Università degli Studi di Pisa

Laurea Specialistica in Informatica

Laurea Specialistica in Informatica e Networking

Advanced Programming

Final Term Paper

Start Date: 8/01/2013
Submission deadline: 28/01/2013 (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.
4. cite references to literature or Web pages from where information was taken.

Introduction

In this project, you will develop a DSL for game development, called GSL (Game Specification Language). Here is an example of the description of a monodimensional Pacman game (http://www.thepcmanwebsite.com/media/pacman_flash/), where Pacman is moved horizontally on a circular board (i.e. leaving on one side enters from the other) eating dots, controlled with the keyboard by the player, ghosts move at a faster speed but randomly change direction:

```
<Game>
   <State> <Start/> <Play/> <Win/> </Lose> </State>
   <Control>
      <Keyboard> <Left/> <Right/> <Escape> <Keyboard/>
   </Control>
   <Rule> <LoseLife/> <Rule>
   <Scene>
```

```
        <Transition> <MoveGhost/> <MovePacman/> </Transition>
        <Graphics> <UpdateBoard/> </Graphics>
        <Events>
           <PacmanDotCollision/>
           <PacmanGhostCollision/>
        </Events>
     </Scene>
  </Game>
```

The GSL is used to generate code for the game, by means of XSLT transformations, for example:

```
  <xsl:template match="/Game/State/Win">
    class Win extends GameState {
      public void enter(GameSession session, Object[] params) {
        session.setStatus("Win");
    }
  <xsl:template>

  <xsl:template match="/Game/Control/Keyboard/Left">
       if (keyDown(KeyEvent.VK_UP)) {
         String[] values = { "Left" };
         gameSession.fire("Keyboard", values);
       }
  <xsl:template>
```

Exercise 1

Design a set of classes to represent the syntax of GSL, a subset of XML without attributes and GSLT transformations, a minimal subset of XSLT.

Exercise 2

Implement a recursive descent parser for GSL and GSLT without using external libraries or parser generators. Split the parser into a lexical analyzer and a syntax analyzer, as presented in the slides of the course.

Exercise 3

Design a set of classes to represent a game, including one for each of the element in GSL, and provide methods that implement a generic game mechanism, for example initialization, event firing, state transition and rule applications.

Exercise 4

Write a minimal XSL transformer, which, given a GSL tree node and a set of XSLT transformations, will produce the result of the transformations applied to the GSL node. Notice that the tree node passed to the transformer might not be the root node but still the match condition may specify a full path from the root.

Exercise 5

Build a code generator that, given a GSL root node and a GSLT tree, applies the transformations to the GSL tree, or parts of it as suitable , in order to produce the code to run the game.

Exercise 6

Build a full Pacman game, completing the GSL and the GSLT of the introduction.

Exercise 7

Most programming languages provide libraries to deal with regular expressions. Some languages instead provide regular expression constructs. Describe in which category fit the most common programming and scripting languages. Discuss the benefits or drawbacks of each approach from the perspectives of language designers and programmers.