Università degli Studi di Pisa

Laurea Magistrale in Informatica

Laurea Magistrale in Informatica e Networking

Advanced Programming

Final Term Paper

Start Date: 10/01/2012
Submission deadline: 12/02/2012 (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 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 exercise if necessary.
4. cite references to literature or Web pages from where information was taken.

Introduction

Consider developing a framework for dealing with object containers and providing a query facility on them similar to LINQ (Language Integrated Query), called PAQL. The PAQL query language provides the following constructs:

`element`	for declaring a class of objects to be stored in containers
`container`	for declaring a class of containers
`query`	for declaring a query on the elements of a container

For example:

```
element Book {
   [KEY] int ID;
   String title;
   String author;
   double price;
}

container<Book> books;
```

```
query<Book>            bookQuery(books);
```

The declaration `element` corresponds to a class declaration whose fields are all `public`. The attribute `KEY` indicates that the field is to be considered a primary key.

`container<Book>`	defines a class whose elements are instances of class `Book`
`query<Book>`	defines a query object that returns a list of elements of a container satisfying the query

Implementing the PAQL involves a code generator that produces classes implementing the following interfaces:

```
interface IBook {
   Book(int ID, String title, String author, double price);
   int ID;
   String title;
   String author;
   double price;
}

interface IBooks {
   Books();
   Book get(int ID);
   void insert(Book p);
}

interface IQuery<T> {
   List<T> execute();       // performs the query and returns the
                            // list of results
}
```

The generated classes can be used in the following way:

```
Books books = new Books();
books.insert(new Book(1, "Iliad", "Homer", 9.99));
books.insert(new Book(2, "Odissey", "Homer", 10.99));
List<Book> allBooks = query<Book>(books).execute();
for (Book book : allBooks)
  System.out.printl(book);
```

Esercise 1

Write a LL(1) grammar for PAQL.

Esercise 2

Define suitable classes to represent declarations expressed in PAQL.

Esercise 3

Write a recursive descent parser for the PAQL that produces a representation of the input using the classes of Exercise 2.

Esercise 4

Write a code generator that produces classes and methods that implement the PAQL.

Esercise 5

Extend the PAQL providing the ability to specify condition, and extend parser and code generator by means of inheritance and polymorphism:

```
query<Book> bookQuery(books where CONDITION);
```

The clause `CONDITION` provides conditions that elements of the container must fulfill to be listed among the results. A clause is made as a conjunction of elementary clauses, which consists of comparisons between fields, for instance:

```
query<Book> homerBooks(books where author == "Homer" and price < 10.0);
```

Allow using variables in the clauses, like:

```
query<Book> homerBooks(books where author == "Homer" and price < x);
```

Consider the following possible relations: ==, <, <=, >, >=, !=.

Show the code generated for all the examples above.

Exercise 6

Ad a clause for ordering the results, for example:

```
query<Book> homerBooks(books where author == "Homer" and price < 10.0,
ordered title);
```

Show the code generated for the example.

Esercise 7

Explain the unification of arrays and objects in the object model of JavaScript. What is the difference between a prototype in JavaScript and a class in other languages?