Università degli Studi di Pisa

Laurea Specialistica in Informatica

Advanced Programming

Final Term Paper

Copyright © 2014, Giuseppe Attardi.
Only copies for strictly personal use, in order to prepare the submission, are allowed. Any other use is forbidden and will be persecuted.

Start Date: 22/08/2014
Submission deadline: 12/09/2014 (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

We will develop PAM, a DSL for generating music. For example:

```
inst flute {
    note 1 A5;
    note 1 A5;
    note 1 C5;
    note 1 0;
}
inst piano {
    loop 4 { note 1 C3; note 1 E3; note 1 G3; }
    loop 4 { chord 1 C3 E3 G3; }
}
```

The score is for two instruments (flute and piano), each consisting of a sequence of events. An event can be a single note, with duration and pitch (e.g. note 1 A5 represents a note of duration 1 beat and pitch A5, i.e. the note A on the 5th piano octave, pitch 0 represents silence); a chord, with duration and set of notes (e.g. chord 1 C3 E3 G3 has duration 1 beat and consists of 3 notes; or a loop, specifying a duration and a list of events.

Exercise 1

Design a hierarchy of classes to represent PAM scores.

Exercise 2

Write a recursive descent parser for analyzing PAM score and represent it with the classes of Exercise 1.

Exercise 3

Write a sequencer, that takes a PAM score and produces a sequence of simplified MIDI events. Such events consists of three numbers:

channel (0-15), On/Off (0-1), note (0-127)

The sequencer should return a generator, i.e. an object providing the following interface:

```
interface sequence {
    TimedMidiEvent next();
}
```

where `TimedMidiEvent` contains the time of the event and the `MidiEvent` itself. Null should be returned at the end of the sequence.
The sequencer should exploit subtype polymorphism whenever feasible.
Show the sequence produced for the score in the introduction.

Exercise 4

Implement a variant of the sequencer that produces a generator which computes the events on the fly, without expanding the whole sequence.

Exercise 5

Extend the previous solution allowing loops to be nested.

Exercise 6

Explain the issue of variance (covariance, contravariance and invariance) for generic types.
Show an example of code where assuming covariance for generic types would lead to an error.
Describe how bounded polymorphism addresses this issue and how it is addressed instead in C++.