Programming Project 0

17-654

The goals of this assignment are for you to:

1. Install Eclipse
2. Install the Fluid/Crystal plugin
3. Configure Eclipse so that you can write and debug an extremely simple "analysis"—really a syntax tree walk.
4. Improve the minimal syntax tree walk so that it prints actual values for certain classes of nodes.

To demonstrate that you have accomplished these four tasks, you will hand in:

1. The Java file or files that you have written for parts three and four.
2. Sample output from running your tree walk on input provided as part of this assignment.

You may work on this programming project in pairs. Pair projects will be given a single grade. You are free to choose your own partner, subject to one constraint: the instructors reserve the right to assign pairs in the case that students lacking significant previous Java experience are unable to find a more experienced partner. Please be sensitive to this criterion as you pair up.

PART 0 (0 Points, but note that you'll have to accomplish this in order to continue to part 1!)

Install Eclipse on your computer. You want Eclipse version 3.0.1 for your operating system. Various Fluid group members use Windows, Mac OS X, and RedHat, so any of these OS choices should work fine for you. You will get the Eclipse install from one of the standard Eclipse mirror sites. Start at www.eclipse.org, and it should be easy to find from there.

Download and install the material necessary for this assignment. Look in the "Course Documents" area of the Blackboard web site for "Crystal plugins" and "Programming Project 0". You will want to download all the files in both areas.

Unzip the "Eclipse Update/Config Instructions" and follow them. Note that there are two .htm files in there; make sure you read them carefully and follow their steps in the correct order.

After installing the plugin, go into Eclipse and open the "Java Perspective." You will see a bunch of packages in the left-most pane of the Eclipse window. Navigate your way into the one named "demo-crystal." In the src subdirectory you will find package demo.crystal.analysis. In this package is a java file containing class "Proj0Analysis" (the file will be named Proj0Analysis.java, of course).

You will be modifying this file throughout this assignment. Feel free to add more files and classes to the demo.crystal.analysis package as needed. Just remember to turn them all in.

Inside class Proj0Analysis are three methods: minimalAnalysis, fullCreditAnalysis, and extraCreditAnalysis. Each of these is invoked once per compilation unit at analysis time. Inside each of these methods are the lines:

```
// Your Code Goes Here
output.append("NYI\n");
// Your changes end here
```

When you work on the various parts of this programming assignment, you will be placing all the code you write in between these comments, in place of the

"output.append("NYI\n");

Your output should be appended to the StringBuffer that is an argument to each of these methods.

Your analysis will be run on a file similar to Project0Example.java, which can be downloaded from the "Programming Project 0" folder in the "Course Documents" area of the class Blackboard web site. The test file will include only syntactic elements found in Project0Example. This means that your code should work fine as long as you get the right results on Project0Example.

PART 1 (75 Points)

Write a simple visitor traversal that visits each node in the syntax tree rooted at "cu". On entry to each node, the visitor should print out either:

1. The "name" of the node, for all nodes that are **INamedNode**s, or
2. The class of the node, for all other nodes.

Each time the visitor descends a level in the tree, the output should be indented by two spaces. This indentation should be undone after all children of a node have been visited.

Sample output for a correct traversal of Project0Example.java looks like the contents of file MinimalExampleResults.txt on the class web site.

Note that the installation and configuration in Part 0 is the most important part of this assignment. As a result, code that does a traversal producing output that does not perfectly match the sample results will still get nearly full credit on this part.

HINT: The Crystal interfaces include a Visitor class. Maybe you should consider using it somehow…

PART 2 (25 Points)

For this problem you will enhance your visitor traversal to print more detailed information for some syntax tree nodes. The changes you should make are:

1. Print "{\n" on entry to an **IBlock**, and "}\n" on exit from it.
2. Print the actual value of float and int literals.
3. Enhance the traversal of **IField** nodes to print their modifiers, Java type binding, name and initialization expression (if any). Note that this will require partial override of the normal traversal of these nodes.
4. **DO NOT** enhance the traversal of any nodes not listed here. So, for example, printing of the initialization expressions for Example1.res and example2.t will still look like it did for part 1.

Sample output for the correct traversal of Project0Example.java looks like the contents of file FullExampleResults.txt on the class web site.

If you get this far, you have completed the programming part of the assignment.

EXTRA CREDIT (10 Points)

Those desirous of extra credit can gain karma by enhancing their traversal still more. For this problem, you will enhance your visitor traversal to print more detailed information about **`IMethod`** nodes. In particular, you should:

1. Print the modifiers for all methods
2. Print the return type of each method
3. Print the fully qualified name of each method
4. print a "(" followed by traversing the arguments of the method, followed by ")" and then the body of the method.
5. Indentation remains as before.

In order to accomplish this you will have to fully replace the traversal of IMethod nodes with a custom traversal.

Sample output for the correct traversal of Project0Example.java looks like the contents of file ExtraCreditExampleResults.txt on the class web site.

Notes on all 3 parts:

- For parts 1 and 2, you should match the white-space shown in the example output exactly.
- For the extra credit problem only, you will not be graded on matching the exact white-space shown in the example results. Feel free to produce better or prettier output if you like. To get the 10 points extra credit you must do at least as well as specified, but better is permitted.
- The first line of the example output includes the path to the Compilation Unit—`Users/dfsuther/Documents/School/Eclipse/crystal-workspace/crystal-rt/Project0/edu/cmu/isri/six54/Project0Example.java` when I ran it on my computer. The exact path shown in your output will vary depending on where you place your workspace on your computer.
- Note that the analysis driver in demo-crystal will call all three versions of the analysis on each compilation unit before moving on to the next one. You should share code within your implementation as much as makes sense. In particular, it would be wise to re-use the code for non-enhanced node traversals.
- To hand in your homework, copy the debug output from the Eclipse "Console" pane into a clearly identified file. Put any Java files you have written or modified[1] into the same directory. Zip the directory up, and hand it in via Blackboard.

[1] Yes, this does mean that you should hand in the modified version of Proj0Analysis.java along with any files you write yourselves.