Exam-1 of Modelling & System Development 2010/2011
14-dec-2010, Educ-Beta, 11:00 - 13:00

Lecturers: Wishnu Prasetya

PART I (30 pt)

For each multiple choice question below, there is only one good answer. You get +3 pt for each good answer, 0 pt if you dont't answer, and -0.8 pt for a wrong answer. A negative total is rounded up to 0.

1. Here are a number of statements about the *Unified Process* (as meant by Arlow and Neustadt). Which of them is correct?

   (a) Unified Process is one of many approaches in software development process.

   (b) Unified Process is part of UML.

   (c) Unified Process is a software modelling approach.

   (d) Unified Process is an initiative by OMG to unify software modelling and development.

2. Which of these is NOT correct?

   (a) We do the Unified Process in *iterations* to make the process more flexible towards changes in requirement by the stakeholders/customers.

   (b) The Unified Process allows us to overlappingly schedule 'Design', 'Implementation', and 'Testing' workflows.

   (c) The Unified Process allows us to overlappingly schedule 'Inception' and 'Construction' phases.

   (d) In Unified Process, each iteration produces in principle a 'base line'; the next iteration improves this.

3. Which of these is NOT correct ?

   (a) In the 'Requirement' workflow we identify our use-cases; in the 'Analysis' workflow we identify key classes and their initial models; in the 'Design' workflow we refine our use-cases and class models.

   (b) A use-case captures *functional* requirement. Additionally we may also have *non-functional* requirements, which have to be formulated separately because usually they cannot be modelled by use-cases.

   (c) In the 'Design' workflow we replace each use-case with a class diagram, and the flows within each use-case with either association or aggregation.

   (d) It makes sense to put more effort in the 'Requirement' and 'Analysis' workflows (than the effort on 'Implementation' or 'Test' workflows) during the Inception phase.

4. An on-line weather service regularly collects data from weather stations. Using a web browser, a user can request a summary of the collected data (e.g. average temperature).

   We identify two use-cases: (i) collect data, and (ii) request weather summary. Which of the following is correct?

   (a) With respect to (i) and (ii), 'user' is a primary and secondary actor.

   (b) With respect to (i) 'user' is a primary actor; with respect to (ii) weather station is a primary actor.

   (c) With respect to (i) 'weather station' is a secondary actor; with respect to (ii) 'user' is a primary actor.

   (d) With respect to (i) 'user' is a secondary actor; with respect to (ii) 'user' is a primary actor.

5. What does the *main flow* of a use-case describe?

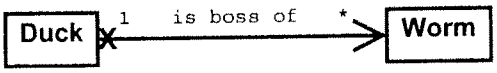
   (a) It describes the steps that have to be carried out in the use-case, and how they are sequenced.

   (b) It describes how objects flow within the use-case.

   (c) It describes how objects *and* states flow within the use-case.

   (d) It describes how the use-case is related to other use-cases in a use-case diagram.

6. Consider this class diagram:



   What do the 'navigation' cross and arrow in the above model tell us?

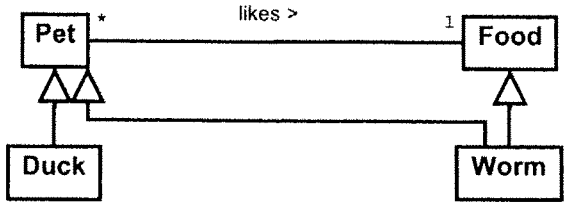
   (a) A duck can ask its worms to get their properties. A worm, on the other hand, cannot in principle ask the properties of its duck-boss.

   (b) It tells us that we are supposed to read the association as 'a duck is a boss of zero or more worms', and not the other way around.

   (c) It emphasizes the fact that in the above association a duck associated to possibly multiple worms.

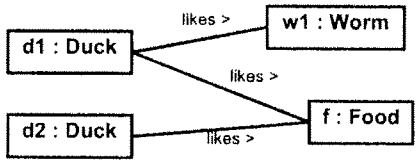
   (d) It is used to prevent a worm, in the above association, to become a boss of a duck.

7. Consider this class diagram:

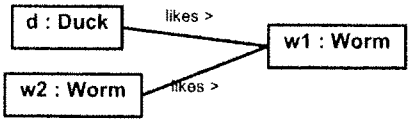


Which of the following is a valid instance of the model above?

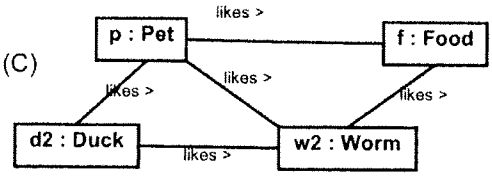
(A)



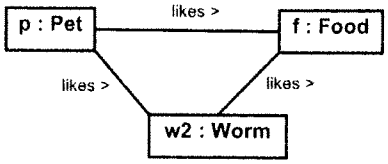
(B)



(C)

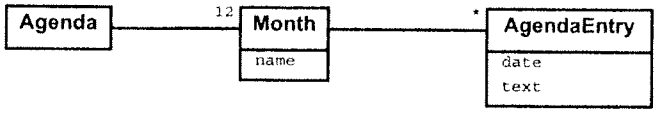


(D)

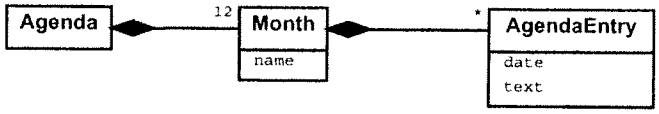


8. An agenda consists of 12 months, each consisting of agenda entries. Consider the following two alternatives to model this:

(i)



(ii)



Which of the following is correct?

(a) (i) does <u>not</u> allow two agendas to share the same agenda entry.

(b) (ii) allows two agendas to share the same agenda entry.

(c) in model (i), removing an agenda-entry from an agenda will <u>not</u> accidentally remove it from another agenda.

(d) in model (ii), removing an agenda-entry from an agenda will <u>not</u> accidentally remove it from another agenda.

9. What does Liskov's Substitution Principle advise us?

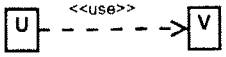
(a) It advises us to keep the classes that collaborate in a use-case consistent to the use-case.

(b) It advises us to keep a subclass (in particular when we override operations or add new operations) consistent with its superclasses.

(c) It advises us to keep every many-to-many association consistent on both its sides.

(d) It advises us to make the attributes of all our classes, when possible, private.

10. What does the following class diagram tell us? (U and V there are classes)



(a) U is a special subclass of V.
(b) It simply means that U somehow uses V.
(c) It means that U imports V.
(d) It implies that V is an interface, implemented by U.

PART II (70 pt)

1. Use-case modelling [20 pt]

Consider an Appointment System (AP). A patient can request the system to make an appointment with a doctor. The patient provides the time she wants and the name of the doctor. The system first asks the doctor if she is free at the given time. If so, the patient can confirm the request, and the system will reserve the time for her.

Up to 24 hours before the appointment time the patient or the doctor can cancel the appointment.

Any user of the system, including patients and doctors, must first register to the system. Only then can he access the system's other functionalities. To make appointment or to cancel, a registered user must first login.

Patients and doctors should be able to view their appointments.

Your task:

1. Make a Use-case Diagram modelling the above situation. It should include *all* use-cases and actors implied by the requirement above.
2. For each use-case, either give a self-explaining name or give a short description. Specify who are the primary and secondary actors of that use-case.

   You do NOT have to provide a full description of your use-cases. Just providing the things requested above is sufficient.

2. Class modelling [30 pt]

A company is organized in a number of departments. It has a director, and each department has its own manager. Each employee is assigned to at most two departments, except the director and managers. They are also employee, but a director is not assigned to any department. A manager is assigned to exactly one department.

Each employee has name, address, and salary. The director can add or remove an employee, or re-assign her to another department. The director can change the salary of an employee. A manager can change the address of any employee in her department.

Every month, the company pays the salaries to their employees. However, some employees are hired from external agencies. Their salaries are paid weekly.

Your task: provide a class diagram that models the above situation.

- Be as complete as possible in your modelling.
- Try to exploit inheritance. Be careful: don't let a subclass inherit things that do not belong to them.

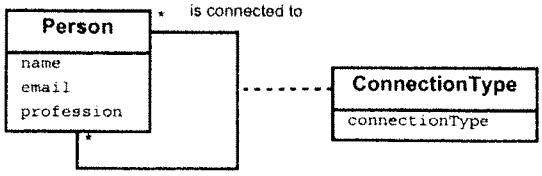
3. Refining model [10 pt]

Here are models of a social network, where people can add other people into their 'connections list'. The two models are the same; look at the (B) version if you find (A) a bit confusing.

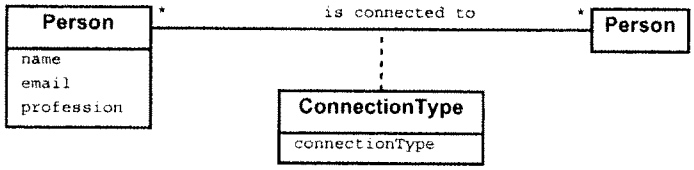
Notice that the association is many-to-many.

When two persons (e.g. Ann and Ben) are connected by the association below, we also specify the kind of connection, which is either a friendship kind of connection, or a professional kind of connection. This is specified by the association class `ConnectionType`.

(A)



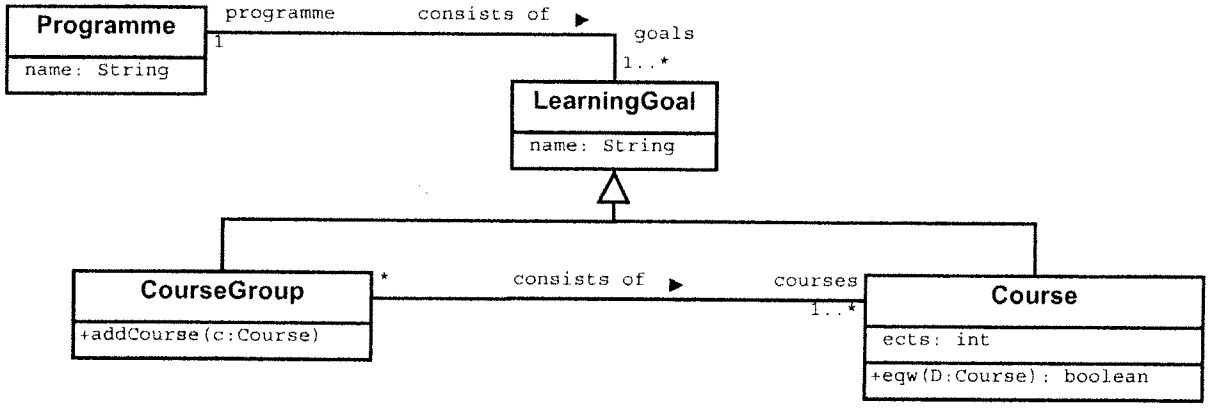
(B)



Your task: reify the model to get a model without an association class, and without many-to-many associations.

4. Specifying class [10 pt]

Consider the model below:



Notice that between CourseGroup and Course we have a many-to-many association, but any LearningGoal will belong to exactly one Programme.

Your task: express the following requirements in OCL:

1. The ects of every course should be positive, and at most 10.
2. The operation `eqw(D)`, when called on a course `C`, should check whether it has the same ects load as that of the course `D`.
3. A course group and the courses that belong to it should all belong to the same programme.
4. The operation `addCourse(c)` can only be invoked on a CourseGroup that does not already contain a course with the same name. After the operation, `c` should be a member of the CourseGroup's courses.