Systems Analysis & Design
CS183 Spring Semester 2008

Dr. Jonathan Y. Clark
Email: j.y.clark@surrey.ac.uk

Course Website: www.computing.surrey.ac.uk/personal/st/J.Y.Clark/teaching/sad/cs183.html



Course Textbook:
Systems Analysis and Design With UML 2.0
An Object-Oriented Approach, Second Edition

**Chapter 7:**
**Structural Modelling**



Adapted from slides © 2005 John Wiley & Sons, Inc.



## Key Ideas

- A structural or conceptual model describes the structure of the data that supports the business processes in an organization..
- The structure of data used in the system is represented through *CRC cards*, *class diagrams*, and *object diagrams*.



## STRUCTURAL MODELS



## Purpose of Structural Models

- Reduce the “semantic gap” between the real world and the world of software
- Create a vocabulary for analysts and users
- Represent things, ideas, and concepts of importance in the application domain

## Classes and Objects

- **Class** – **Template** to define specific instances or objects
- **Object** – **Instantiation** of a class
- **Attributes** – **Describes** the object
- **Behaviours** – specify what an object **can do**



## Basic Characteristics of Object Oriented Systems

- Classes and Objects
- Methods and Messages
- Encapsulation and Information Hiding
- Inheritance
- Polymorphism



## Helpful Hint….'Compile'

- C Classes
- O Objects
- M Methods and Messages
- P Polymorphism
- I Inheritance
- (Last, but not least)
- E Encapsulation



## Classes and Objects

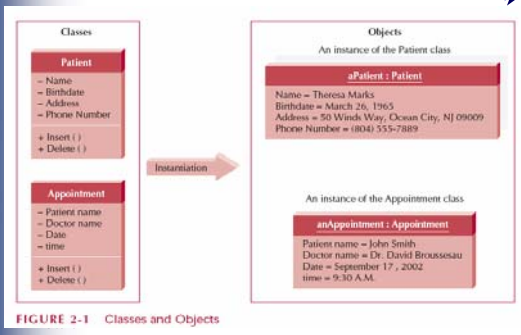



FIGURE 2-1 Classes and Objects

## Encapsulation and Information Hiding

- Encapsulation
  - combination of data and process into an entity
- Information Hiding
  - Only the information required to use a software module is published to the user
- Reusability is the Key Point
  - an object is used by calling methods



## Inheritance

- Superclasses or general classes are at the top of a hierarchy of classes
- Subclasses or specific classes are at the bottom
- Subclasses inherit attributes and methods from classes higher in the hierarchy

## Class Hierarchy

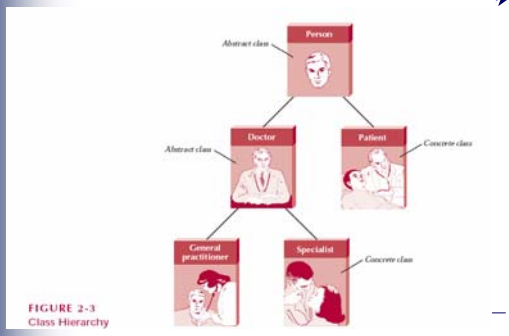



FIGURE 2-3
Class Hierarchy

## Inheritance

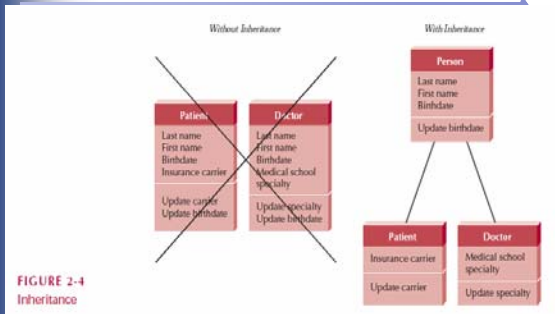



FIGURE 2-4
Inheritance





## Classes

- **Templates** for creating instances or objects
  - Concrete (can have real instances)
  - Abstract (only exists to hold subclasses)
- Typical examples:
  - Application domain, user interface, data structure, file structure, operating environment, document, and multimedia classes



## Attributes

- Units of information relevant to the description of the class
- Only attributes important to the task should be included





## Operations (Methods)

- Action that instances/objects can take
- Focus on relevant problem-specific operations (at this point)



## Relationships

- *Generalization*
  - Enables inheritance of attributes and operations [...is a kind of...]
- *Aggregation*
  - Relates parts to the whole [..is a part of..]
- *Association*
  - Miscellaneous relationships between classes

## CLASS-RESPONSIBILITY-COLLABORATION CARDS



## Responsibilities and Collaborations

- Responsibilities
  - Knowing
  - Doing
- Collaboration
  - Objects working together to service a request



## A CRC Card

**Front:**

| Class Name: Patient | ID: 3 | Type: Concrete, Domain |
|---|---|---|
| Description: An Individual that needs to receive or has received medical attention | | Associated Use Cases: 2 |

| Responsibilities | Collaborators |
|---|---|
| Make appointment | Appointment |
| Calculate last visit | |
| Change status | |
| Provide medical history | Medical history |



## Back of CRC Card

**Attributes:**

Amount (double)

Insurance carrier (text)

**Relationships:**

**Generalization (a-kind-of):** Person

**Aggregation (has-parts):** Medical History

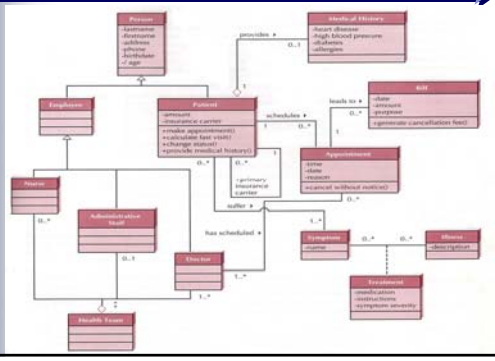
**Other Associations:** Appointment



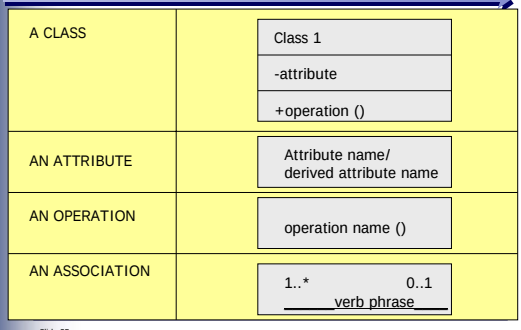
## CLASS DIAGRAMS



## Example Class Diagram



## Class Diagram Syntax






## More on Attributes

- Derived attributes
  - /age, for example can be calculated from birth date and current date
- Visibility
  - + Public (not hidden)
  - # Protected (hidden from all except immediate subclasses)
  - - Private (hidden from all other classes)



## More on Operations

- Constructor
  - Creates object (ie. creates instance)
- Destructor
  - Removes object (ie. removes instance)
- Query
  - Makes information about state available
- Update
  - Changes values of some or all attributes



## Generalization and Aggregation

- *Generalization* shows that a subclass **inherits** from a superclass
  - Doctors, nurses, admin personnel are **kinds of** employees
- *Aggregation* classes **comprise** other classes
  - Health team class **comprised of** doctor, nurses, admin personnel classes



## More on Relationships

- Class can be related to itself
- Multiplicity
  - Exactly one (1), zero or more (0..*), one or more (1..*), zero or one (0..1), specified range (eg. 2..4), multiple disjoint ranges (eg. 1..3,5)
- Association class (class describing a relationship)



## Simplifying Class Diagrams

- A view mechanism can show a subset of information
  - Eg. A use-case view that shows only that part of the diagram referring to a particular use case.
  - Eg. A view showing only aggregations
  - Eg. A view showing only the class name and attributes

## Object Diagrams

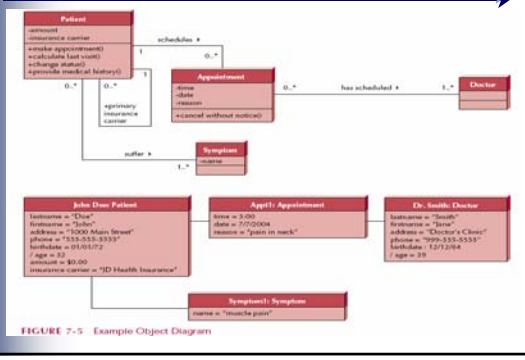



FIGURE 7-5 Example Object Diagram



## CREATING CRC CARDS AND CLASS DIAGRAMS





## Object Identification

- Textual analysis of use-case information
  - Nouns suggest classes
  - Verbs suggest operations
- Creates a rough first cut
- Common object list
- Incidents
- Roles





## Patterns

- Useful groupings of classes that recur in various situations
- Contain groups of classes that collaborate or work together
- Enable reusability





## Steps for Object Identification and Structural Modelling

1. Create CRC cards by performing textual analysis on the use-cases.
2. Brainstorm additional candidate classes, attributes, operations, and relationships by using the common object list approach.
3. Role-play each use-case using the CRC cards.
4. Create the class diagram based on the CRC cards.
5. Review the structural model for missing and/or unnecessary classes, attributes, operations, and relationships.
6. Incorporate useful patterns.
7. Review the structural model.





## Summary

- *CRC cards* capture the essential elements of a class.
- *Class and object diagrams* show the underlying structure of an object-oriented system.
- Constructing the structural model is an iterative process involving: *textual analysis, brainstorming objects, role playing, creating the diagrams*, and *incorporating useful patterns.*