

Lab 10

As programs get larger, you will have more and more `.java` files and your debugging needs will also grow to the point that you will need the help of a debugger (you just can't put enough `println` any more!)

The Eclipse Integrated Development Environment (IDE) is a powerful software development environment. It is used by professional software developers writing much larger and more sophisticated programs and thus contain a myriad of features that may appear daunting at first. Spend time to familiarize yourself with it and remember that it takes time to learn any new tools.

1 Using Eclipse

1. Eclipse is already installed on the lab computers. Find Eclipse and start the application. You will be asked to select a directory as workspace. A workspace is the top level directory under which you keep all of your work. By now, you should already have a folder where you are keeping all of your programs, so navigate there and choose to launch there.
2. Eclipse organizes work into “projects”. Generally, you will use a separate project for each lab/assignment. This is equivalent to the subdirectories (A1, A2, etc) you have been using for each assignment. Create a project by clicking the down-arrow next to the leftmost icon in the toolbar (it's a window with a + in the corner). Choose to create a Java Project. Name it `lab10`.
3. Create a new Java class by clicking the green circle button with a C and a + on it (around the middle of the toolbar). Name the class `HelloWorld`. Now write your standard hello-world program. Run it by clicking on the “play” button (green button with a white triangle pointing right). Text will appear in the Console view toward the bottom of the Eclipse window.
4. Now that you know how to create and run a program in Eclipse, let's learn how to debug with it. Download `KWArrayListBuggy.java` and `ArrayListTest.java` from `~dxu/handouts/cs206/lab10` and load these into Eclipse. Run the tests and you should see an `ArrayIndexOutOfBoundsException`.
5. The `ArrayIndexOutOfBoundsException` tells us that it is triggered in the `reallocate` method of `KWArrayListBuggy`. That means that we want to see what is going on inside that method. To do this, we set a breakpoint, which tells Eclipse to pause execution of our program at a certain point. To set a break point, double-click in the left margin of the editor window at the line where you want execution to stop.

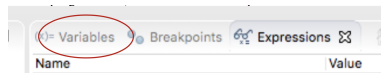



After double-clicking, it should look like this:


```
21      in
22      wh
23      {
```

The little blue dot is called a breakpoint. In our case, set the breakpoint on the first line of `reallocate`.

6. Now, click to debug your program by pressing the “bug” button, to the left of the usual `run` button.
7. Your test will start running as normal. Then, Eclipse will ask you to “Confirm Perspective Switch”. Say yes.
8. Eclipse will reconfigure its views. When you see this display, it means you are in the middle of a debugging session.



9. In the top-right section, click on the “Variables” pane. You will see all your variables. For example, you should see that `this` is a `KWArrayListBuggy`. If you click on the arrow to the left of `this`, you’ll see the instant variables of the class, informing you that `capacity` is 10, `size` is 10, and `theData` is an array.
10. To make your program move forward by one step, click the  Step Over button. (It’s in the normal toolbar toward the top of Eclipse.) You will see the highlighted line select the `for` loop. Keep stepping until you see the error.
11. Now that you have found the problem, abort debugging by clicking on the **Terminate** button (red square).

12. Return to the normal display by clicking  to choose the “Java Perspective” near the top-right corner of Eclipse
13. Update the code to fix the problem and re-run the test.
14. You will find several more bugs in this file. Fix them all.

A few more debugging tips:

- Next to “Step Over” you’ll find “Step Into”. The difference between these is that “Step Over” tries to get to the next line in the current method, while “Step Into” will move the highlighted line into any methods that are called on the current line. (If the current line does not contain a method call, “Step Over” and “Step Into” behave identically.)

- If you want the value of something that's not a plain variable (say, $i+5$), you can enter the expression in the Expressions pane, an alternative to Variables in the top-right section of Eclipse.
- When you have a list, Eclipse will allow you to see the list contents by clicking an arrow that will appear in the Variables pane.
- Eclipse supports conditional breakpoints, which trigger only when a certain condition holds. Visit the Breakpoints tab (next to Variables) to set this condition.

The best way to use the debugger is to make every time you click “Step Into” or “Step Over” a tiny scientific experiment. Before clicking, form a hypothesis about what you expect to happen. Then, after clicking, check whether your hypothesis is confirmed. If it's not, you've learned something new that might lead you to your bug.