Lecture "Software Engineering SoSe 2023

Free University of Berlin, Institute of Computer Science, Software Engineering Group Lutz Prechelt, Linus Ververs, Oskar Besler, Tom-Hendrik Lübke, Nina Matthias

Exercise sheet 9 **OCL** to 2023-06-19

Task 9-1: Terms

a) What is meant by the term *"information hiding" in* the context of software development? Explain the principle and how it relates to the *"Need to Know"* principle.

b) Why is the principle of information hiding useful? Explain this with an example.

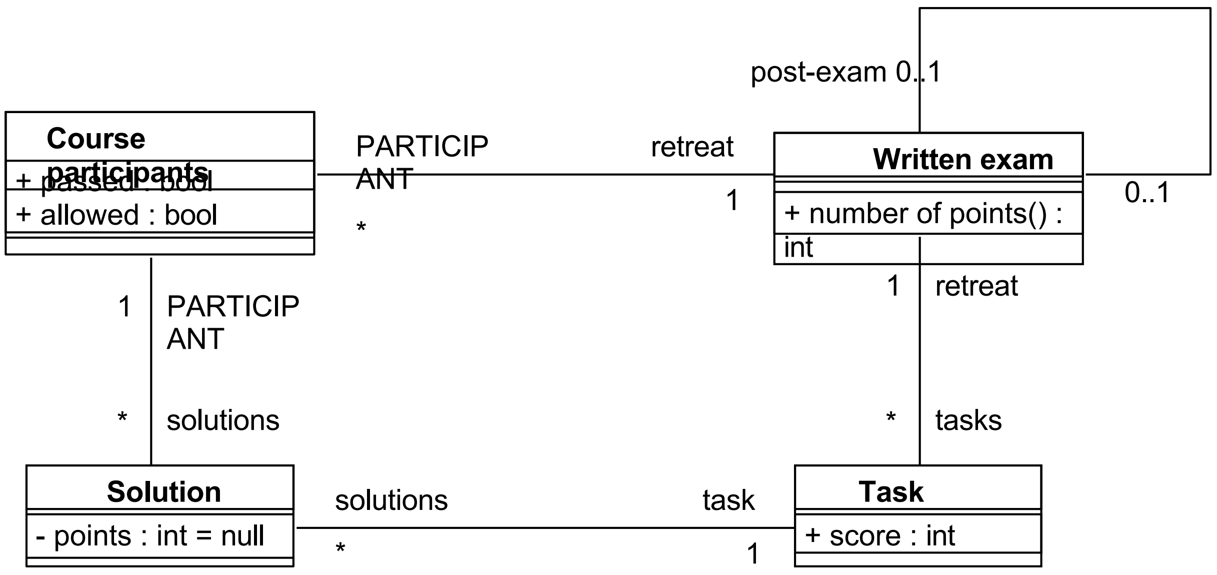
c) What is meant by the *"design by contract"* principle? In which context is it used and why?

d) Explain how the terms *OCL, invariant, design by con- tract, precondition, constraints, postcondition are* related.

Task 9-2: Read and write OCL

An examination management system is to be used to manage examination results and students' scores on solutions to individual tasks.

The following UML class diagram models part of the required data.



a) Based on the diagram, express the OCL constraints `c1` to `c3 in` natural language, i.e., in a form that you would use in a conversation with a person without a computer science background.

```
context task inv c1: score > 0

context student inv c2: solutions->size() = exam.tasks->size() context exam inv

c3:
    participant->forAll (t |
        t.passed implies t.solutions->exists (l | l.points
            > 0 and l.task.exam = self
        )
    )
```

b) Specify OCL constraints that formalize the following facts. Take care to formulate syntactically sound OCL expressions. This includes paying attention to the exact notations of classes, attributes, operations, and associations from the UML class diagram.

Also check the OCL 2.4 specification (`http://www.omg.org/spec/OCL/2.4/ PDF`) if you are missing expression means.

1. In every exam there is at least one task with exactly one point.
2. A post retreat cannot have a post retreat.
3. If a student is admitted, there is also a solution from him/her for each exam task.

Task 9-3Æ: OCL model extensions

This task is based on the same UML model as task **9-2**. Now, the exam correction is also to be modeled: During the correction, the instructor goes through all solutions individually, evaluates each of them and calculates the total score for each participant.

a) Now extend the model with a `Correct` operation with a suitable signature and a new attribute; beyond that, the class diagram must not be changed in any way (i.e. no new classes, changed visibilities, etc.). The `Correct` operation is called as soon as the instructor has corrected a solution.

In particular, this should make it possible to

- the attribute `points` in `solution` is filled, and
- the achieved total score of the participant is updated.

First, verbally describe what exactly `correct is` supposed to do. Find suitable places (and for the new attribute: also a suitable name) for the new members in the class diagram.

b) For now, assume in a first version that `correct` only once

may be called per copy of the class `solution`; the number of points once set is unchangeable thereafter.

Specify both the strictest possible preconditions for your operation `corri gate`, and all effects *(postcondition) of* the operation completely in OCL.

c) Now assume the more realistic requirement that `correct` can be called multiple times, e.g. to correct the score of a solution during an exam review.

Again, specify as strict preconditions as possible and all effects of the operation in OCL.

d) Specify the effect of the `exam.score` operation using the OCL keyword `result`.

Note: The resulting expression fits comfortably on one line. If needed, research OCL collections and their operations.