Semester Project – Part 3 – CSE244 – Spring 2001

Due: Monday, May 7, 2001, 12:00 noon

April 10, 2001

The final part of the project for the Spring 2001 Semester will focus on developing a syntax-director translator that can take a GUI specification as input and generate all of the appropriate Java files for implementing the graphics related to the user interface. However, there is no functionality associated with the GUI. All that is available is the interface (e.g., buttons, options, menus, dialog boxes, etc.); actions by the user only display a particular feature or aspect of the GUI. This final part of the project is divided into three tasks:

1. Development of an approach for syntax-directed translation from the GUI specification language to Java source code, as part of the yacc/pcyacc grammar specification. In this task, you need to identify those portions of the grammar that contain the relevant information that can be extracted for assisting in the code-generation process.

2. Integrate and test your combined lex/yacc (pclex/pcyacc) translator. The redefinition of the parsing stack is required and should help to facilitate your efforts and translation.

3. Compile your generated source code using javac and execute your program using java to demonstrate your prototype GUI.

To support the final part of the project, we eliminate the center mouse button, since most of the class will be utilizing the PC.

Now, let’s review the important details for the project. First, a number of files are available for your inspection and use on the course web page:

- guif.pgm - a sample GUI specification
- guif.l - the lex/pclex specification
- guif.y - a yacc/pcyacc specification
- projp3.ps/projp3.pdf - PS/PDF Versions of this file
- awtmenu.txt - sample Java AWT menu bar code
- swingmenu.txt - sample Java Swing menu bar code

The lex and yacc specification should compile and execute without errors on the sample program.

Don’t forget that you must thoroughly integrate and test your GUI syntax-directed translator. In addition to making sure that your translator runs on the sample (guif.pgm), you must also conduct tests using your own sample and other programs to test various combinations. Once your translator is able to generate code, you can then compile and execute the “code” for the GUI via javac and java, respectively.
Turn in a hard copy of the project by Monday, May 7, 12:00 noon, to the main CSE office (4th floor UTEB). For the final project, document your efforts as follows:

- Overview of your completed project and its documentation (2 to 3 pages).
- Design/Implementation approach for generating the java source code. Make sure you also indicate how the parsing stack has been redefined!
- Testing plan and approach, test cases and test results.
- Lex (guiho.l) and yacc (guiho.y) specifications, and any other files that are required to support your code generation.
- Instructions for compiling and running your GUI syntax-directed translator.

DO NOT, I repeat, DO NOT, hand in lex.yy.c/yylex.c or y.tab.c/yystab.c!!! The documentation must be word-processed! Submit an electronic version of your files (zipped) by Monday, May 7, at 12 noon, with the file named: “person1_person2proj3.zip”, e.g., “demurjian_ajamianproj3.zip”.

A Translation Example

The remainder of the project description gives an example to help you understand the techniques that can be utilized to facilitate the syntax-directed translation from the GUI program specification to Java source code. We focus on the menu panel (menu bar in Java) from the sample program, which consists of various pulldown menus, File, Environ, Shading, Font, and Size. For the menu panel (bar), you would need to generate the Java AWT code segment (with many lines missing):

```java
public class MenuWindow extends Frame {
    public MenuWindow() {
        MenuBar mb;
        mb = new MenuBar();
        setMenuBar(mb);
        Menu m1 = new Menu("File");
        mb.add(m1);
        Menu m2 = new Menu("Environ");
        mb.add(m2);
        Menu m3 = new Menu("Shading");
        mb.add(m3);
        Menu m4 = new Menu("Font");
        mb.add(m4);
        Menu m5 = new Menu("Size");
        mb.add(m5);
    }
}
```

In order to realize this generated code as part of a syntax directed translation via yacc, the parsing stack can be redefined to store strings of characters. Moreover, by being clever regarding output files, it is possible to open an output file for each of the critical code segments that must be generated, which can later be combined to create the “complete” Java program. To illustrate this, the next page contains sample code for creating the menu panel (bar) given in the above Java code.
%union {char trans[100]; int XYZ;}

%type <trans> menu_item word

...etc...

opt_menu_panel : T_PANEL T_ITEMS T_ARE
{
    fp = fopen("menupanel.java","w");
    fprintf(fp, "%n\t\t");
    fprintf(fp, "\n public class MenuWindow extends Frame {{\n\n    fprintf(fp, "\n\t private MenuWindow() {");
    fprintf(fp, "\n\t \t MenuBar mb;"");
    fprintf(fp, "\n\t \t mb = new MenuBar(); ";
}
menu_item_list T_SEMIC
{
    fprintf(fp, "\n\t}"");
    fprintf(fp, "\n\t}"");
    }
| epsilon
;

menu_item_list : menu_item_list T_COMMA menu_item
{
    num_items++;
    }
| menu_item
{|// num_items is a global integer counter initialized to 1
    num_items++;
    }
;

menu_item : T_LPAREN T_LPAREN word
{
    fprintf(fp, "\n\t \t Menu m%d = new Menu("%s")", num_items, $3);
    fprintf(fp, "\n\t \t mb.add(m%d)\", num_items);
}
mouse_action T_RPAREN pull_down_list T_RPAREN
;

word : T_WORD {strcpy($$, yytext);}
;

Notice that menu_item is called for each item in the list. After all items have been processed, the file menupanel.java contains the required source code for the menu panel. Sample source code from The Java Tutorial, both the second and third editions, for AWT and Swing menu bars can be found on the course web page.