CSci 121: Lab 1

"Tooling Up" - Getting ready for Serious Coding

Objectives

The purpose of this lab is twofold:

- To familiarize yourself with BlueJ -- an IDE (Integrated Development Environment).
- To begin to explore the utility of an *IDE*.

In future assignments, we will see how BlueJ provides many powerful tools to aid in debugging (we will only scratch the surface here).

You will also find that the programming exercises here are a bit more challenging than those you did in your CS120 labs (if you are already familiar with an IDE, there will be plenty left to do).

BlueJ is available on the LINUX or windows machines in the lab. We encourage you to use the facilities of the lab. You may wish to install BlueJ on your personal machine; just go to http://www.bluej.org/ for information.

Starting A Project

1. Open a web browser and navigate to the course webpage. (I won't repeat this in future labs; of course you will always start with this webpage to find links to lab instructions.)

2. Create a cs121/labs directory, using whatever way that is more comfortable for you.

With BlueJ

1. Start up BlueJ.

2. From the Project menu, select *"New Project*". Navigate to your cs121/labs directory and name the project "Lab1". Note that this has the effect of creating a new folder; you can check your "labs" folder to see the presence of the "Lab1" folder. You will be adding files to this folder. When I say "Lab1 folder", this is the folder I mean.

3. Download the file Factor.java to your Lab1 folder. Do this simply by right-clicking on its link at course website, choosing "Save as...", and navigating to your Lab1 folder to save the file.

4. This program is supposed to read in a number typed by the user and print the prime factors of the number. For example, if the user types in the number 15, the program outputs 3 and 5 (on separate lines). Just to make it easier to read, the file you imported is listed a few pages later in this document.

5. With the BlueJ project window in the foreground, select the Edit menu and "Add Class from File...". Select the file Factor.java to add it to the project. A box representing the class should appear in the project window. Note that you can move the box around inside the project window, which may not appear very useful at this point. It will be later on. (This is a trivial example of a *UML diagram*; we will discuss such diagrams briefly in cs121.)

6. Take a look at the code by double-clicking on the Factor box in the project window. Note that the box representing the class is cross-hatched; that's because it hasn't been compiled yet.

7. Fill in your name and the other indicated information in the documentation at the head of the file, and save it ("File/Save").

CSci 121: Lab 1

```
 import java.util.*;

/*****************************************************************
Factor.java: My first CS121

Program Name: Login:
Course: CS 121
Section:
Date:
Description: This program reads in an integer from the user and
prints out all the factors of the number. It is assumed that the
number is positive.
*****************************************************************/
public class Factor {
/*****************************************************************
 This is the main method that gets the number from the user and then
 invokes the method for factoring.
*****************************************************************/
    public static void main(String args[]) {
      int numberInput
      Scanner keyboard = new Scanner(System.in);
      System.out.println("This program will break up a number " +
                         "into its prime factors.");
      System.out.print("Enter the number: ");
      numberInput = keyboard.nextInt();
      listFactors(numberInput);
    }//main
/*****************************************************************
Method: listFactors
Precondition:   The parameter num is a positive integer.
 Postcondition: The factors of num are listed, one per line.
*****************************************************************
    public static void listFactors (int num)
    {
      int divisor = 2;
      System.out.println("The factors are:");
      while (divisor < num); {
        while (divisor % num == 0) {
          num = num / divisor;
          System.out.println(divisor);
        }  //  while
        divisor++
      }     //  while
    }       //  listFactors
}           //  Factor
```

CSci 121: Lab 1

Syntax Errors

1. To compile all files in a project in BlueJ, all you need to do is click on the "Compile" (in BlueJ) in the project window. This compiles all java files in the source folder (only one in this case).

2. BLUEJ: Unless you happened to notice and fix the syntax errors that are present, this program won't compile. In your program I thoughtfully included three syntax errors. As you should know, it is best only to pay attention to the **first** error, correct that one, recompile and go on to the next one. Fortunately, BlueJ knows this as well. It always highlights the line of your program at which it thinks the first error occurred, and tries to explain it. Plus, note that when you get the message, there is a question mark next to it. Clicking on the question mark will give you more elaborate hints as to the origin of the error. Do this for all errors that come up.

3. Fix the syntax errors one by one. Given your experience, it should not be hard to track them down.

Running Programs, Semantic Errors and Interactive Debugging

1. Running the file is almost as simple, but we have to brace ourselves because there is a logic error in this code. Before running the program, call up the "debugger" window. You do that by selecting "View/Show Debugger" (or alternatively hit control-D). Observe the position of the "Terminate" button, the one with the big "X" on it. You'll need it in a moment.

2. Position the cursor over the "Factor" box. Right click on it; you will get a pop-up menu. Select the choice "void main (String [] args)". What this does is run the main method. You will get another "Method Call" window asking for the "arguments" of the main method. Just click the "Okay" button of that window, or hit return. The program will then run in a console window that pops up automatically.

3. The program will prompt you for a number. Type one in, and hit return. What happens? Do you see why? Hint: to stop an infinite loop, press the "Terminate" button in the debugger window.
   Clearly, there are **semantic** errors in your program as well.

4. Maybe you can spot the reason for the infinite loop. (Hint: The fix just requires deleting *one* symbol.) But even if you do, imagine that you don't. Here's how we can see the infinite loop in action.

CSci 121: Lab 1

5. "Seeing a loop in action" is somewhat inconvenient without a debugger. But in BlueJ, you can stop a running method in its tracks and observe the contents of the parameters, local variables, and instance variables involved. The first step in this process is to set a *breakpoint*. This is a statement in the program at which the system will always come to a halt to allow you to see the values of the variables. Setting a breakpoint in BlueJ is very easy. Note that the source window has a margin on the left. By placing the mouse in that margin and clicking, you can position a little "Stop sign" next to any statement you want (to get rid of the stop sign, click again). Try it. In particular, position one stop sign next to the statement,
   while (divisor < num);
6. Run Factor again. When the program reaches the breakpoint, it stops and a debugger window appears. In addition, an arrow in the code shows you exactly where the program has stopped. Look carefully for the local variable names divisor and num in the depbugger window and note their values. Take a couple of moments to just look and understand what is happening at this point in the program.
7. Hit the Continue button in the debugger window. This causes the program to run full speed to the next breakpoint (or to the end of the program, if there aren't any). Look in the debugger window. What's happening to divisor?
8. You can also step through the program one statement at a time by hitting the "Step" button. That won't be extremely useful in this case, but try it anyway. Run Factor yet again, but this time after it stops at the breakpoint, hit the "Step" button (the second from the left). Pay careful attention to what happens to the variables.
9. By now it should be apparent where the error is. Fix it, and get rid of that infinite loop.
10. Now that the program is behaving more reasonably, type in a number and look at the answer (if there is indeed an answer at all). Try the number 36, for example. And try 15 and 105. Something is still wrong, isn't it? Evidently there are some more semantic errors.
11. Fix them! I won't be too explicit as to how you go about this, except to use the debugger (setting breakpoints and observing the values of variables) to help. But here are some hints. The idea underlying the algorithm is to try out all divisors from 2 up to the number in question. If any of these divisors divide the number (for example, we know that 3 divides 15), then that divisor is printed out, and the number is divided by the divisor. In fact, the divisor might go into the number more than once (think of 36, which has two factors of 2), so the inner while loop keeps dividing the number by the same divisor until it will go no longer. Then we go on to the next divisor (that's the outer loop). Somehow, the code is not expressing this idea properly.
12. It's not too hard to see what leads to the problem of 36 giving the wrong answer. But before concluding that all is well and good, don't forget to also try testing your code on 15 and 105 (these "benchmarks" test a particular feature of the algorithm).

Congratulations! You have just created, run, and debugged a Java program with an IDE.

Finishing Up

Electronically submit your code with all the bugs fixed, and a text file with the results for a few test cases.

If you do not finish the exercises in this session, all materials you produce for this lab are due by the beginning of next week's lab session.

Save all the work you do for this lab and all subsequent labs!