Carleton University
Department of Systems and Computer Engineering
SYSC 2004 - Object-Oriented Software Development - Winter 2015

Lab 2 - Developing Java Classes with BlueJ

Objectives

In this lab, you'll learn how to develop a Java class using the BlueJ environment. You'll learn how to edit the source code for a class and test it interactively using the object bench and object inspectors. You'll also learn how to thoroughly test your class by running a suite of test cases that are based on the JUnit test framework.

Attendance/Demo

To receive credit for this lab, you must demonstrate your work. **Also, you must submit your lab work to cuLearn by the end of the lab period**. (Instructions are provided in the *Wrap Up* section at the end of this handout.)

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before Lab 3.**

Required Reading

If you do not have a copy of *Objects First with Java*, download a PDF copy of Chapter 2 from the Third Edition from http://www.bluej.org/objects-first/evaluation.html.

- Third Edition or Fourth Edition: Chapter 2, Sections 2.1 through 2.15.
- Fifth Edition: Chapter 2, Sections 2.1 through 2.18.

Getting Started

Step 1: Create a folder named Lab 2.

Step 2: Download heater.zip from cuLearn and store it in your Lab 2 folder.

Step 3: Right-click on the heater.zip folder and select Extract All... to extract all the files into a project folder named heater.

Step 4: Open the heater folder. Double-click on the BlueJ project file, which is named package. The will launch BlueJ and open the *heater* project.

A class diagram containing classes Heater and HeaterTest will appear in the BlueJ main window. You can ignore HeaterTest until you reach Exercise 3.

Exercise 1 - Using BlueJ to Develop a Class

Step 1: You have been provided with a very incomplete implementation of class Heater. Double-click on the Heater class icon. The Java source for the class will appear in an editor window.

Step 2: If you have the Fifth Edition of *Objects First with Java*, do Exercise 2.92. In Fourth Edition, this exercise is Exercise 2.83. In the Third Edition and the PDF of Chapter 2, this exercise is Exercise 2.82.

Don't create a new BlueJ project; instead, modify the Heater class that was provided to you.

Don't modify the setIncrement method. You'll finish this method in Exercise 2.

To compile your Heater class, click the Compile button in the editor window or right-click on the Heater class icon and select Compile from the pop-up menu. **Don't use the Compile button to the left of the class diagram.** If you click this button, BlueJ will also compile the HeaterTest class, which will result in a compilation error. This error will disappear when you work on Exercise 2.

Step 3: After you've written the no-parameter constructor, mutator methods warmer and cooler and accessor method getTemperature, interactively test your Heater class. To do this:

- right-click on the Heater class icon. A pop-up menu will appear.
- create a new Heater object by selecting new Heater() from the menu. A Create Object dialogue box will appear.
- click OK. A red box with rounded corners will appear in the object bench towards the bottom of the BlueJ window. This box represents the newly created Heater object whose name (identity) is heater1.

Perform these tests:

- Right-click on heater1 (i.e., right-click on the red box the represents the Heater object, not the icon that represents the Heater class). A pop-up menu will appear, listing the Heater object's methods. Select *Inspect* from the pop-up menu. An Object Inspector window will appear. Use the inspector to verify that the constructor correctly initialized the temperature field to 15.
- Right-click on heater1 and interactively call getTemperature by selecting this method from the menu. When the Method Result dialogue box appears, verify that method returns the correct value (15).
- Interactively call warmer and use the object inspector to verify that the temperature field now contains 20.
- Interactively call getTemperature and verify that this method returns 20.

- Interactively call cooler and use the object inspector to verify that the temperature field now contains 15.
- Interactively call getTemperature and verify that this method returns 15.

If necessary, correct your code and and repeat all of the tests. Do not start Exercise 2 until your class passes all of the tests.

Exercise 2 - Adding Attributes and Operations to the Heater Class

Step 1: If you have the Fifth Edition of *Objects First with Java*, do Exercise 2.93. In the Fourth Edition, this exercise is Exercise 2.84. In the Third Edition and the PDF of Chapter 2, this exercise is Exercise 2.83.

You will need to modify the signature of the constructor from:

```
public Heater()
```

to:

```
public Heater(int minTemp, int maxTemp)
```

Your class must declare four fields, named temperature, increment, min and max. Do not use different names for the fields.

Do not change the signatures of the warmer, cooler, getTemperature and setIncrement methods.

Do the exercise exactly as it's written (code a bit, test a bit, code bit, test a bit...) As you work through the exercise, use the object bench to interactively call methods on Heater objects and use an object inspector to verify that the object's state is correct after each method call.

The Heater class provided to you has javadoc comments placed immediately before the constructor and the methods. These comments describe the behaviour of the methods you coded for Exercise 1, but do not reflect the changes you make to the class in Exercise 2. For example, after you've modified warmer, the comment that states that this method always increases the heater's temperature setting by 5 is no longer correct. The comment should be edited to be consistent with the revised method code.

Make sure that you edit the javadoc comments for the constructor, warmer and cooler, to reflect their revised behaviour.

Step 2: You have just used a process known as *incremental*, *iterative development* to build class Heater. Even though you tested the methods one-by-one as you coded them, you should now perform some *unit tests* on the entire class.

- Create a new Heater object on the object bench. When the Create Object dialog box appears, initialize constructor parameters minTemp and maxTemp to 10 and 30, respectively.

- Open an Object Inspector window and verify that the constructor correctly initialized fields temperature, increment, min and max to 15, 5, 10 and 30, respectively.
- Repeatedly call warmer, and every time you call warmer, call getTemperature. Verify that getTemperature returns the heater's current temperature. Verify that the heater's temperature increases in 5 degree increments, and cannot be raised above 30.
- Repeatedly call cooler, and every time you call cooler, call getTemperature. Verify that getTemperature returns the heater's current temperature. Verify that the heater's temperature decreases in 5 degree increments, and cannot be lowered below 10.
- Call setIncrement to change the temperature increment from 5 to 8. Use the object inspector to verify that the increment field now contains 8.
- Test the warmer and cooler methods. Repeatedly call warmer, and verify that the heater's temperature increases in 8 degree increments, but cannot be raised above 30. Repeatedly call cooler, and verify that the heater's temperature decreases in 8 degree increments, but cannot be lowered below 10.
- Call setIncrement to change the temperature increment from 8 to -3. Use the object inspector to verify that the increment field was not modified (it should still contain 8).

If necessary, correct your code and and repeat all of the tests. (The technique of repeating all the unit tests after you modify a class, instead of testing only the code you modified, is known as *regression testing*. It ensures that when you change one part of a class, you don't inadvertently introduce new flaws that cause previously correct methods to fail their tests.)

Exercise 3 - Unit Testing with JUnit

Interactive unit testing of classes can quickly become tedious. Fortunately, a popular Java *test framework* named JUnit is built into BlueJ. Using this framework, we can easily build classes that test other classes. Typically, every test class contains a *suite* of *test cases* that test a single class or a group of closely-related classes. To test a class, we simply run the test cases in its test class and observe the results. Regression testing is easy, because we can use a single command to run all the test cases, every time we modify the class.

Step 1: First, we have to enable JUnit:

- From the menu bar, select Tools > Preferences... A Preferences dialogue box will appear.
- Click the Interface tab.
- Click the box labelled Show unit testing tools (a check-mark will appear in the box when the tools are enabled).

- Click OK.

Step 2: Class HeaterTest contains the suite of test cases that test class Heater.

- Compile both classes by clicking the Compile button to the left of the class diagram.
- Right-click on the HeaterTest class icon.
- Select Test All from the pop-up menu to run all the test cases. (Another way of running all the tests, without requiring you to display the pop-up menu, is to click the Run Tests button to the left of the class diagram.) A Test Results dialogue box will appear, listing the test cases that were executed. If all the methods you wrote for Exercise 2 are correct, there should be green check-marks to the left of all the test cases. An x to the left of a test case indicates that it failed.

If any of the test cases fail, the first thing to do is verify that the constructor is correct (repeat the first couple of interactive tests from Exercise 2, Step 2). You should also verify that accessor method getTemperature is correct.

Once you're certain that the constructor and getTemperature are correct, you can start searching for bugs in the other methods. Use the object bench to call methods interactively. Use an object inspector to help you determine where the problems are (e.g., before and after you call a method, what values are stored in the object's fields? Which values are correct? Which values are incorrect? What section of code in the method you just called changes those values?)

- If HeaterTest.testWarmer fails, the problem is likely in your warmer method.
- If HeaterTest.testCooler fails, the problem is likely in your cooler method.
- If HeaterTest.testMax fails, the problem is likely in your warmer method. Check the logic that determines whether the new temperature would exceed the maximum temperature.
- If HeaterTest.testMin fails, the problem is likely in your cooler method. Check the logic that determines whether the new temperature would drop below the minimum temperature.
- If HeaterTest.testSetIncrement fails, the problem is likely in your setIncrement method.
- If HeaterTest.testZeroAndNegativeIncrement fails, the problem is likely in your setIncrement method.

Test any changes you make to Heater, by performing interactive testing (using the object bench and object inspectors) and running all the JUnit test cases.

You've completed unit testing when every JUnit test case passes.

Wrap-up

1. Remember to have a TA review and grade your solutions to the exercises, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/sign-out sheet.

2. The next thing you'll do is package the project in a *jar* (Java archive) file named heater.jar. To do this:

    2.1. From the menu bar, select Project > Create Jar File... A dialog box will appear. Click the Include source and Include BlueJ project files check boxes. A check-mark should appear in each box. Do not modify the Main class field.

    2.2. Click Continue. A dialog box will appear, asking you to specify the name for the jar file. Type heater or select the BlueJ icon named heater in the list of files. **Do not use any other name for your jar file** (e.g., lab2, my_project, etc.).

    2.3. Click Create. BlueJ will create a file named heater that has extension .jar. (Note: you don't type this extension when you specify the filename in Step 2.2; instead, it's automatically appended when the jar file is created.) The jar file will contain copies of the Java source code and several other files associated with the project. (The original files in your heater folder will not be removed).

3. Before you leave the lab, log in to cuLearn and submit heater.jar. To do this:

    3.1. Click the Submit Lab 2 link. A page containing instructions and your submission status will be displayed. After you've read the instructions, click the Add submission button. A page containing a File submissions box will appear. Drag heater.jar to the File submissions box. Do not submit another type of file (e.g., a .java file, a RAR file, a .txt file, etc.)

    3.2. After the icon for the file appears in the box, click the Save changes button. At this point, the submission status of your file is "Draft (not submitted)". If you're ready to finish submitting the file, jump to Step 3.4. If you instead want to replace or delete your "draft" file submission, follow the instructions in Step 3.3.

    3.3. You can replace or delete the file by clicking the Edit my submission button. The page containing the File submissions box will appear.

        3.3.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the File submissions box, then click the Overwrite button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the Save changes button.

        3.3.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the Delete button., then click the OK button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, click the Save changes button.

3.4. Once you're sure that you don't want to make any changes to the project you're submitting, click the Submit assignment button. A Submit assignment page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the Continue button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading".