Lab 5

Due 12 March

Handout 5
CSCI 136: Spring, 2004
10 March

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 CompressedGrid from the cs136/FilesToCopy folder on Cortland into the folder where you keep your CS 136 programs in your home directory. Also copy the file bailey.jar into that same folder.

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 create a folder in your home folder called "workspace" where it stores files it needs.

Once the Eclipse window has appeared, the first thing that you will need you to do is to tell Eclipse where to find the structure library. From the Window menu select Preferences. Click on the triangle in the left pane next to "Java". Then click on "ClassPath variables" in the list that opens. In the right pane you will find a list of classpath variables, including "OBJECTDRAW". It is easier to tell your program about a library if a variable has been set up for it. You will now tell set up a variable for the structure library. Click on "New..." at the right edge of the screen, and then type "STRUCTURE" 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 select the file "bailey.jar" that you downloaded. Click on OK twice to get out of preferences.

To start working on the "CompressedGrid" program, open the "File" menu and select "Import...". Select "Existing Project into Workspace" and click the "Next" button. Now, click the "Browse" button and navigate to the CompressedGrid 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!)

The window has three main regions. On the left, it displays the names of files and their contents in an outline form.

On the left side of the Eclipse window, in an area called the "Package Explorer " pane you should now see an entry for "CompressedGrid". 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 is now 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) whose main method you wish to execute. Then go up to the "Run" menu and select "Run as" and from there select "Java application". Eclipse will then immediately start executing the main method of the file 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 GridTest with Eclipse, it will crash with a null pointer error if you click in the middle of the grid. This will be indicated by the appearance of multiple lines in red in the Console pane in the lower right corner. Scroll to the top of the Console pane and you will see that you got a `NullPointerException`. The line immediately below that message will tell you that the error occurred while executing method `getCurrent` of `CurDoublyLinkedList`. If you click on that line, it will open the file in the edit pane and show that line of `CurDoublyLinkedList`. That line is:

```
return current.value();
```

suggesting that the value of `current` was null at that point.

The next line in the Console pane shows that the error occurred during the execution of method `handleAtEnd` from class `CompressedTable`. If you click on this, it will take you to that method. Notice that the line selected is one that calls the `getCurrent` method. That is, the error occurred when method `handleAtEnd` called method `getCurrent`, which crashed when evaluating `current.value()`.

Similarly the next line in the console shows this occurred during the execution of `updateInfo` in `CompressedTable`. We can trace this back further to method `onMouseClick`, but after that we get into system calls that aren't very meaningful.

2 Running the Debugger

Identifying where the program blew up is helpful, but we need more information. Why was `current` null? What did the list look like at that point? We can discover this using the debugger.

The first thing we do is to set one or more "breakpoints" in the program. Find the line in `handleNotAtEnd` that caused the error (i.e., click on the second line after `NullPointerException`). Double click in the left margin of the pane so that a small blue circle appears. If you are running the program in the debugger, the program will stop at that point in the program. 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 "Application". (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 `tableInfo.getCurrent()`. 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 bottom of the pane). It shows all of the active variables in your program at this point. For example it should show `successorPosn`, `oldInfo`, and `newInfo`. If you click on the triangle next to `successorPosn` you can see that it is an object with instance variables `col`, `numCols`, `numRows`, and `row`. Moreover the values of those instance variables will be shown (e.g., `numCols` and `numRows` will both be 20, while the values of `col` and `row` will depend on where you click). 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. Please write on a piece of paper (to be turned in later) the values of `count` and `current` as well as the positions (row and col only) of the elements of the doubly linked list. (I'll let you figure out how to find out this information.)

While it won't be very interesting at this point, 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 click on the green triangle, the program will start executing again. Stopping 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. Ignore the first one. The second 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 third arrow (which goes up and then down) to step over that line. Finally, the fourth arrow (going up and to the right) will finish executing that method and pop you out to the method that called it.

Because the program is about to crash, the only thing you can do is select the "step into" and then "step over" twice before the program crashes.

For the rest of this lab, I would like you to determine what has happened to make the value of current null when it crashes. In particular, look at the method `updateInfo` of `CompressedTable`. Depending on where the new element is added, it will call either method `handleEndOfList` or `handleNotAtEnd`. In each case, the method `addAfterCurrent(...)` will be called. Please write down a picture of the list, including head, tail, and current after each call to `addAfterCurrent` (until it crashes).

It may be helpful to note that each object used in your program is given a unique id number at run-time. Use the information gathered from the debugger to explain where the first error occurs in the program. (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 explain how the error can be corrected.

Incidentally, you can go back and forth between the debug view and the usual "Java" view of Eclipse by clicking on icons along the left border of the window. Clicking on the icon with a "J" on it takes you to the Java view, while clicking on the bug icon on the left border 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 either at the end of lab or at the beginning of class on Friday.