Prof. Dr. Peter Thiemann
Matthias Keil Winter Semester 2016/2017

Compiler Construction

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2016ws/

Exercise Sheet 1

1 Visitor pattern (2+2+2 Points)

In the lecture, a simple straight-line programming language was introduced (cf. slides 28–30). Assume that programs to be interpreted are given in abstract syntax as described by the presented implementation.

Example The program `a := (print(3),5)` is represented as

```
Stm prog = new AssignStm("a",new EseqExp(new PrintStm(
new LastExpList(new NumExp(3))),new NumExp(5)));
```

1. Implement a visitor which gives the maximum number of arguments of any print statement within any subexpression of a given statement. Extend the given abstract syntax classes with appropriate methods.
2. Implement a pretty-printing visitor for the straight-line programming language.
3. Implement an interpreter for programs in the straight-line programming language. Remember that expression sequences are evaluated from left to right. Interpreting expressions is more complicated than interpreting statements as expressions return a value *and* have side effects. *Hint:* The interpreter can also be implemented with the visitor pattern.

Hint: The course page contains a ZIP file with an Eclipse project skeleton for this exercise.

2 Tool dry-run (0 Points)

Visit the exercise page[1]. There you will find SableCC and the Library Package. The Standard Project Template for exercise 3 contains an Ant build script which you can use to generate Java code for the parser, compile and run your Java code and build a submission Jar. Check that the Standard Project Template works with your version of Eclipse, and make yourself familiar with the build targets.

[1] http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2016ws/exercises/

3 Shopping list (3+1 Points)

You have to lex and parse a shopping list. Each line of the shopping list contains a shopping item. Each item consists of

- the amount (number of pieces),
- a description of the product, and
- the price per piece in Euro.

They are seperated by a comma.
Example:

```
12, eggs                    ,  0.20
3,  parcels of TOP spaghetti,  2.00
6,  bottles 7up             ,  0.90
2,  bottles of French wine  ,  12.50
```

1. Write a lexer and parser specification of shopping lists for the above format for SableCC.
2. Compute the total prize for all items on a (parsed) shopping list. Output the price in the correct format. For the above example, the total prize is `38.80`. Structure your code according to the visitor design pattern.
   Hint: SableCC already provides a class `DepthFirstAdapter` in the generated `analysis.*` package. Extending this class will save you a lot of work.

Test your implementation with the provided test cases.

Submission

- Deadline: **03.11.2016, 12:00 (noon)**. Late submissions will not be accepted.
- Submit your solution to the subversion repository. Your submission will consist of one folder (exercise1) which includes your solution.
- Your solution to exercise 1.1 must be sent as a .jar file named `interp-<your name>.jar`. It must (at least) contain the source code of the extended abstract syntax classes, a class `PrettyPrintVisitor`, a class `MaxArgsVisitor` and a class `Interpreter`.
- Your solution to exercise 1.3 must be sent as an executable .jar file named `shopping-<your name>.jar`[2]. When invoking your solution with

  ```
  java -jar shopping-<your name>.jar list.txt
  ```

  it should output the total price.
- Make sure that your .jar files always contain the *source files* of your program.

[2]Standard Project Template: edit project.properties