Lab 4

Due 15 February

Handout 4
CSC 062: Spring, 2007
12 February

Eclipse and Debugging

1 Using Eclipse

Today we are going to discuss using the Eclipse IDE and how to use the associated debugger. Eclipse is a free IDE that runs on virtually all computer platforms, and is already installed on all of the Macintoshes in the CS laboratory.

Copy the folder Assn4-Eclipse from the cs062/labs folder on your desktop into the folder where you keep your CS 062 programs. Also drag the file objectdraw.jar to your folder. The objectdraw.jar file is in folder cs062 on your desktop.

Then start Eclipse by clicking on the Eclipse icon, a big purple icon with a sun eclipsed by a moon. It can be found in the dock at the bottom of the screen. The first time that you run Eclipse, it will ask you which folder you want to use as its "workspace", where it stores files it needs. Please select the folder you created for your CS 062 programs as the workspace.

Once the Eclipse window has appeared, the first thing that you will need to do is to tell Eclipse where to find the objectdraw library, which is used in this program. From the Window menu select Preferences. Click on the triangle in the left pane next to "Java", and then the triangle next to "Build Path". Then click on "ClassPath variables" in the list that opens. In the right pane you will find a list of classpath variables. It is easier to tell your program about a library if a variable has been set up for it. You will now set up a variable for the objectdraw library. Click on "New..." at the right edge of the screen, and then type "OBJECTDRAW" in the Name field of the dialog box that pops up. (*It is very important that you write this in all caps, as otherwise projects I set up for you will be confused.*) Then click on the "File..." button and then navigate to and select the file "objectdraw.jar". Click on OK twice to get out of preferences.

To start working on the "BadCompressedGrid" program, open the "File" menu and select "Import...". Select "Existing Project into Workspace" (you may have to click on the triangle next to "General" to see this), and click the "Next" button. Now, click the "Browse" button and navigate to the BadCompressedGrid folder in your directory. Click "Finish." When you import the project, Eclipse will automatically use the project name that we have given it. You should not change the project name. (In fact, Eclipse won't let you!)

On the left side of the Eclipse window, in an area called the "Package Explorer " pane you should now see an entry for "BadCompressedGrid". Click once on the triangle to the left of "CompressedGrid" and then once on the triangle to the left of "(default package)' '. Then double-click on "GridTest.java".

The large area (called the editing pane) on the top right side of the screen displays the contents of any files that you double click, such as "GridTest.java". Also double-click on "CurDoublyLinkedList.java". Its code should now be showing on the right side of the screen. To go back to looking at "GridTest.java" you can click on the tab at the top of that pane. To open any other file, just double-click on it in the left pane. To close a file whose code you no longer need to see, click on its tab above the editing pane and then click on the "`x`" to the right of the name in the tab.

If you have any compile-time errors in your program, a red "X" will appear next to the file name, both in the Package Explorer and in the editing pane. An error message will appear in the Console pane, at the bottom right of the window. (This pane is also where the results of any `System.out.println(...)` commands will appear.) If you click on the error message, the editing pane will scroll to the line with the error, which will have an "X" on the left margin. If you hold the mouse over the "X" in the left margin, an error message will eventually appear. If it provides a suggestion for a fix in a second pop-up window, you can usually double click on the suggestion and it will perform it for you.

If you fix the error, the "X" will turn gray, and it will disappear altogether when you save the file. To see how this works, introduce an error into one of the classes and save the file. Then fix the error and save once more.

Eclipse uses an incremental compiler to recompile anything needing compilation whenever you save your program. Thus you will never need to do anything to explicitly compile it. You can execute an application in Eclipse by clicking on the file name (in the far left pane) that extends `WindowController` (if you were running an application it would be the class that contains the `main` method to be executed. Then go up to the "Run" menu and select "Run ..." and from there select "Java applet". Click on the icon with a small "+" at the left toward the top of the window. If necessary type in the name of the configuration, and browse to find the Project folder and then the applet to be executed.

When you click on the "Run" button on the bottom right side of the window, Eclipse will immediately start executing the applet class that you selected. Once you have run a program, you can re-execute it by either going through the same process or, more simply, by just clicking on the running person on the toolbar above the file names.

When you run BadGridTest with Eclipse, it will freeze after you change the color of a square and hit the "Display list" button. (With other kinds of errors, you might have the program crash with an exception. This will be indicated by the appearance of multiple lines in red in the Console pane in the lower right corner. Examining these lines would tell you what kind of exception was raised, and in which line it was generated.)

A frozen program is a bad thing to have to debug. However, because the problem appeared immediately after we changed the color of a square, it is reasonable to assume that the error was generated during the `updateInfo` method of `CompressedGrid`.

2 Running the Debugger

Identifying roughly where the program blew up is helpful, but we need more information. Why did the program freeze when we clicked on "Display list"? Did the list somewhow get corrupted?

The first thing we do is to set one or more "breakpoints" in the program. Find the line in `updateInfo` that calls find(posn). Double click in the left margin of the pane so that a small blue circle appears. If you were running the program in the debugger, the program would stop at that point in the program, which would allow you to take a look at the list. Now you are ready to run the debugger.

Make sure "GridTest.java" is selected and then go to the Run menu and select "Debug as..." and "Java Applet". (To run debug again, you need only click on the picture of the bug to the left of the running person above the Package Explorer pane.) The window will take a bit longer to appear than when running the program. Wait a few seconds after the grid appears for the debugger to come up in the background. Click on one of the middle squares of the grid and wait for the debugger to come forward.

When the debugger window comes forward you will see a somewhat more complex display. The upper left pane (labeled "Debug") shows all of the threads running in your program. There will be several shown there, but only one will be selected. It is the thread corresponding to the execution of your program.

The pane below the Debug pane shows the line of your program that was executing when it hit the breakpoint. In this case, it is the line with `find(posn)`. You can scroll in this window as usual or bring up other files from the tabs.

The pane in the upper right corner of the window should be labeled "Variables" (if not, click the Variables tab at the top of the pane). It shows all of the active variables in your program at this point. For example it should show `row`, `col`, `optimizeBefore`, and `newInfo`. The values of those instance variables that are primitive types will be shown (e.g., `row` and `col` will be integers whose values will depend on where you clicked). If a variable represents an object, you will be able to click on the triangle next to it to see the values of its instance variables. (Unfortunately, the info for colors seems to be pretty difficult to decipher.) If you don't want a variable expanded out, just click on the triangle next to it again, and it will fold up to the previous form.

At the top of the list is the word `this`. If you click on the triangle next to it, you will see all of the instance variables of the object executing the code. In this case, you will see all of the instance variables of the compressed table. Click on `tableInfo` to get information on the current state of the doubly linked list. The value of `size` will be an integer, while the values of current, header, and trailer will be listed as `DNode<E>` with an id number. Open up `current` by clicking on the triangle next to it to determine which nodes come before and after it. Use this information to write down a complete description of the list as it

currently exists (including the value of current). You might check to see what the key is of the data field for current. It should represent the position (0,0).

The icons to the right of the word "Debug" on the top of the Debug pane allow you to control the execution of your program. If you were to click on the green triangle, the program would start executing again. It will stop again only if it hits another breakpoint or terminates. If the program is executing, clicking on the two tall yellow rectangles will cause it to pause. Clicking on the red square will terminate your program.

The four yellow arrows allow step by step execution of your program. The first one (turning down) allows you to step into the method called in the highlighted line. If you don't want to see the details of that method's execution, but instead go to the next line of the method shown, click on the second arrow (which goes up and then down) to step over that line. Finally, the third arrow (going up and to the right) will finish executing that method and pop you out to the method that called it.

I suggest that you execute the program a line at a time, each time updating your picture of the list in order to see where it becomes corrupted. When you find the spot where it gets corrupted, you will probably have to quit the program and restart it in the debugger so that you can "step into" the method called in the step where it gets corrupted so that you can see what is going wrong.

For the rest of this lab, I would like you to determine what has happened to make the program freeze (it is actually in an infinite loop). You should be able to determine where this is by carefully tracking the list represented by tableInfo. Please write down a picture of the list, including head, tail, and current after each step that changes the list.

You may need to run the program several times to see where the error occurs, we each have a tendency to assume things are all right, when an error has actually already occurred!. Finally, please correct the error and explain what was wrong.

Incidentally, you can go back and forth between the debug view and the usual "Java" view of Eclipse by clicking on icons in the upper right hand corner of the window. Clicking on the icon with a "Java" next to it takes you to the Java view, while clicking on the bug icon with "Debug" next to it will take you to the debug view. Another handy shortcut is that double-clicking on the tab for a file in the edit pane will expand the edit pane to fill the window. Double-clicking it again will reduce it to the original size. This can be handy if you want to read more for editing and then reduce it so you can see the output from running the program.

3 What to hand in

Please hand in the info requested (including pictures of the linked list at different stages), as well as the explanation of how the error can be corrected. Please hand this in at the beginning of class on Friday.