Topic 3: Object-Oriented Design Process

Part 3: Design Techniques Ch2.6-2.12

Process Revisited

Software Process Phase	Design Technique Used
Analysis	Use Cases
Design	CRC Cards
Implementation	UML Class Diagrams

Use Case

… for Analysis

Software Development Phases

- Use Case
- An analysis technique for describing requirements
- Lists steps to complete a task

- Ex: Word processor use cases for
  - install the program
  - load document
  - print

UML: Unified Modeling Language

- Object-Oriented Paradigm modelling notation
- Clear and effective way to model many aspects of a software system using a commonly understood language
- Programming language independent
- Enables a variety or analysis and design techniques
- A subset of UML will be used in this course
  - Use Case Diagrams are used to model system functionality

Requirements Discovery: Use Cases

- Use-cases are a scenario based technique

- The set of use cases should describe all possible interactions with the system.
  - Does not show sequence of actions.

Use Case: Actor

- Entity outside the software system
  - interacts with the system
  - Operates on objects in the system but cannot be operated upon by objects in the system.

- Represents coherent role played by users

Use Case: Actor

- A user of software system may take on more than 1 role, usually at different times

- An actor may represent more than one user

Primary and Secondary Actors

- Primary Actors
  - Actors who initiate a scenario (use case) causing the system to achieve a goal

- Secondary Actors
  - Actors supporting the system so primary users goals can be completed (do not initiate the use case or scenario)

Use Case Guidelines

- Task must be of value to user

- Format
  - Can have a loose format to fit your needs.
- Content
  - Has a descriptive title/name.
  - Lists actor's interaction with program

- Should have variations to describe alternative cases.

- Should have preconditions whenever necessary

Example: Buy Goods

- Shoppers shall be able to place order on products displayed
- System shows the billing and shipping info for user to confirm.
- System sends info to billing and shipping.
- Generates invoice

Example: Buy Goods

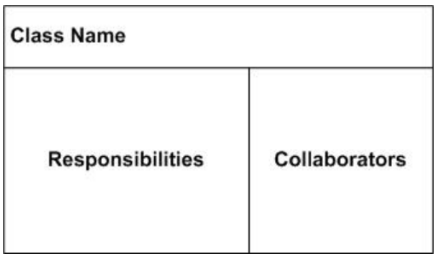
1. The user will indicate that she wants to order the items that have already been selected.
2. The system will present the billing and shipping information that is stored.
3. The user will confirm that the existing billing and shipping information.
4. The system will present the amount that the order will cost, including applicable taxes and shipping charges.
5. The user will confirm that the order information is accurate.
6. The system will request that the *billing system* should charge the user for the order.
7. The *billing system* will confirm that the charge has been placed for the order.
8. The system will submit the order to the *shipping system* for processing.
9. The *shipping system* will confirm that the order is being processed along with tracking ID.
10. The system will indicate to the user that the user has been charged for the order and present her with tracking ID.
11. The user will exit the system.

CRC Cards

… for Design

CRC Cards

- CRC cards is an index card listing of:
  - Class Name
  - Class Responsibilities
  - Class Collaborators



CRC Cards: Guidelines

- Purpose
  - Supports an informal design process
- Physical card support
  - Walk-through a use-case deciding which classes do which tasks
  - Lay cards out on table and re-arrange them as needed.
- Limit responsibility of the class
  - No "God" object
  - If too much on a card, split into two classes

CRC Cards: Process

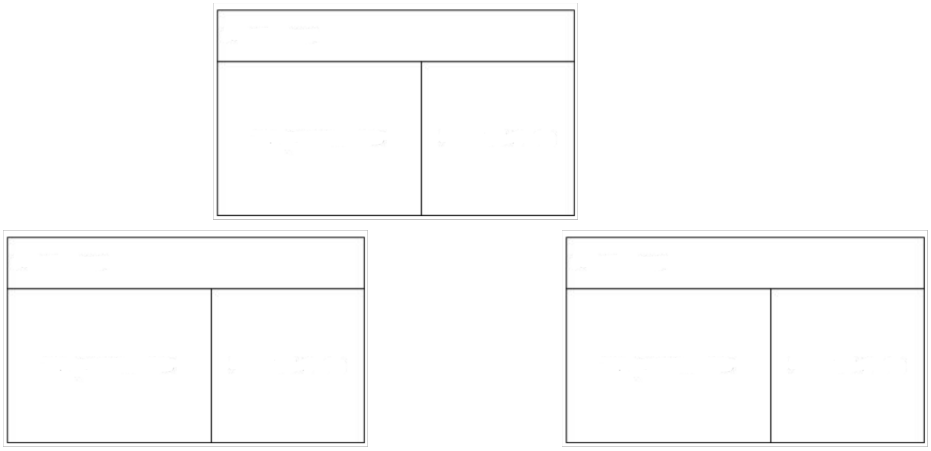
1. **Find classes**: For each discovered class, write its name on the top.
   - Look for nouns in use cases

2. **Find Responsibilities:**
   - Not method names (use high-level responsibilities)
   - Message class example:

3. **Define Collaborators:**
   - no particular order; which classes does this one use
   - does not line-up with responsibilities

4. **Move cards** until logical:
   - Don't list all details; just enough to show it can do its job

CRC Cards: Example



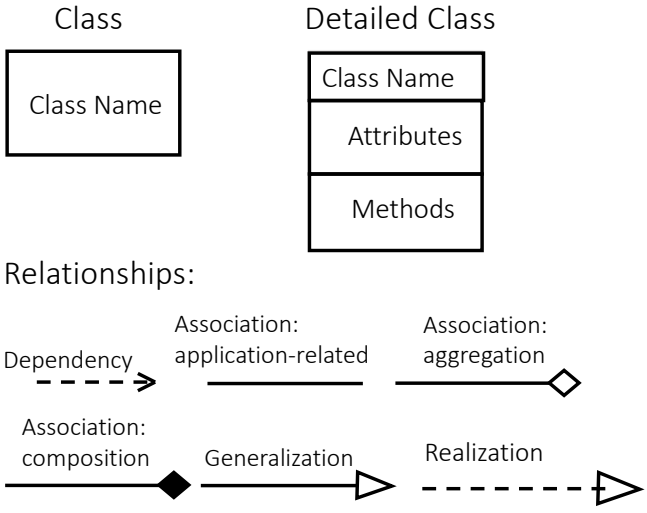
UML Class Diagram

… for Design Communication and Implementation

UML Class Diagram

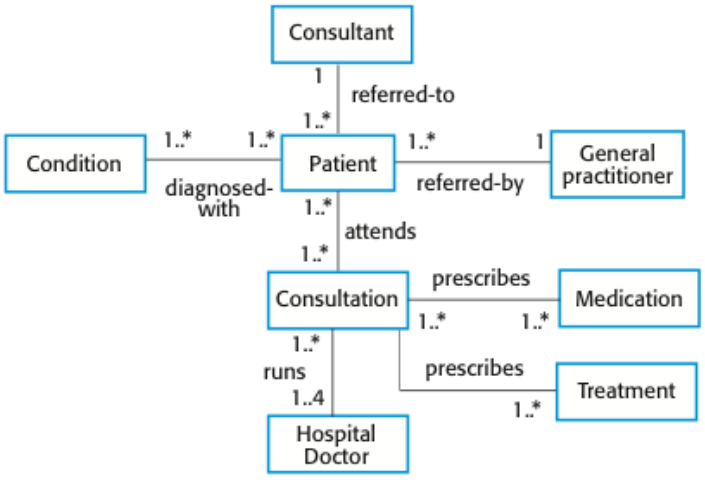
- A diagram showing classes and relationships between them
- Simplest form:
  - Class
  - Association
  - Multiplicity

UML Class Diagrams



Class Diagrams

- We often label the associations with its meaning
  - Helps understand what is being modelled



Class Diagrams

- Contains
  - Class Name
  - Attributes (fields)
  - Operations (methods)
- Types (optional)

- Visibility precedes attributes and methods
  - Public
  - Protected
  - Private

Class Diagrams

- Associations between classes:
  - Application specific (related)
  - Uni-directional (directed association)
  - Aggregation: “has a”
  - Composition: “composed of”

Class Diagrams

- Generalization: <specific> classes “are” <general> classes
- Realization: <implementation> classes “implements” <interface> classes

Class Diagrams

- Dependency: Class A depends on class B if A uses at least one feature of B, e.g., it accesses one of B's data fields or invokes one of its methods.

- Association Class: adds attributes, operations, and definitions to associations