CS 1110, LAB 3: CLASSES, METHODS, AND TESTING

Name: _____ **Net-ID**: _____

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab03.php

You may wish to use that version of the instructions.

The purpose of this lab is to get you used to writing methods, and to introduce you to the basics of testing. As a warning, we will tell you right now: **The Java file we give you has errors in it; do not look for them and test them in the beginning**. You should only correct it when you are told. The point of this lab is get you in the habit of testing your programs. Adopting this testing habit will prove to be unbelievably useful, particularly for the first assignment.

Requirements For This Lab. The very first thing that you should do in this lab is to download the file IntPoint3d.java from the course web page:

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab03/IntPoint3d.java

This is a variation of the file Point3d.java demonstrated in class. The primary difference is that the coordinates are of type int instead of double. You will be working on this file, so save it in a new directory. When you are done with this lab, you may want to save this file; email it to yourself or put it on a USB storage key.

This lab will involve three different components. First, you will need to write answers to written questions on these sheet. Second, you will need to modify the contents of IntPoint3d.java to add new methods. Finally, you will need to make a JUnit test case. When you are done, you should show *all three* to your lab instructor, who will record that you did it. You do not need to submit the actual paper, and you do not need to submit the computer files anywhere.

As always, if you do not finish during the lab, you have **until the beginning of lab next week to finish it**. You should always do your best to finish during lab hours. Remember that labs are graded on effort, not correctness.

---

1. Create a Test Class and Test the Constructor

For this part of the lab, you will create a JUnit test that tests both the constructor and the three getters. You are to continue testing, until you get no error messages. You are to go step-by-step as follows:

1.1. **Create the JUnit Class.** A JUnit test is a special class that is used to test other classes. In DrJava, select the menu `File` item `New JUnit Test Case...`. You should save it, at which point you wlll be asked to give a name for the new class. You should call it `IntPoint3dTester.java` and save it in the same directory that your `IntPoint3d.java` is in.

1.2. **Change the Name of Method `textX`.** DrJava always creates an initial test method for you called `testX`. We like to be more descriptive with our method names. Rename `testX` to `testConstructor`, and change the JavaDoc comments appropriately.

1.3. **Implement the Test Cases.** In the body of method `testConstructor`, write Java statements that do the following:

- Create an instance of `IntPoint3d.java` and save its name in a variable. Note that there is only one constructor, and it requires three integer arguments.
- Insert tests to verify that the constructor sets the fields properly.

For help on how to set up the tests, look at this example on how to test the constructor of the class `Worker.java` shown in lecture:

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab03/example-test.pdf

Remember, the special procedure `assertEquals(expected, computed)` stops all testing and outputs an error message if `expected` does not equal `computed`. Also remember that, unlike other methods, assertEquals is not proceded by an object. We will talk about why this is later.

1.4. **Compile the Two Classes.** Save the file, and click your mouse on the button `Compile`.

1.5. **Test.** Click the button `Test`. If an error message appears, study the message, the constructor and the relevant getter function to determine what is wrong. The error could be anywhere.

1.6. **Fix and Repeat.** You now have permission to fix the code in `IntPoint3d.java`. However, you should restrict your fixes to the constructor and getter methods only, as this is the only thing that you are testing. Do not fix any other methods yet.

After you fix the code, you should repeat steps 1.4 - 1.6 until there are no more error messages.

---

2. Test Function `hasAZero`

This function should return true if at least one of the x, y, or z coordinates is 0. If none of them are zero, it returns false. That is what the specification says, and you should not make any changes to the function before testing it.

In class `IntPoint3dTester`, you should make up another test function, `testHasAZero`, that will test the function `hasAZero`. A possible test case is any set of values (x, y, z).

Because this method can be "true in more than one way", you are going to want more than one test case. For example, you will want a test case in which x is 0, but none of the other coordinates are. You will also want a test case in which y is 0, but none of the other coordinates are. This is because maybe you might have a variable, and the method works for field x, but not for field y (and vice versa).

How many test cases do you think that you need in order to make sure that the method is correct? Perhaps 6 or 7 or 8? Write down (on the next page) a list of test cases that you think will suffice to assure that the function is correct:

2.1. **Implement Your Test Cases.** In test procedure `testHasAZero`, implement all the test cases that you think you need. The test procedure may have to create more than one instance of `IntPoint3d` in order to implement all of your test cases.

2.2. **Test and Fix.** Now compile the test program (click the button `Compile`) and run it (click the button `Test`). If you get an error message, look at the program and fix errors. Continue testing and debugging in this fashion until running the test program does not produce an error.

---

3. Add A Function Method to IntPoint3d

You may notice that there is a specification for a function called `length` in the class definition for `IntPoint3d`. However, there is no code for this function; only comments. You should write this function.

The specification of this function is that it should return the distance between this point and zero. For a point (x, y, z), this distance is $\sqrt{x^2 + y^2 + z^2}$. Fix this method so that it does just that.

Once you are done fixing the function `length`, compile your class and try out the method using these statements in the Interactions pane:

```
IntPoint3d d;
d = new IntPoint3d(3, 4, 5);
d.length()
```

Write your answer here:

Now do the same thing using this single statement:

```
new IntPoint3d(3,4,5)).length()
```

Write your answer here:

Note that the second way, just above, contains a new-expression that is *not* assigned to a variable. This is legal, and there is nothing wrong with it. The new object (folder) is created and stored in class (file-drawer) `IntPoint3d`, and the name of the folder is the result of the new-expression in the parenthesis. The function length() is called for the object in the parenthesis.

---

4. Add A Procedural Method to IntPoint3d

You may notice that there is a procedure called `averageWith` in the class definition for `IntPoint3d`. However, there is no code for this procedure; only comments. You should write this procedure.

The specification of this procedure is that it should average the coordinates of this point with those of the point q, and store the result in the fields for this point. For example, suppose the current object is (2,4,4) and q is the name for the object (4,6,-8). In that case, this procedure should average the x-coordinates and assign $x = (2 + 4)/2$. The procedure should do the same thing for the y and z coordinates.

4.1. **Test Your Code.** Once again, you should test and fix your procedure. To create a test case for a method like this, you work out the correct answer on a sheet of paper. The JUnit test should then verify that the program gives you the same answer that you worked out on paper.

Finding good test cases for a method like this is tricky, so we are giving you some specific ones to try out:

- The average of (2,4,4) and (4,6,-8) is (3,5,-2)
- The average of (1,1,1) and (2,2,2) is (1,1,1) [**Remember**: type int]
- The average of (0,0,0) and (0,0,0) is (0,0,0)

You should implement the three test cases above. Make a new test procedure called `testAverageWith` and add the test cases above using the special procedure `assertEquals`. If you do not get any error messages, then you are done.

---

5. Generate Javadoc Specifications

Now that you have finished working with `IntPoint3d`, there is one thing left to do. You are going to create a webpage for `IntPoint3d` that looks exactly like what you might find in the Java API.

Click the Javadoc button in DrJava. You will be asked where to save the result. When done, you should see a browser window open with a description of class `IntPoint3d`. More importantly, the comments in your class definition that are surrounded by "/** ... */" should appear in the generated webpage. You should see your class specification "An instance is a coordinate in three dimensions"; in the Method Summary and Method Detail sections you should see the specifications for your methods.

Check to see that this is indeed the case.