ICS 121 Topic 10:

Object-Oriented Analysis and Design

Object Modeling Technique
Unified Modeling Language
Class Modeling
Dynamic Modeling

Design - Review

- Design transforms requirements into
  - an architecture diagram
    - subsystems, modules and their relationships
  - a detailed design
    - a specification of the abstract interface, data structures, and algorithms of each module
- Also develops
  - a review plan for ensuring the design meets the requirements
  - a test plan for ensuring the implementation meets the design

Object-Oriented Analysis and Design

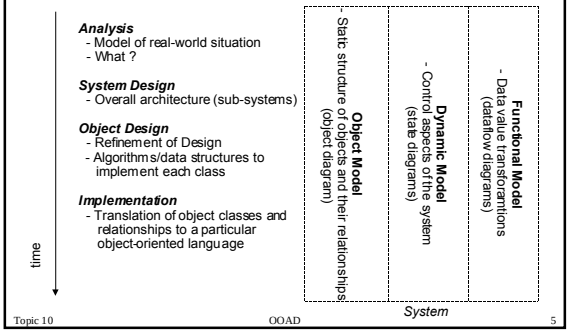
- Object-Oriented Analysis is the "requirements phase"
  - an alternative semi-formal technique focused on objects in real world
- Object-Oriented Design is the "design phase"
  - refine object definitions and their interactions until "implementable"

- Semi-formal technique
  - functional modeling
  - class modeling
  - dynamic modeling
- These steps focus on
  - data
  - actions
  - and their relationships
- Reuses familiar tools
  - E-R diagrams
  - Finite State Machines
  - Data flow diagrams
- Steps and diagrams
  - are typically performed in parallel after initial class definition
  - must be kept in synch

Object Modeling Technique (OMT)

- OMT [Rumbaugh et al.,1991] consists of
  - building three complementary models of the system
  - adding implementation details to the models
  - implementing the models
- OMT includes a set of
  - phases [processes]
  - diagramming techniques
- OMT has four phases
  - object-oriented analysis builds a real-world model
  - system design determines overall architecture of system
  - object design decides upon data structures and algorithms
  - implementation translates design into programming language

OMT Stages and Models



Unified Modeling Language

- The Three Amigos join forces to form a single UML
  - Grady Booch at Rational has his own technique
    - Object-Oriented Analysis and Design [Booch, 1994]
  - 1994: Jim Rumbaugh joins Rational
    - Object Modeling Technique [Rumbaugh, et.al. 1991]
    - UML is initiated as a combination of the models in OMT and OOAD
  - 1995: Ivar Jacobson joins Rational
    - Object-Oriented Software Engineering [Jacobson, et.al., 1992]
    - the use case model is added to UML

UML is just a common notation, however,
not a methodology
rather it can be used with a variety of methodologies

Object-Oriented Modeling

- Builds a real-world model from requirements
  - client interviews
  - domain knowledge
  - real-world experience
- The model addresses three aspects of objects
  - overall functionality
  - class structure and relationships
  - sequencing of interactions and events

Model is more precise and concise than the informal problem definition

UML Models

- Use Case Model
  (takes the place of the functional model in OMT)
  - user orientation
  - what does the system do? how are various results computed (without regard to sequence)?
- Class Model
  - static structure
  - what objects are in the system? how are they related?
- Dynamic Model
  - behavioral aspects
  - what events occur, when do they occur and in what order?

OMT Analysis and Design: Steps

- Functional Modeling
- Class Modeling
- Dynamic Modeling
- Refine the Class Model (add Operations)
- Iterate and refine all models
  - After the first iteration, steps may occur in parallel or out of order
  - All models must be kept in synch as changes are made

Class Modeling

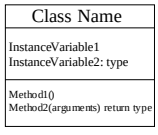
- Identify objects and classes
- Prepare a data dictionary
- Identify associations between objects
- Identify class attributes and initial set of operations
- Organize object classes using inheritance

Classes, Attributes and Operations

- Attributes define the properties of the objects
  - every instance of the class has the same attributes
  - an attribute has a data type
  - the values of the attributes may differ among instances
- Operations define the behavior of the objects
  - action perfomed on or by an object
  - available for all instances of the class
  - need not be unique among classes

Class	Attributes	Operations
ball	radius, weight	catch, throw
football	air pressure	pass, kick, hand-off
baseball	liveness	hit, pitch, tag

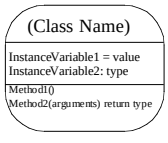
Object Model Notation, review



Classes are represented as rectangles;

The class name is at the top, followed by attributes (instance variables) and methods (operations)

Depending on context some information can be hidden such as types or method arguments

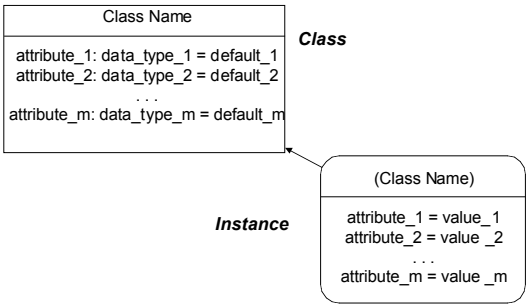


Objects are represented as rounded rectangles;

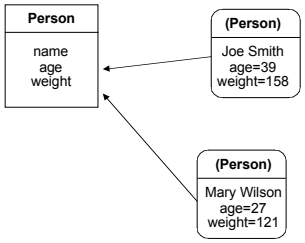
The object's name is its classname surrounded by parentheses

Instance variables can display the values that they have been assigned; pointer types will often point (not shown) to the object being referenced

Object Instantiation Notation



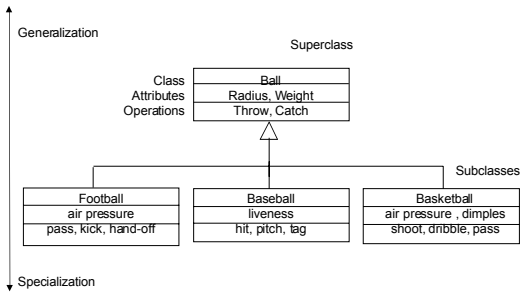
Instantiation: example



Inheritance

- Classes with a set of similar attributes and operations may be organized into a hierarchical relationship
- Common attributes and operations are factored out and assigned to a broad superclass (generalization)
  - generalization is the "is-a" relationship
  - superclasses are ancestors, subclasses are descendants
- A class can be iteratively refined into subclasses that inherit the attributes and operations of the superclass (specialization)

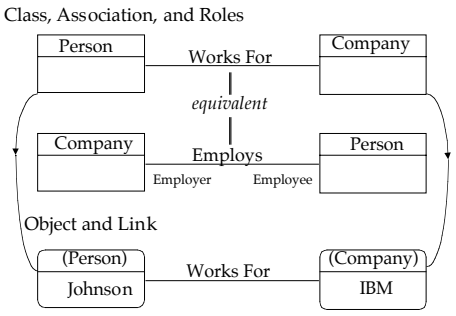
Object Inheritance Notation



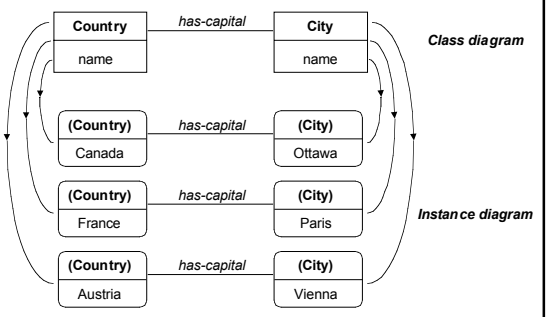
Association and Links

- An association is a relation among two or more classes describing a group of links, with common structure and semantics
- A link is a relationship or connection between objects and is an instance of an association
- A link or association is inherently bi-directional
  - the name may imply a direction, but it can usually be inverted
  - the diagram is usually drawn to read the link or association from left to right or top to bottom
- A role is one end of an association
  - roles may have names

Object Association Notation



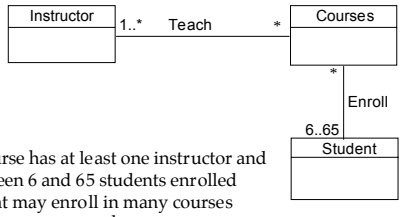
Association and Links



Multiplicity of Associations

- Multiplicity is the number of instances of one class that may relate to a single instance of an associated class
  - 1-to-1 — 1
  - 1-to-many (0 or more) — *
  - 1-to-(zero-or-one) 'optional' — 0..1
  - 1-to-(one-or-more) 'required' — 1..*
  - 1-to-n — n

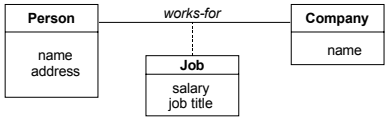
OMT Multiplicity Notation



Each course has at least one instructor and between 6 and 65 students enrolled
A student may enroll in many courses
An instructor may teach many courses

Association Class

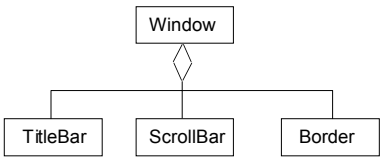
- Enables you to add attributes, operations and other features to associations



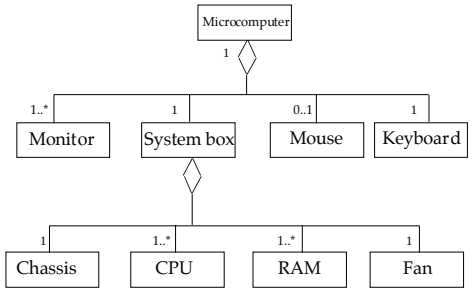
Aggregation

- Aggregation is a special form of association that indicates a "part-of" relationship between a whole and its parts
- Useful when the parts do not have independent existence
  - A part is subordinate to the whole
- In an aggregation, properties and operations may be propagated from the whole to its parts

Object Aggregation Notation



Multilevel aggregation



An Example

FastData Inc. wants a subsystem to process office supply orders via the Web. The user will supply via a form their name, password, account number, and a list of supplies along with an indication of the quantities desired. The subsystem will validate the input, enter the order into a database, and generate a receipt with the order number, expected ship date, and the total cost of the order. If the validation step fails, the subsystem will generate an error message describing the cause of the failure.

Purpose of Example

- We will demonstrate the UML using this example
  - Class modeling will be done first
  - Dynamic and Functional modeling will occur next lecture
  - Detailed design will also occur next lecture
- Things to remember
  - This example does not demonstrate how the technique is applied to ALL problems. Be sure to distinguish between the details of the example and the details of the technique!
  - Ask questions if you do not understand!
  - You will have to apply this technique in Homework 5!

Concise Problem Definition

- Define the problem concisely
  - Use only a single sentence
- "FastData, Inc. employees may order office supplies via the Web and receive a receipt confirming the order"
- This is the first step towards identifying the classes of the subsystem

Informal Strategy

- Identify the constraints governing the system
  - Use only a single paragraph

    "FastData, Inc. employees may order office supplies via the Internal Web and receive a receipt confirming the order. The order must include the user name, user password, account number, and the list of supplies. A receipt must be generated containing an order number, ship date, and total cost. If the order is valid, it must be entered into an order database. If the order is invalid, an error message must be generated."
- We now have more information to be used in identifying classes for the subsystem

Formalize the Strategy

- Identify the nouns of the description, which serve as the basis for identifying the subsystem's classes.
  - Look for out-of-domain nouns (and throw them out!)
  - Look for abstract nouns (use these for attributes)
  - The remaining nouns are good candidates!

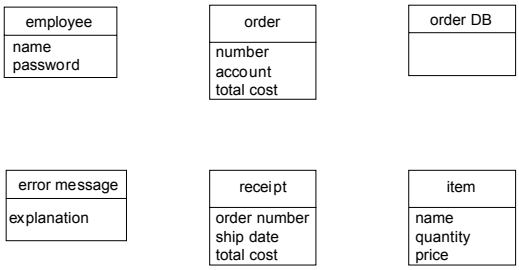
    "FastData, Inc. employees may order office supplies via the Internal Web and receive a receipt confirming the order. The order must include the user name, user password, account number, and the list of supplies. A receipt must be generated containing an order number, ship date, and total cost. If the order is valid, it must be entered into an order database. If the order is invalid, an error message must be generated."

Nouns

- Out-of-Domain
  - Internal Web
- Abstract
  - user name
  - user password
  - account number
  - order number
  - ship date
  - total cost
  - list of supplies
  - office supplies -> item
- Good Candidates
  - employee
  - item (was office supplies)
  - receipt
  - order
  - order database
  - error message
- Notes
  - We have decided not to worry about the Web in this design. Instead we focus on the inputs and outputs and defer the Web details until later.

Topic 10 OOAD 31

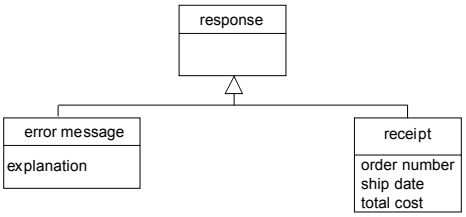
Class Model



Topic 10 OOAD 32

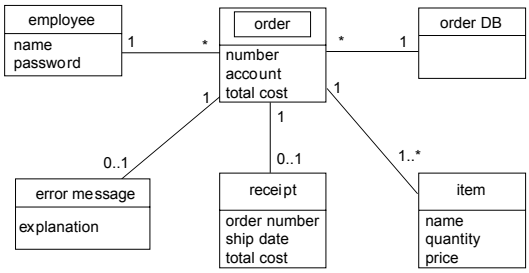
Class Model, *continued*

Since both receipts and error messages will be generated as output it might make sense to have them as subclasses of a more general class. We do not know enough yet to assign it attributes however.



Topic 10 OOAD 33

Class Model, relationships



Topic 10 OOAD 34