École Polytechnique de Louvain (EPL)
Programming Paradigms (LSINF2335)
Course assignment

Context-Oriented Ruby

During the course you learn about the reflective capabilities of Smalltalk and Ruby. This makes it possible to have a broader view of reflection, and also to compare some of the advantages and disadvantages of one with respect to the other.

During the course you also learn about Context-Oriented Programming (COP) in Smalltalk, but alas, not in Ruby. The question is then, does Ruby provide the necessary reflective facilities to implement a COP? If so, how? In particular, Ruby has certain language primitives that Smalltalk does not: think for example of mixin modules, singleton methods, class-level macros, eval and certain hook methods. Do these permit to implement COP more straightforwardly or elegantly? This assignment aims at exploring these questions.

1 Assignment

1.1 Objective

The goal of this assignment is to develop a COP framework in Ruby offering functionality similar to the one seen during the theory and practical sessions. The COP framework should support at least:

1. First-class contexts.
2. Definition of context-specific behaviour
3. Activation of adapted behaviour according to context changes.
4. Composition of adaptations (i.e. a `proceed` mechanism).
5. The necessary abstractions to support different resolution policies for conflicting adaptations. As a particular instantiation, implement at least a policy based on context activation age.

These goals correspond 1-to-1 with the objectives of the practical sessions on COP for Smalltalk, as follows:

- Objective 1: COP Infrastructure
- Objective 2: COP Adaptation
- Objective 3: COP Adaptation
- Objective 4: COP Composition
- Objective 5: COP Resolution

You can use the practical sessions as a precise definition of the expected outcome for your COP framework in Ruby.

1.2 Approach

- To implement the COP framework, you should use the reflective facilities of Ruby.
- Although you are free to use the Smalltalk COP framework seen in the course as source of inspiration for your design and implementation, think carefully about alternative solutions that are possible in Ruby (since Ruby has language constructs that Smalltalk doesn't). Document your choices in the report.
- Describe in the report the main differences between your COP framework and the one implemented in Smalltalk.
- To validate your implementation, make sure you have testing code that can easily be executed, and that covers the functionality of your COP framework as thoroughly as possible.

2 General Guidelines

2.1 Organisation

- The assignment is to be carried out in **groups of two**. Working individually is also an option, though not particularly encouraged.
- It is recommended to work using pair programming, or any other agile technique in which both programmers are actively involved. In the end both will know the code equally well, and it will contain less bugs.
- Feel free to ask your course staff for help or hints if you are stuck.
- For simple questions, e-mail is fine. For complex questions, it is more efficient to see the instructor or assistant personally; ask for an appointment in this case.

2.2 Submission

- The deadline for submitting the assignment is **13 May 2016**.
- The complete solution should be uploaded as a single file to the Evaluation / Assignment section of the course in Moodle. If your submission comprises more than one file, put everything in a compressed archive.
- You can update your submission as many times as you want before the deadline.

2.3 Evaluation

Your mark for the course depends on your solution of the assignment (design, correctness, completeness, efficiency, readability, documentation and tests), the presentation and depth of the accompanying report, and a final oral presentation and exam.

Code

- The assignments should be submitted as documented source code —that is, source code plus comments, example code, unit tests, `README` files or anything else that is normally used to make a software project comprehensible and maintainable by a third party.
- There may be different possible solutions for the assignment, each with their advantages and disadvantages. You must carefully consider your options and implement the solution you think is best. It is important that you motivate your choices in the report.
- Your implementation should be as elegant (succinct, readable, simple) as possible.

Report

- Provide an overview of your solution, that serves as a gentle introduction to anyone willing to dive into the code. Do not assume that the reader knows the Smalltalk solution.
- Discuss what/how reflective features helped you in developing this assignment. Explain what language features were missing or could have been handy to make your solution even nicer.
- Illustrate the most important design and implementation choices of your solution through architectural diagrams and code fragments.
- The report should contain no more than 7 pages, including diagrams and listings.

Exam

During the exam session, every group will do an oral exam and presentation.

- The oral exam will last 30 minutes.
- The submitted assignment will serve as starting point for the exam:
  1. You will be asked to explain the solution in detail. Feel free to bring a portable computer to demonstrate your running code, present slides, etc. You get maximum 15 minutes for this presentation.
  2. You will be asked to discuss the advantages, disadvantages, possible improvements and alternatives to the provided solution, and possibly to compare it to the Smalltalk solution.
- The oral exam can be taken in English or French. The default is English. If you prefer to present the oral exam in French, please inform the teachers just before the start of your exam.