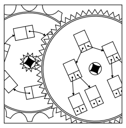
# Laboratory 1: A First Experience with Java

**Objective.** To write a first program within the Java environment.



Intro

**Discussion.** Learning Java is like learning a musical instrument: it requires practice, practice, practice. Solving problems in the text, and stepping through labs will help you gain experience programming.

In this lab we'll start the Java programming environment and enter a pre-written program. Even though we don't yet know many features of Java, the program can be easily understood with a little effort. When we're finished entering the program, we'll make some changes and observe their effects on the running program. When we're finished, we will have learned how to start and stop the programming environment, how to enter programs, and how to make them run.

**Procedure.** The goal of this program is to start the programming environment, enter a program, make it run, save the program, and exit the environment. We will assume that you have access to a programming environment and its documentation.

We first need to enter the following program:

```
// Lab 1: Experimenting with console and drawing windows.

// This next line incorporates all the features of Java Elements
// including the console window and drawing window.
import element.*;

public class Intro // this name should also be the name of the file
{
    public static void main(String args[])
    {
        // the following lines "declare" the resources we need
        DrawingWindow d = new DrawingWindow();
        ConsoleWindow c = new ConsoleWindow();
        Pt p;

        // we now ask the user to press the mouse, and wait
        c.out.println("Press the mouse in the drawing window.");
        p = d.awaitMousePress();

        // the mouse is now pressed, and p is the point of press
        // we draw that point on the screen
        d.draw(p);
        d.draw(new Circle(p,10));
        p = d.awaitMouseRelease();

        // the following two lines print as one.  Why?
        c.out.print("Thanks!  You pressed at ");
        c.out.println(p);
```

```
            // in general, your programs may need these lines
            // to make the system pause:
            c.out.println("Press the mouse to stop the program.");
            d.awaitMousePress();

            // this statement abruptly "kills" the program.
            // usually we will let the program finish more naturally.
            System.exit(0);
        }
    }
```

Please make sure you enter the program *exactly* as you see it here. Those statements that begin with double slashes are comments—they're important, but if you make changes to them it will not change the way the program runs.

This program, you will observe, waits for the mouse to be pressed and released. The location of the mouse press is drawn on the screen and is printed in the console window. That's it!

1. You should make or *compile* your program. During this process of translating Java into a computer-readable representation, the computer checks to make sure that your *syntax* or program structure is correct. If any errors are reported, you should compare the code above with the code you have typed into the computer. If there are differences near the point of error, they should be fixed and the program should be recompiled. If you don't understand an error, relax. Many error messages in Java take a little experience to understand. Scan your program a little more carefully, and then, if necessary, seek assistance.

   If your program had no errors, you are lucky indeed!

2. Once the program is compiled, it can be *executed* or *run*. Tell your environment to run the program. Two windows should appear. You should experiment with dragging the windows about, and resizing the windows. Can you place the console and drawing windows side-by-side?

3. Carefully noting where you're pressing, press the mouse in the drawing window. The exact coordinates of this point will be printed in the console window. The point is drawn and circled so you can see the association between the point and its coordinates.

4. Once the program runs correctly, you should consider the thought questions, below. Some of the questions may cause your program to break. Don't worry! Retyping in the original code should make things work again.

5. At some point in this process, print your program. Save the printout: it represents an interesting moment in your life you may wish to reflect on later!

6. Save your program and leave the environment. At this point, you might start the environment up again. Can you get your saved program to run again?

**Thought questions.** Consider the following questions as you complete the lab:

1. What happens when you press the mouse at one point and release it at another: which point is the one circled? How do you think you might change your program to circle the other point?
2. What is the difference between `c.out.print` and `c.out.println`? What happens when you change the `print` statement to read `println`.
3. The `System.exit(0);` statement essentially caused your program to kill itself. What happens if you remove the statement? What happens if you insert it in the middle of your program?
4. How do you think you could change the radius of the circle? Experiment with your program to see if your assumptions are correct.
5. What happens if you put the comment characters before a valid statement? How might this be useful?
6. Have you told your family you're a programmer? (You should!)