Università degli Studi di Pisa

Laurea Specialistica in Informatica

Laurea Specialistica in Informatica e Networking

Advanced Programming

Final Term Paper

Start Date: 2/07/2013
Submission deadline: 22/07/2013 (send a single PDF file to attardi@di.unipi.it)

Rules:

The paper must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- **to consult or setup an online forum, to request help of consultants in producing the paper**
- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone's code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

The paper must:

1. be in a single PDF file, formatted readably (**font size ≥ 10 pt** with suitable margins, single column), of **no more than 10 numbered pages**, including code: for each extra page one point will be subtracted from the score.
2. include the student name
3. provide the solution and the code for each exercise separately, referring to the code of other exercises if necessary. **Do not include in an exercise code only needed for a later exercise**.
4. cite references to literature or Web pages from where information was taken.

Introduction

In this project, you will develop a version of the DSL language DOT for drawing diagrams. Here is an example of a DOT program:

```
graph g {
 n1[label="Root"]; n2[label="Child"]; n3[label="Nephew"];
 n1 -- n2[style=dotted, label="Parent"] -- n3;
}
```

It defines a graph with three nodes (in the second line), connected by two edges (in the third line). Within square brackets are attributes of the preceding element.

Exercise 1

Design a set of classes to represent DOT diagrams.

Exercise 2

Implement a recursive descent parser for DOT, without using a parser generator. Ensure that each parser method returns a parse tree for the input it analyzes.

Exercise 3

Write a DOT to HTML5 generator, which, given a DOT diagram, will produce HTML5 code using canvas primitives to display the diagram. Provide the code generated for the example in the introduction.

Exercise 4

Extend the language to allow defining directed graphs, where edges are directional and display with arrow heads, like in this example:

```
digraph d {
 n1[label="Top"]; n2[label="Left"]; n3[label="Right"];
 n1 -> n2; n1 -> n3;
}
```

Provide an implementation by extending the classes developed in earlier exercises.

Exercise 5

Extend the generator, so that it adds code for moving interactively the elements of a diagram, using the canvas method `addEventListener()`. Provide the generated code.

Exercise 6

Describe the delegate event model used in a typical graphic framework. Illustrate with a simple example the benefits of using abstractions like delegates or inner classes in such frameworks. What is the difference between a delegate and a lexical closure?