

**When:** Thursday, Oct. 28, 2004, in class (80 min).

**Format:** Open textbook (Object-Oriented Software Engineering: Using UML, Patterns and Java) and your own handwritten (not mechanically or electronically reproduced) notes. Closed everything else.

**Materials covered:** You are responsible for materials covered in lectures on UML, including OCL, and for what your group did for the assignments on requirement analysis and design.

**Overview:** Goals: increase software quality and productivity. Means: learn a modeling language and carry out a group project, and more specifically, go through all phases of the project, and capture all aspects of analysis and design precisely with UML.

## Sample Problems

1. What is UML? Name two advantages of using it.
2. What diagrams are for modeling structures? What diagrams are for modeling behaviors?
3. What diagrams are usually good for capturing requirements? What diagrams are usually good for capturing detailed design?
4. How do we make use of nouns in the requirement description?
5. What's the difference among association, aggregation, and composition?
6. What is a template class? Give an example.
7. How is a transition labeled with `event[guard]/action` taken?
8. Similar as above, you will need to know how to answer questions about other basic concepts, as listed below.
9. Consider developing a campus classroom facilities management system.

The campus has a collection of classrooms. Each classroom has a collection of equipments, which may include microphone, blackboard, overhead projector, TV, Internet connection, etc. Each equipment has information such as history, condition, available forms (built in or for picking up somewhere), as well as information specific to the equipment, such as size for blackboard and TV, mobile or not for microphone, and speed of connection to Internet.

Each classroom has a list of courses scheduled to take place in it, the instructor for each course, requested equipments and setups for each course, and a manager in charge of setting up the equipments.

The system should allow instructors to look up the list of equipments in a classroom, request equipments and setups, and query the setup status. The system should send request for equipments and setups in a classroom to the classroom manager. The manager can update equipment information and setup status. In case of an equipment can not be setup according to the request, the manager should notify the instructor as soon as possible. The system should also allow anyone to send comments to the manager.

Draw class diagrams for this system. Draw use case diagrams for this system.

10. Similar as above, you will need to know how to draw other UML diagrams from more detailed descriptions. You will also need to know how to write invariants and pre and post conditions in OCL from given descriptions.

11. Give three of the most important classes in your group project and show how they are related in a class diagram.
12. Similar as above, you will need to know how to draw other UML diagrams and write invariants and pre and post conditions in OCL for your group project.
13. How do you like the tool support for UML?

## Basic concepts

UML, OCL

use case diagrams

actor, use cases

use case text, main flow of events, exceptional flow of events

relationships: include, extend, generalization, initiate, participate

stereotypes, packages, notes

class diagrams, object diagrams

objects, classes, attributes, operations, responsibilities

attributes and operations: type, signature, visibility

relationships: association, generalization, dependency

association: role, multiplicity, aggregation, composition

abstract classes, interfaces, template classes

interaction diagrams: sequence diagrams, collaboration diagrams

sequence diagrams: object lifeline, focus of control,

create and destroy, call and return, branching and iteration

collaboration diagrams: vertices, links, message sequence number

state diagrams

events, states, transitions

states: initial, final, activities

transitions: source and target states, event[guard condition]/action

activity diagrams

activities, sequencing (transition)

branching (branch and merge, guard condition), concurrency (fork and join)

swimlane

deployment diagrams

components, nodes

constraints

contracts, invariants, pre and post conditions

collections: sets, sequences, bags

operations on collections: comprehension, quantifiers