CIS 162 Lab 8

Debugging

Objectives

After completing this lab you should be able to:

- *find* errors in someone else's code

Step 1: Create a New BlueJ Project

Step 2: Download Debugging.java

We created a class with several methods that have some devious bugs! Create a new class called `Debugging` and delete all of the starter code. Cut and paste the provided code from (`Debugging.java`) into the newly created class. It should compile with no errors.

Step 3: Download DebuggingTest.java

We created a class with two JUnit tests to get you started. Create a new class called `DebuggingTest` and delete al of the starter code. Cut and paste the provided code from `DebuggingTest.java` into the newly created class. It should compile with no errors.

Step 4: Write some tests

- Read what each method in `Debugging` is supposed to do and then write test cases for each (except for `getInventory()`). Write one test case for each of the constructors and `setInventory()`. Write TWO test cases for the remaining five methods. However, `caughtSpeeding()` should have four tests. Therefore, your test harness should have fifteen tests.
- Do not attempt to fix any errors yet (in `Debugging.java`)

STOP! Show DebuggingTest to your instructor.

- Fifteen errors should be identified when running `main()`.

Step 5: Start Debugging

Use the debugging strategies described in Chapter 8 and in class. Each error is quite minor such as missing braces, wrong variable names, incorrect calculations and so forth.

1. Visually inspect the code
2. Sprinkle temporary print statements in strategic locations
3. Use the BlueJ debugger

Step 6: Test using zyBook AFTER all errors have been fixed

1. Upload your improved `Debugging.java` file to the appropriate Ch 8 zyLab.
2. Fix remaining errors (if any)

ALL DONE? Show the zyLab results to your instructor.

Turn In

- `DebuggingTest.java`. The code should be well documented.

Grading Criteria

This lab is worth a possible 3 points. One point for participating and 2 points for the completed tests.