all three sides equal in length), it prints "The triangle is equilateral"
2. If it's isosceles (two sides equal, one side not), it prints "The triangle is isosceles"
3. Otherwise, it prints "The triangle is neither isosceles nor equilateral"

Your job is to fill in the "getTriangleType" method so that the program correctly determines the type of a triangle.

Extra credit [20 points possible]

I will reward you 1 point for each new test case you add to GreeterTest.java or CheaterTest.java.

Submission instructions

Submit your implementations and test files on Canvas for Assignment 1. These are the four files you should submit:

- Greeter.java
- GreeterTest.java
- Cheater.java
- CheaterTest.java

Make sure your programs compile before submitting! Programs that don't compile will receive an automatic 50% deduction - it's your job to debug your code.