SYSC 2004 Object-0riented Software Development

Lab 3

Lab 3:

Background Reading

- *Objects First with Java*, Chapter 3.
- *The BlueJ Tutorial 2.0.1*. This tutorial can be downloaded from `http://www.bluej.org/doc/documentation.html` (also available from cuLearn).
- *The BlueJ Reference Manual 2.0*. This tutorial can be downloaded from `http://www.bluej.org/doc/documentation.html` (also available from cuLearn).

Objective

In previous labs, you've used BlueJ's *Object Inspector* to examine the state of objects. You may have used it as a very simple debugging tool. Another tool that helps us understand the execution of our programs is a *debugger*. The objective of this lab is for you to learn how to use BlueJ's debugger.

Getting Started

1. Download file `debugdemo.zip` from cuLearn. Save the file somewhere on your M: drive or in C:/Temp. (The M: drive is recommended, because C:/Temp is erased when you log out.)
2. Right-click on the `debugdemo.zip` folder and select Extract All... to extract all the files into a folder called `debugdemo`.
3. Download files `lap-timer-works.zip` and `lab-timer-buggy.zip` from cuLearn. Extract the projects in these files.
4. Launch BlueJ.

Part 1: Learning the BlueJ Debugger

1. Download the *BlueJ Tutorial* and *The BlueJ Reference Manual* from cuLearn. (These are the same documents that are available from `http://www.bluej.org/doc/documentation.html`. These are the first and fifth documents on that web page. Note that you do not need to read any of the other

documents now.) Open the tutorial in Acrobat Reader (or any other program that can display pdf files) and display the first page of Section 7, *Debugging*.

2. From the BlueJ menu bar, select Project > Open Project... An Open Project dialogue box will appear. Browse to the folder where your `debugdemo` project is stored. Select the `debugdemo` folder and click the Open button. A class diagram containing classes `Demo` and `Car` will appear. Click the `Compile` button to compile both classes.
3. Work through Chapter 7 of the Tutorial, experimenting with `debugdemo`. (Note that there is a summary of BlueJ's debugging tools in Section 6 of the Reference Manual.)
4. Show a TA your work for Part 1.

Part 2: Experimenting with `lap-timer-works`

1. Open project *lap-timer-works*. This application keeps track of a runner's time in races that consist of consecutive laps. The lap timer keeps track of the number of laps completed, the time of the lap just completed, the average lap time, the total time that the runner has been running, and the average speed of the runner.
2. Double-click on class `LapTimer` and look at the constructor. When a `LapTimer` object is created, its constructor creates an instance of class `TimingEngine` and an instance of class `UserInterface`.
3. Compile the classes. Interactively create an instance of class `LapTimer`. A window containing the user interface for this application will be created. (You may have to shrink or move some windows around your desktop to find the lap timer window).
4. When a race begins, the Start button is clicked . The message displayed at the top of the lap timer window changes from Stopped to Timing... and the lap counter and all the time displays are zeroed. As the runner finishes each lap, the Next lap button is clicked. This causes the lap counter to increase by 1 and the time and average speed displays to be updated. When the runner finishes the last lap, the Stop button is clicked and the timer is Stopped. Clicking the Stop button again has no effect until the timer has been restarted (at the beginning of the next race). Use the lap timer until you understand the values that are displayed each time a button is clicked. In the Lap length field, type 100 and click the Set length button. What happens? Enter a lap length of -200 and click the Set length button. What happens? Enter a lap length of 0 and click the Set length button. What happens?
5. There is nothing to show the TAs for part 2. Continue to part 3.

Important Notes for Parts 3, 4, and 5:

- All the problems that you will fix in the remainder of the lab are in the `TimingEngine` class. That's why it wasn't provided in the "works" project!
- It is recommended that you use the debugger to help locate the errors.
- You may also invoke methods directly on a `TimingEngine` object, and use the inspector on that object.

Part 3: Debugging `lap-timer-buggy` Section A

1. Open project *lap-timer-buggy*. Compile the project, and interactively create a `LapTimer` object. Use the timer. Clearly, there are several problems. First, there's a problem with changing the lap length. Try different values with the "buggy" and "works" versions. What is wrong with "buggy"?
2. Correct this problem in the "buggy" version and retest changing the lap length to make sure it works the same as in the "works" project.
3. Show a TA your work for Part 3.

Part 4: Debugging `lap-timer-buggy` Section B

1. Another problem is that clicking the Stop button two or more times in a row causes the display to change repeatedly.
2. Correct this problem. (Don't forget your testing, including regression testing.)
3. Show a TA your work for Part 4.

Part 5: Debugging `lap-timer-buggy` Section C

1. Finally, several of the values displayed in the window are incorrect.
2. Correct this problem. (Don't forget your testing, including regression testing.) Ensure that you have fixed all the bugs by checking the the "buggy" version now works identically to the "works" version.
3. Show a TA your work for Part 5.
