Advanced Programming

Final Term Paper

Start Date: 21/08/2012
Submission deadline: 11/09/2012 (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 Template Engine, to be used for generating code. A template is a string that may include the following notation:

`<%var>`	refers to a variable named `var`
`<%var.attr>`	refers to the attribute `attr` of object `var`
`<%if(test)>body</%if>`	includes `body` if `test` is true
`<%foreach(list)>body</%foreach>`	repeats `body` for all values in `list`. Within `body` the variable `item` will refer to the current item in the `list`.

A template implements the following interface:

```
class Template {
```

```
    static Template parse(String expr); // create a Template from the
given string
    String expand(Environment env);     // instanciate the template with
the values provided in Environment
}
class Environment  {
    void bind(String name, Value value); // associate a value to a name
    Value get(String name); // return the value associated to a name
}
```

Exercise 1

Define a proper class for `Value`, taking into account that also an `Environment` can be a possible `Value`.

Define subclasses of `Template` suitable to represent the various types of notation.

Exercise 2

Provide an implementation for the above classes. In particular for class `Template`, use the technique of recursive descent for method `parse()`, exploiting a suitable tokenizer, and use polymorphism in method `expand()`.

Assume that the character '`%`' cannot occur in the string anywhere else besides the above constructs.

Exercise 3

Use the template engine in the implementation of a Web Application Framework. A Framework is based on the use of `WebComponent` classes, which provide the following interface:

```
interface WebComponent {
  void Render(TextWriter output);
}
```

The Framework should be capable of generating a `WebComponent`, given an HTML page template and a base class.

For example, given the following (portion of) template:

```
<table>
    <%foreach(books)><tr><td><%item.author></td><td><%item.title></td></t
r></%foreach>
</table>
```

and the class `Book`:

```
public class Book {
    public Book(string author, string title) {
          this.author = author;
          this.title = title;
}
public string author;
public string title;

static Book[] books = new Book[] {
   new Book("Homer", "Odyssey"),
   new Book("Joice", "Ulysses")
};
```

it would generate a class `BookComponent` that derives from Book and implements the `WebComponent` interface.

Provide a code generator that will produce such component as just described.

Exercise 4

Show how to extend the template mechanism and possibly the code generator in order to allow nesting of iterators. In particular show how to handle

```
<table>
    <%foreach(books)><tr>
      <td>
```

```
          <ul><%foreach(item.autors)><li><%item></li></%foreach></ul>
        </td>
        <td><%item.title></td></tr></%foreach>
</table>
```

Assuming that class `Book` has an attribute:

```
List<String>    authors;
```

Show the code produced by the generator.

Exercise 5

Provide a variant of the template in the previous exercise that uses `<ul>` only when there is more than one author of a book.

Show the code produced by the generator.

Exercise 6

Discuss the various types of polymorphism and in particular compare the techniques used for implementing subtype polymorphism in C++, Java and C#.

In which of these three languages polymorphic methods can be virtual?