



**Computer Science Department/College of Engineering  
and Computer Science**

**CSc 20: Programming Concepts and Methodology II**

**Lab 1– Reviewing jGrasp environment and its  
Debugging**

## **Objective:**

This lab is to introduce you to a few important debugging features of jGRASP.

## **Overview:**

Most modern Integrated Development Environments (IDEs) , including jGRASP and Eclipse, have features that permit you to more easily debug your programs. These debugging features are much more powerful (and useful) than the brute-force approach of using `println()` to output to the console the value of a variable. One useful feature of a debugger is the ability to set breakpoints. When a program encounters a breakpoint, the execution of the program is temporarily suspended, and you have the option to inspect the value of any variable that has scope at that point of the program.

For this part of the lab:

- Start jGRASP.
- Download the files `courseDemo.java`, `course.Java`, `Instructor.java`, `Textbook.java` from the SacCT.
- Compile and run the programs.

The debugging features of jGRASP that you'll learn are the step-in and auto-step features that permit you to inspect variable values while a program is executing. If you are familiar with a debugger, it should take you a few moments to complete this part of the lab.

1. To set a break point in jGRASP, position the mouse pointer and left-click the mouse on the left border of the edit window. If done successfully, a red dot will be displayed (Figure 1). A break point is that part of the code where execution will be halted. You can set multiple break points.


```

public class CourseDemo
{
    public static void main(String[] args)
    {
        // Create an Instructor object.
        Instructor myInstructor =
            new Instructor("John", "Doe", "ABC010");

        // Create a TextBook object.
        TextBook myTextBook =
            new TextBook("Building Java Programs",
                "Stuart Reges", "Pearson");
    }
}

```

Figure 1: Setting a break point in jGRASP

2. Execute the program in debug mode, by clicking on the “debug” icon: 

3. The program will execute, and halt when it reaches the breakpoint. At that point, the debugger buttons (which previously were disabled) will become available and enabled (Figure 2).

```

public static void main(String[] args)
{
    // Create an Instructor object.
    Instructor myInstructor =
        new Instructor("John", "Doe", "ABC010");
}

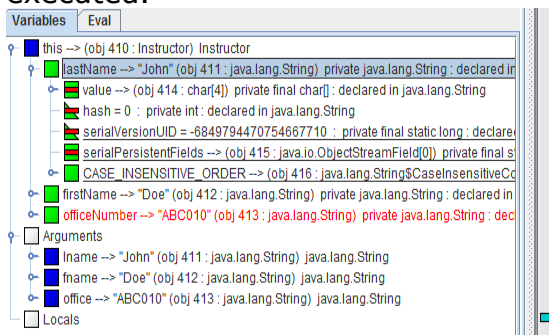
```

Figure 2: jGRASP's debugger features

4. Use the “step in” feature, which you can invoke using the “step in” button:



Click the button, and you'll see the blue, right-pointing arrow in the editor panel designate that part of the code that is being currently executed. Expand the “Variables” window, to see the values of all of the variables that currently have scope. The variable name that is displayed in red is the variable that is currently being updated (Figure 3). Using this feature you can much more easily debug your program. Now you can inspect the execution of for-loops, and easily determine whether a while and/or switch statement is executed.




```


@param lname The instructor's last name.
@param fname The instructor's first name.
@param office The office number.
*/

public Instructor(String lname, String fname,
    String office)
{
    lastName = lname;
    firstName = fname;
    officeNumber = office;
}

```

Figure 3: Inspecting variables that currently have scope

5. Compare the “step in” function with the regular “step” function: . Note the difference.

6. You can also “watch” the program execute line by line. Select the auto step button:  Set the delay to 1 second, and click on the “step-in” button to start watching the programs.

### **Activities:**

1. Copy instructor’s java programs (courseDemo.java, course.Java, Instructor.java, Textbook.java) from SacCT into your working directory.
2. Compile and debug your programs by running the main method using steps 1 to 6 above.

### **Deliverables:**

Write up ½ page (in MS Doc or PDF). (1) Explain the results related to these steps 1 through 6 above. Upload your document to SacCT. (2) Name the debugger function(s) you are using for each step.