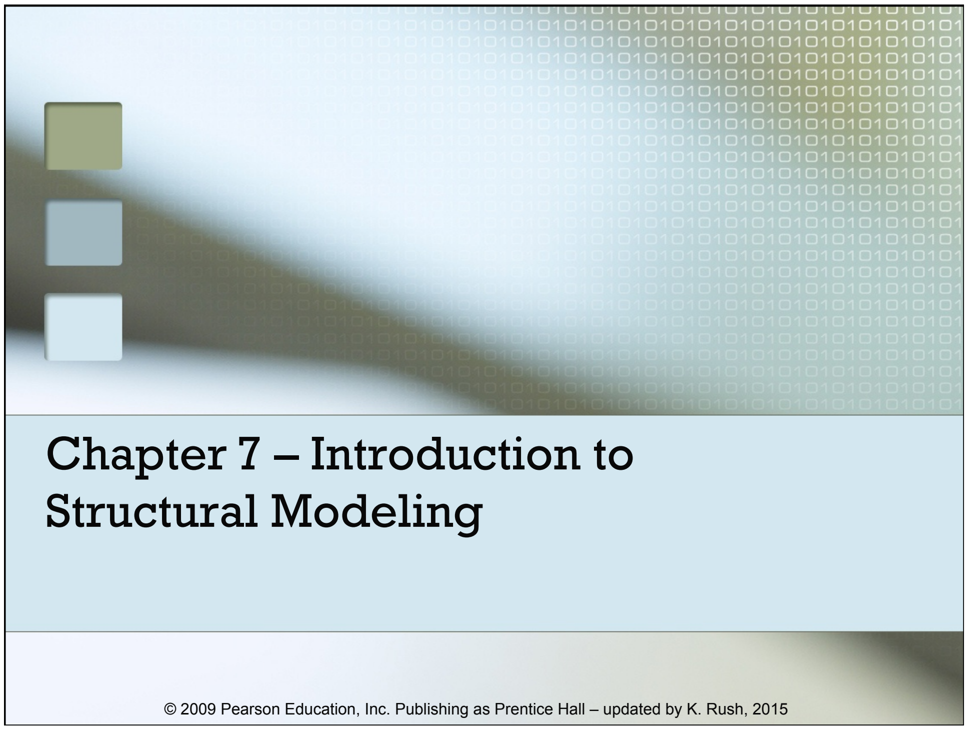
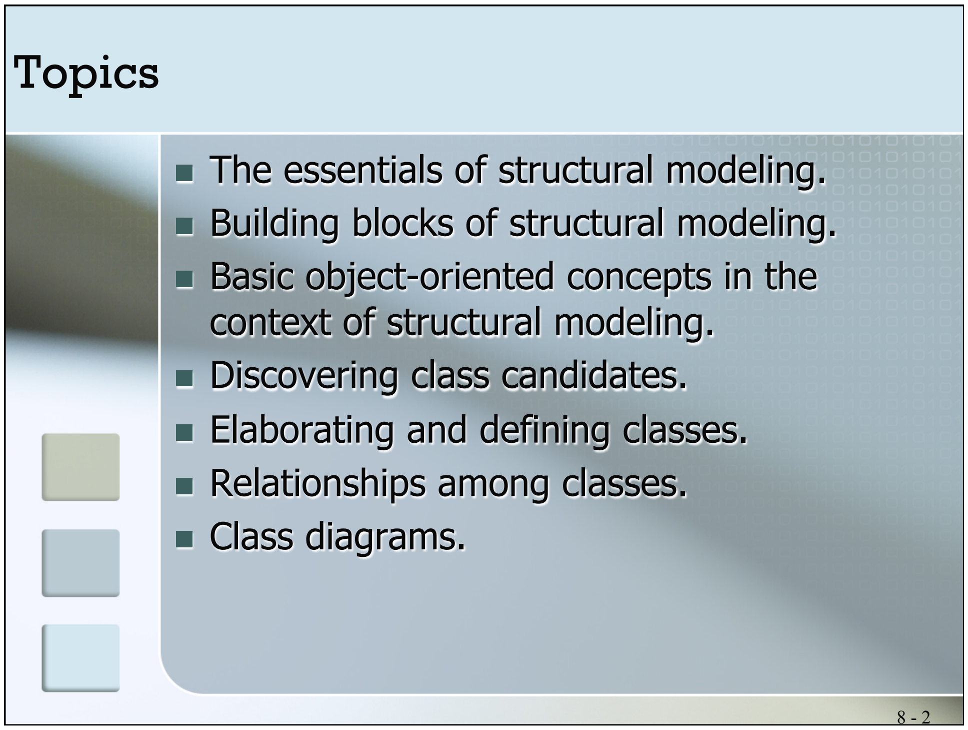


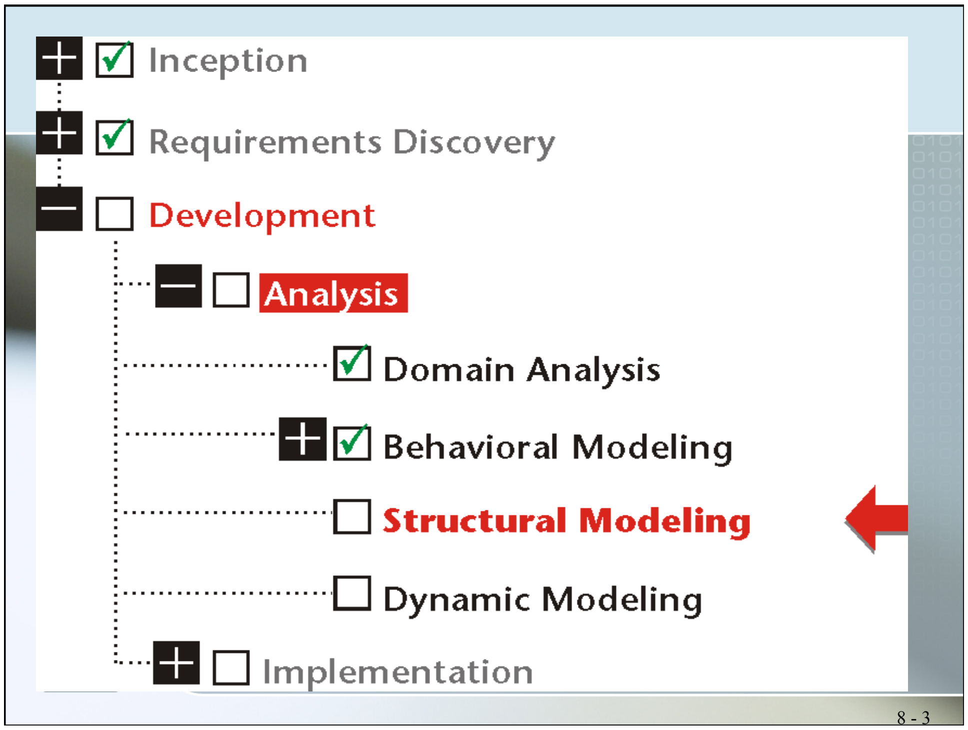
# Chapter 7 – Introduction to Structural Modeling

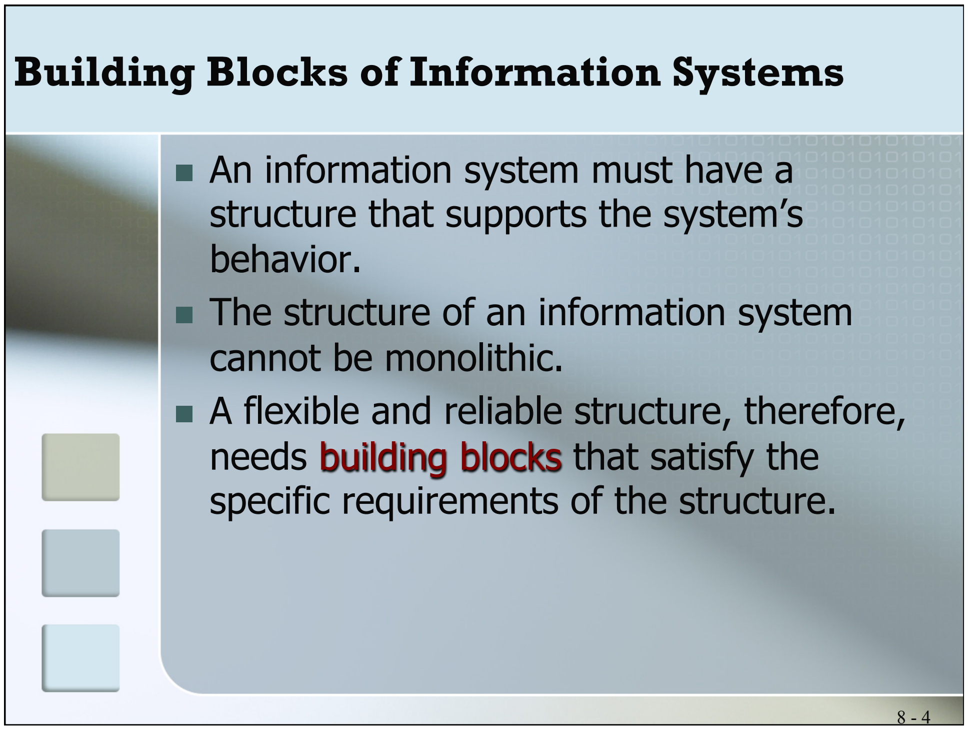


## Topics

- The essentials of structural modeling.
- Building blocks of structural modeling.
- Basic object-oriented concepts in the context of structural modeling.
- Discovering class candidates.
- Elaborating and defining classes.
- Relationships among classes.
- Class diagrams.

8 - 2





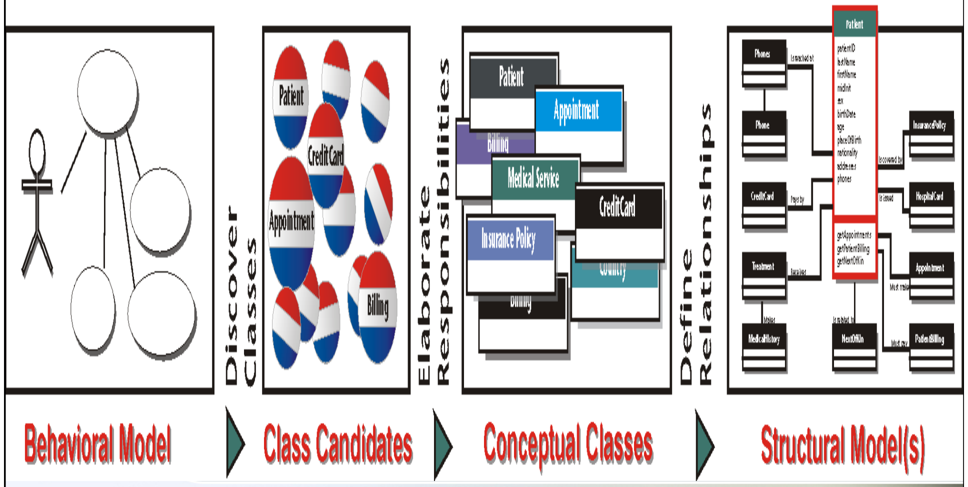
## Building Blocks of Information Systems

- An information system must have a structure that supports the system's behavior.
- The structure of an information system cannot be monolithic.
- A flexible and reliable structure, therefore, needs building blocks that satisfy the specific requirements of the structure.

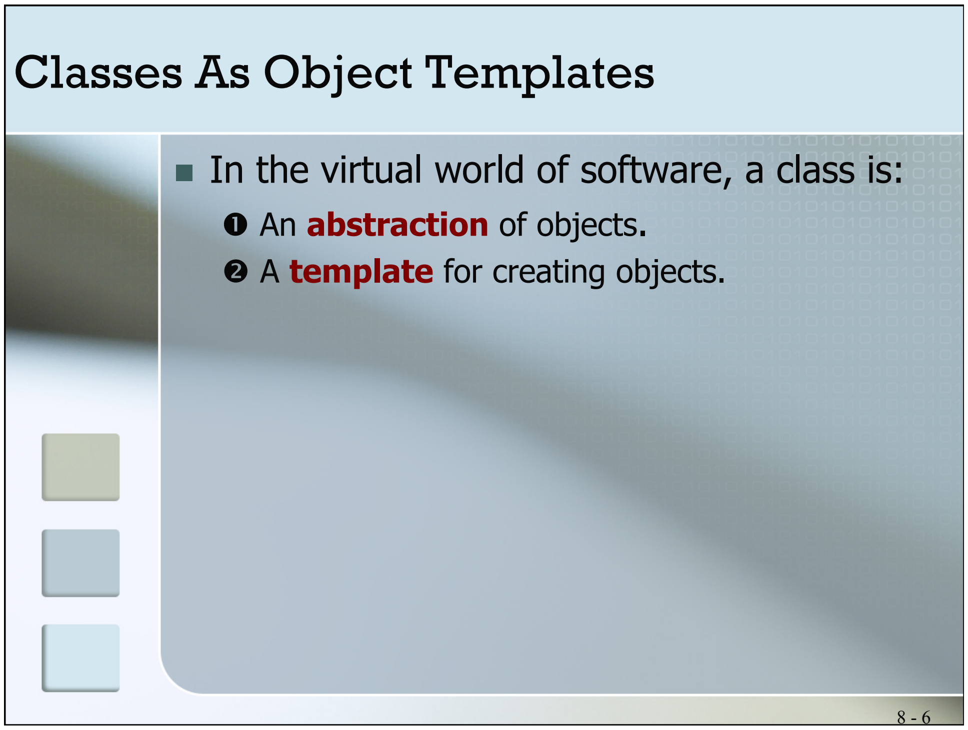
8 - 4

## Building the Conceptual Structure

### From Use Cases to Structural Modeling



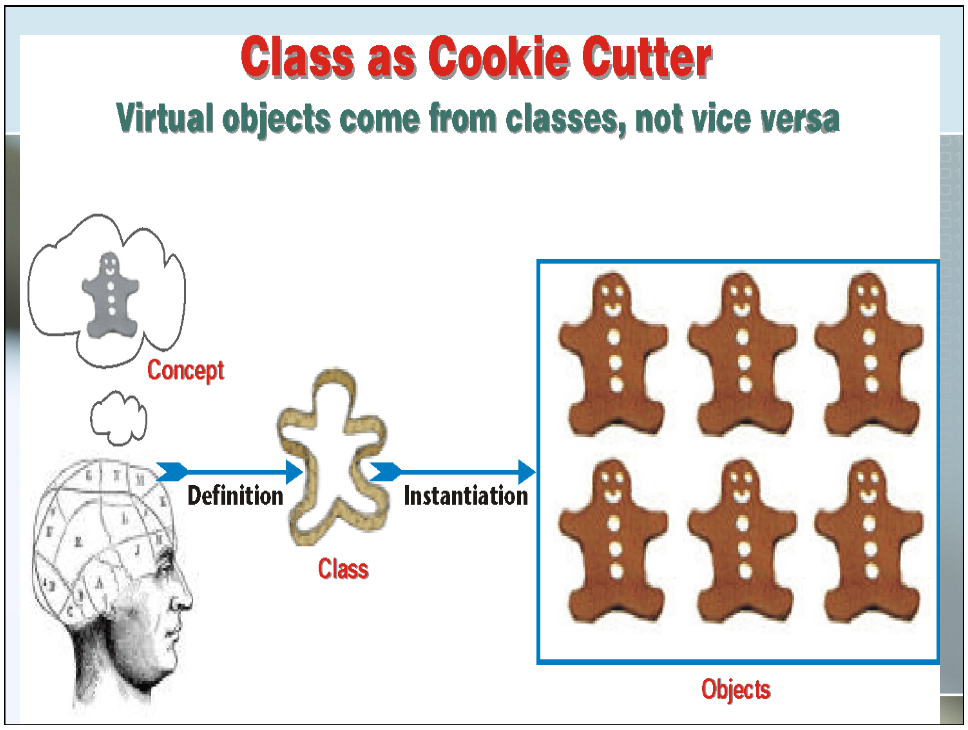
8 - 5

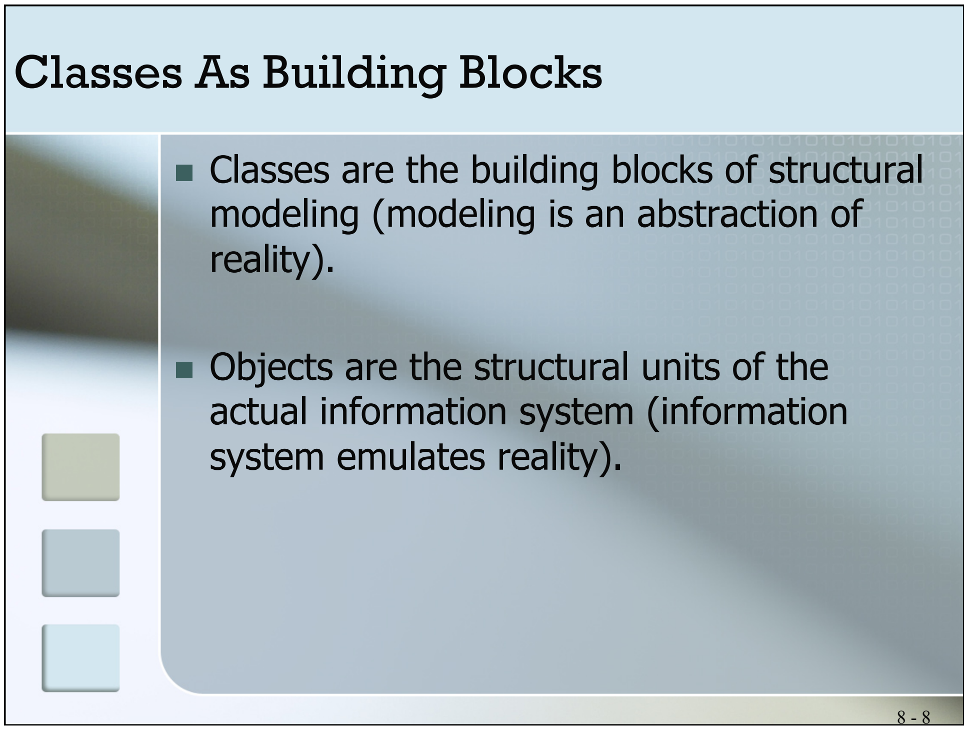


## Classes As Object Templates

- In the virtual world of software, a class is:
  - ❶ An **abstraction** of objects.
  - ❷ A **template** for creating objects.

8 - 6

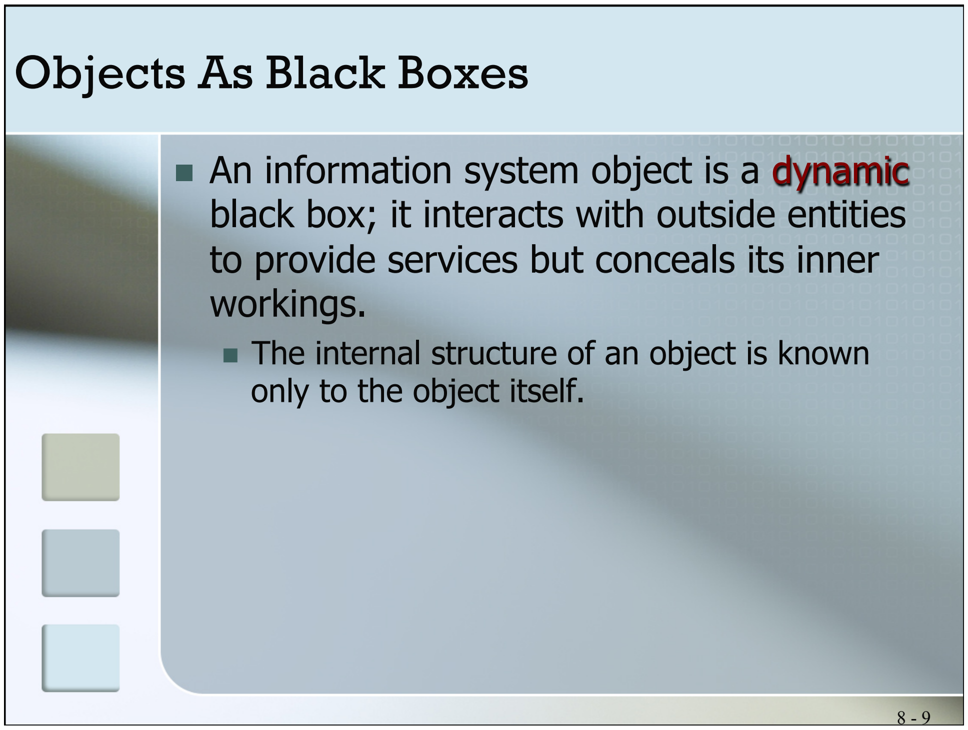




## Classes As Building Blocks

- Classes are the building blocks of structural modeling (modeling is an abstraction of reality).
- Objects are the structural units of the actual information system (information system emulates reality).

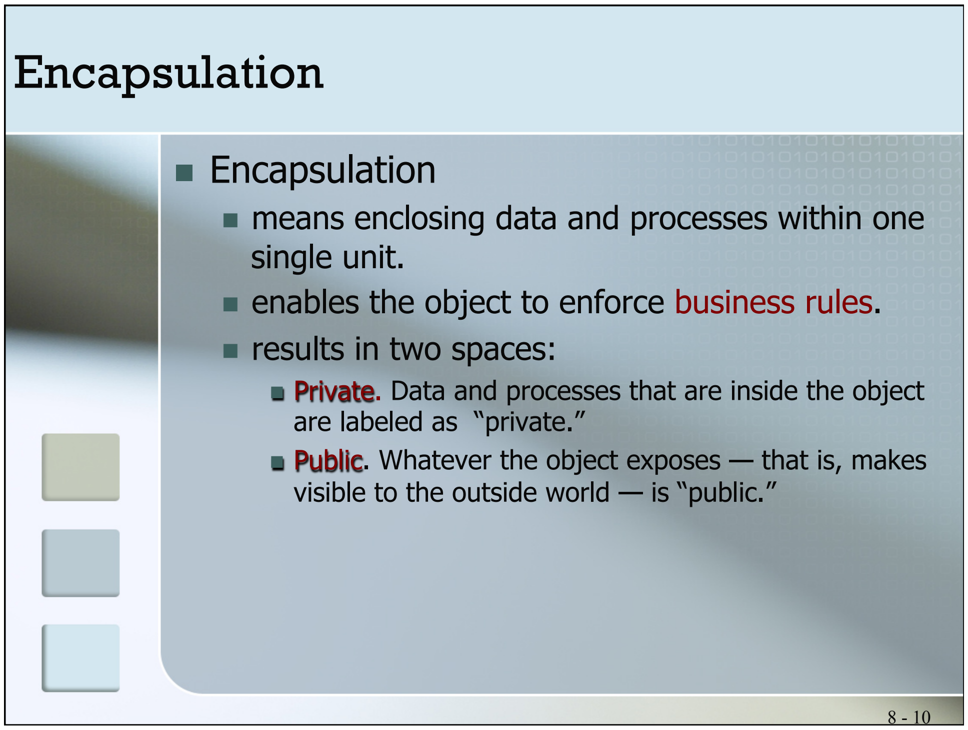
8 - 8



## Objects As Black Boxes

- An information system object is a dynamic black box; it interacts with outside entities to provide services but conceals its inner workings.
  - The internal structure of an object is known only to the object itself.

8 - 9



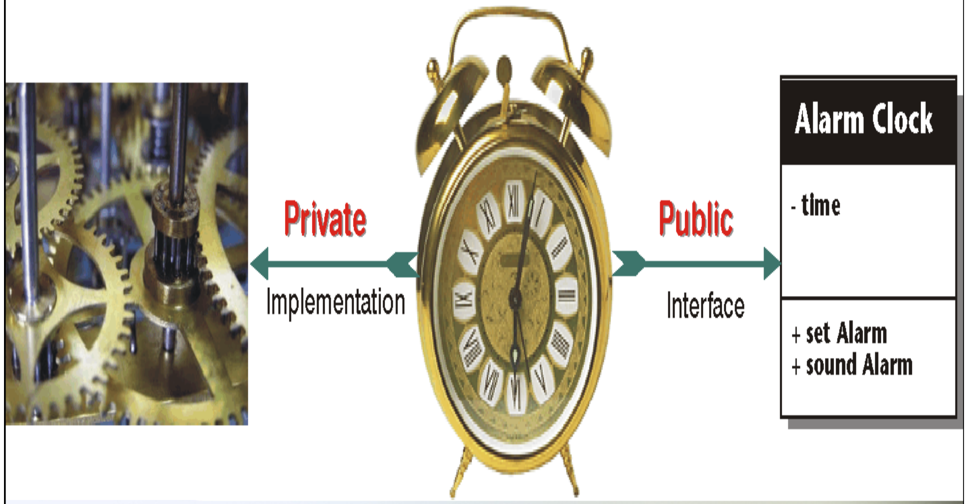
## Encapsulation

- Encapsulation
  - means enclosing data and processes within one single unit.
  - enables the object to enforce business rules.
  - results in two spaces:
    - Private. Data and processes that are inside the object are labeled as "private."
    - Public. Whatever the object exposes — that is, makes visible to the outside world — is "public."

8 - 10

## Encapsulation

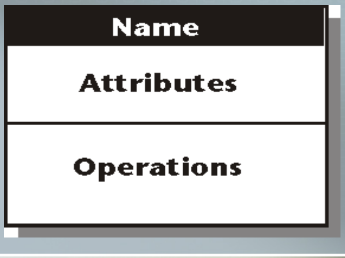
### Public vs. Private



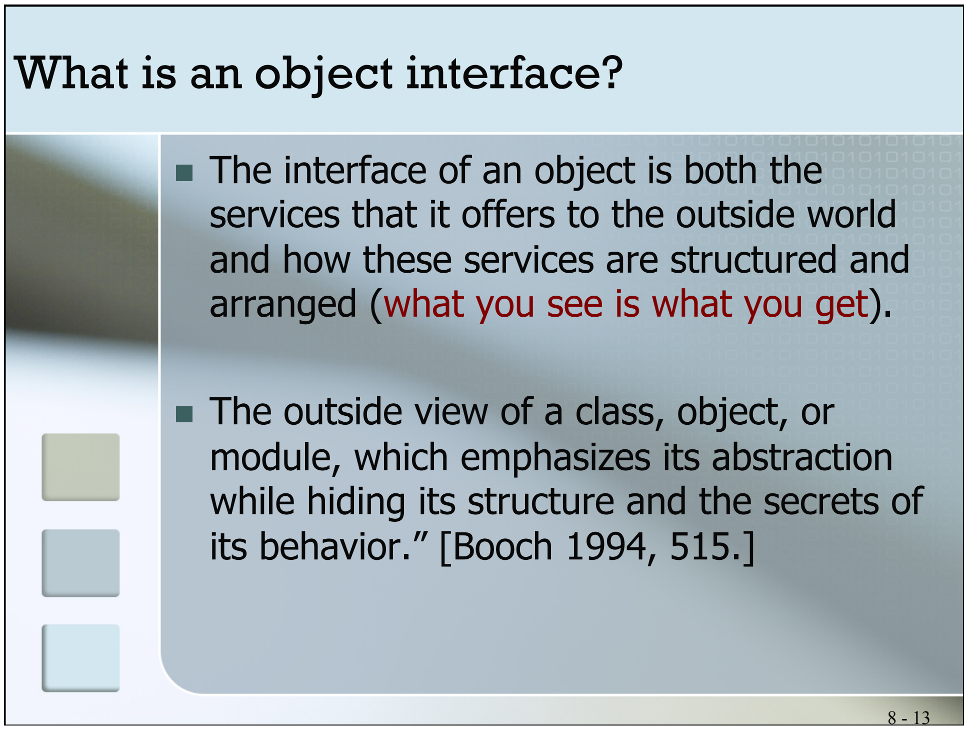
8 - 11

## Information Hiding

- Information hiding conceals the inner entities and the workings of the object from outside entities.
- The box metaphor for classes and objects emphasizes: encapsulation and information hiding.



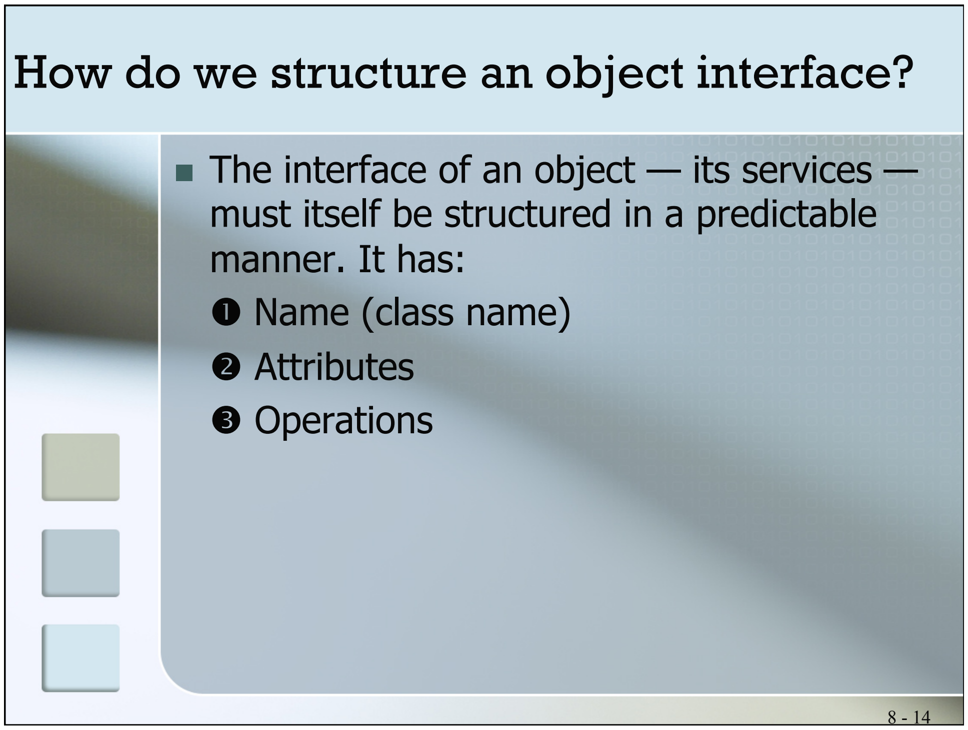
8 - 12



## What is an object interface?

- The interface of an object is both the services that it offers to the outside world and how these services are structured and arranged (what you see is what you get).
- The outside view of a class, object, or module, which emphasizes its abstraction while hiding its structure and the secrets of its behavior.” [Booch 1994, 515.]

8 - 13



## How do we structure an object interface?

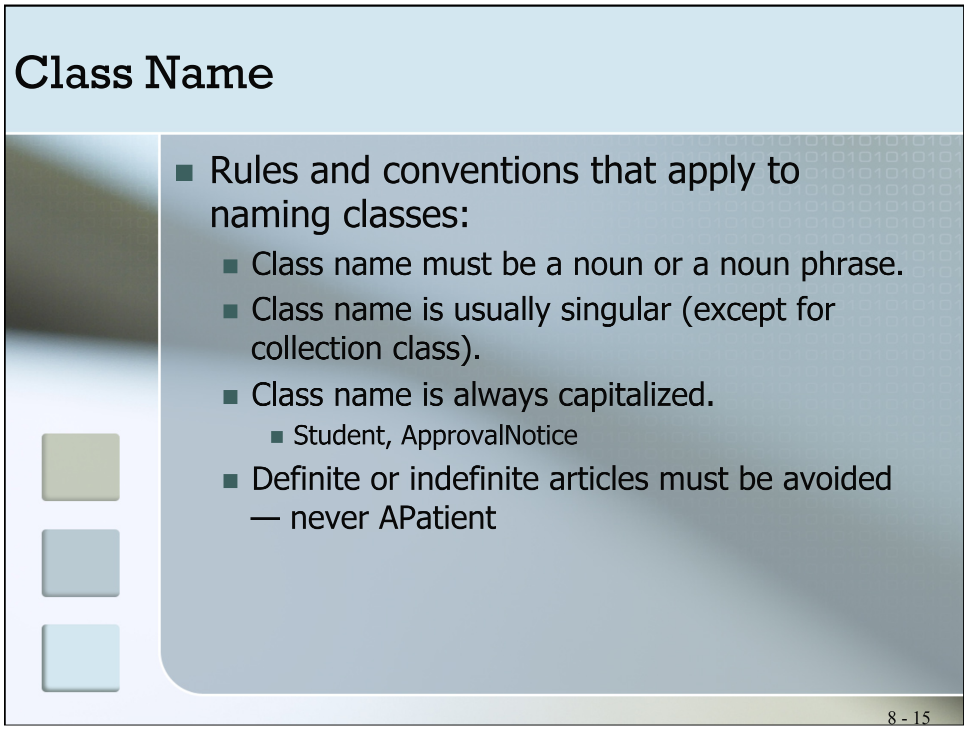
- The interface of an object — its services — must itself be structured in a predictable manner. It has:

  ❶ Name (class name)

  ❷ Attributes

  ❸ Operations

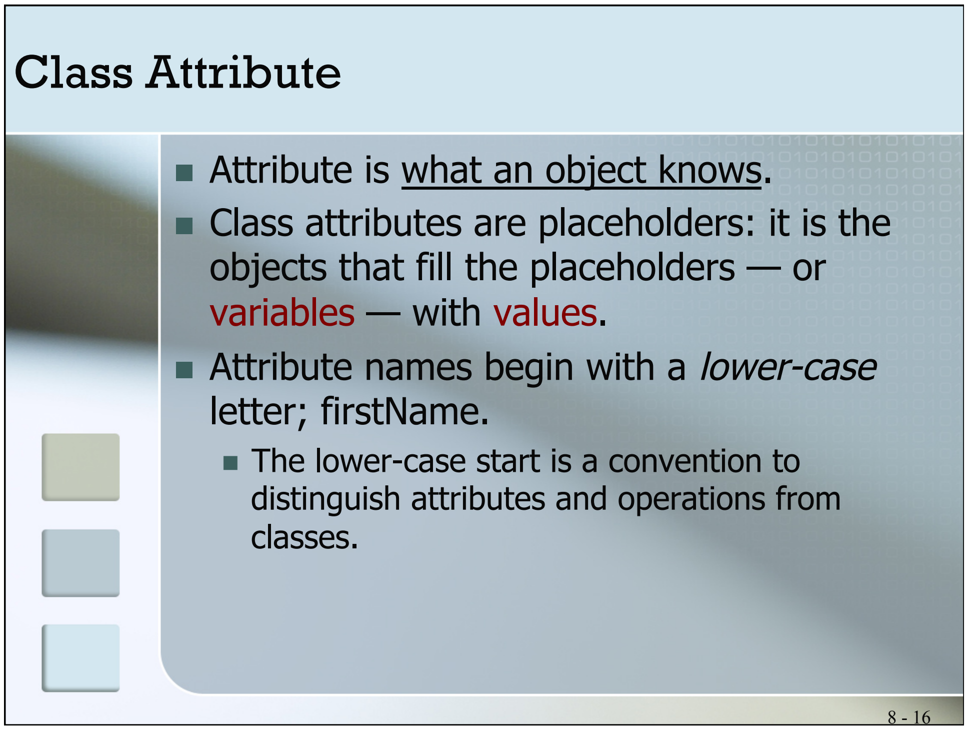
8 - 14



## Class Name

- Rules and conventions that apply to naming classes:
  - Class name must be a noun or a noun phrase.
  - Class name is usually singular (except for collection class).
  - Class name is always capitalized.
    - Student, ApprovalNotice
  - Definite or indefinite articles must be avoided — never APatient

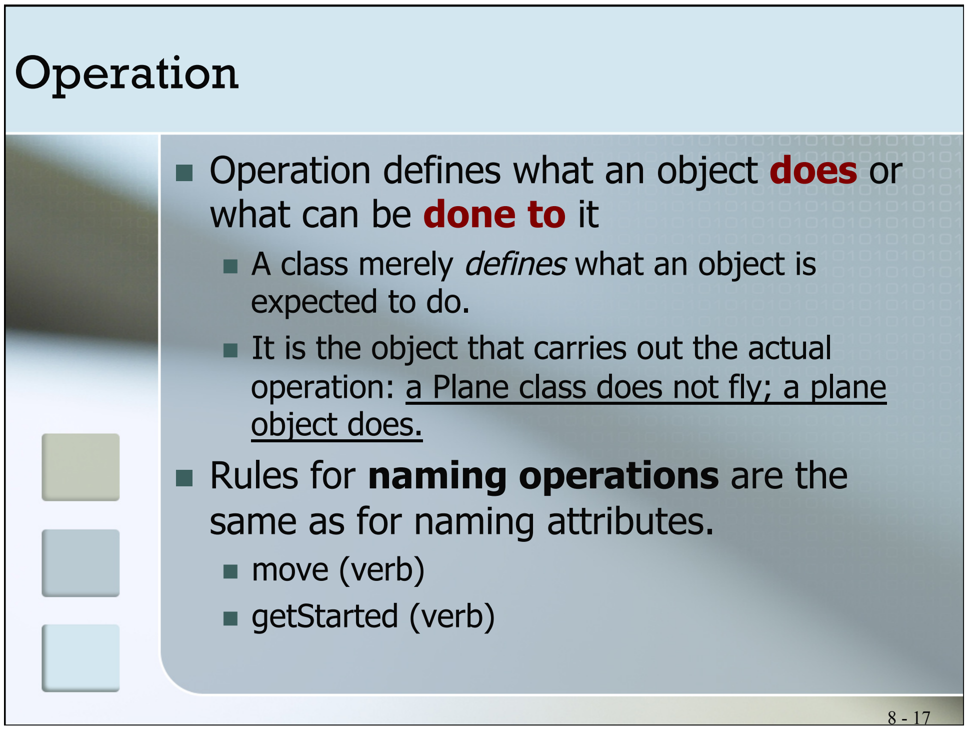
8 - 15



## Class Attribute

- Attribute is what an object knows.
- Class attributes are placeholders: it is the objects that fill the placeholders — or variables — with values.
- Attribute names begin with a *lower-case* letter; firstName.
  - The lower-case start is a convention to distinguish attributes and operations from classes.

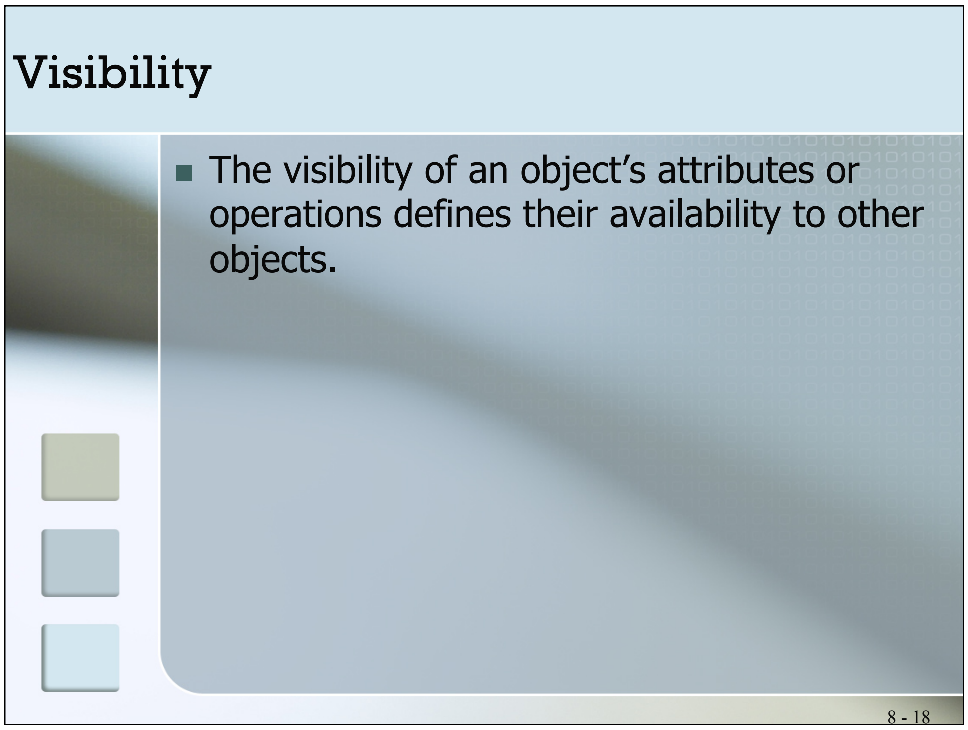
8 - 16



## Operation

- Operation defines what an object **does** or what can be **done to** it
  - A class merely *defines* what an object is expected to do.
  - It is the object that carries out the actual operation: <u>a Plane class does not fly; a plane object does.</u>
- Rules for **naming operations** are the same as for naming attributes.
  - move (verb)
  - getStarted (verb)

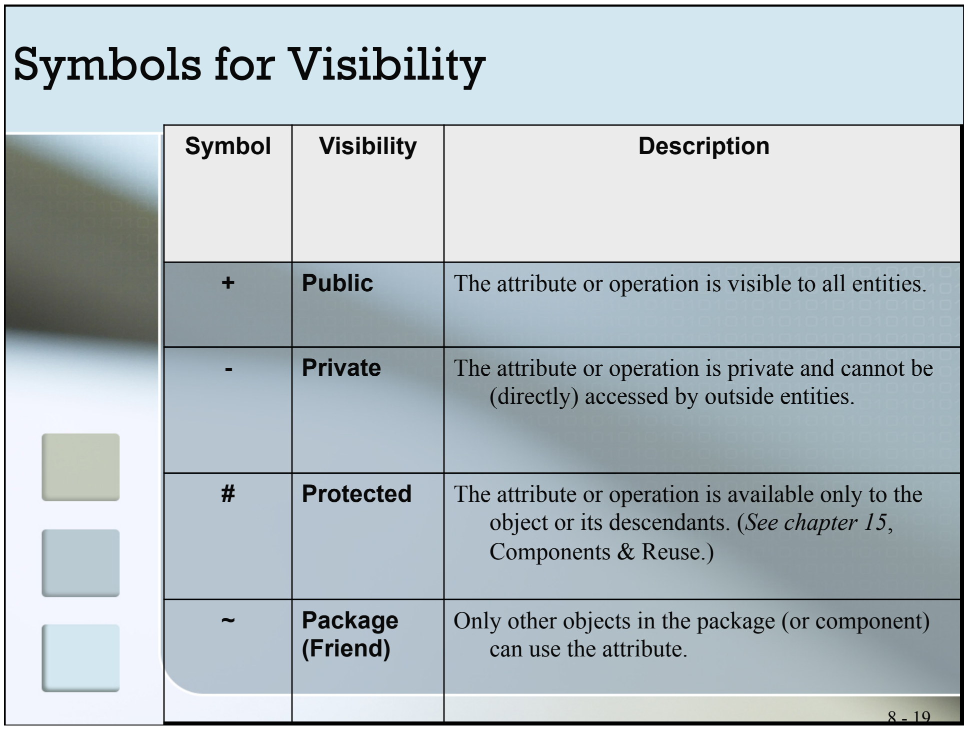
8 - 17



## Visibility

- The visibility of an object's attributes or operations defines their availability to other objects.

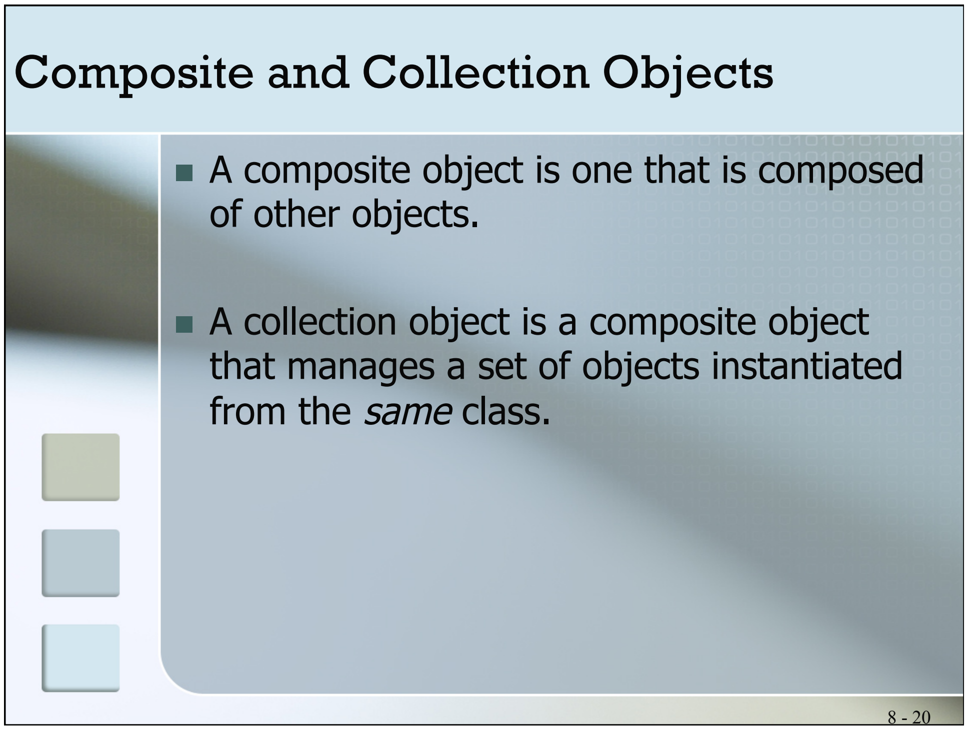
8 - 18



## Symbols for Visibility

| Symbol | Visibility | Description |
|---|---|---|
| + | **Public** | The attribute or operation is visible to all entities. |
| - | **Private** | The attribute or operation is private and cannot be (directly) accessed by outside entities. |
| # | **Protected** | The attribute or operation is available only to the object or its descendants. (*See chapter 15*, Components & Reuse.) |
| ~ | **Package (Friend)** | Only other objects in the package (or component) can use the attribute. |

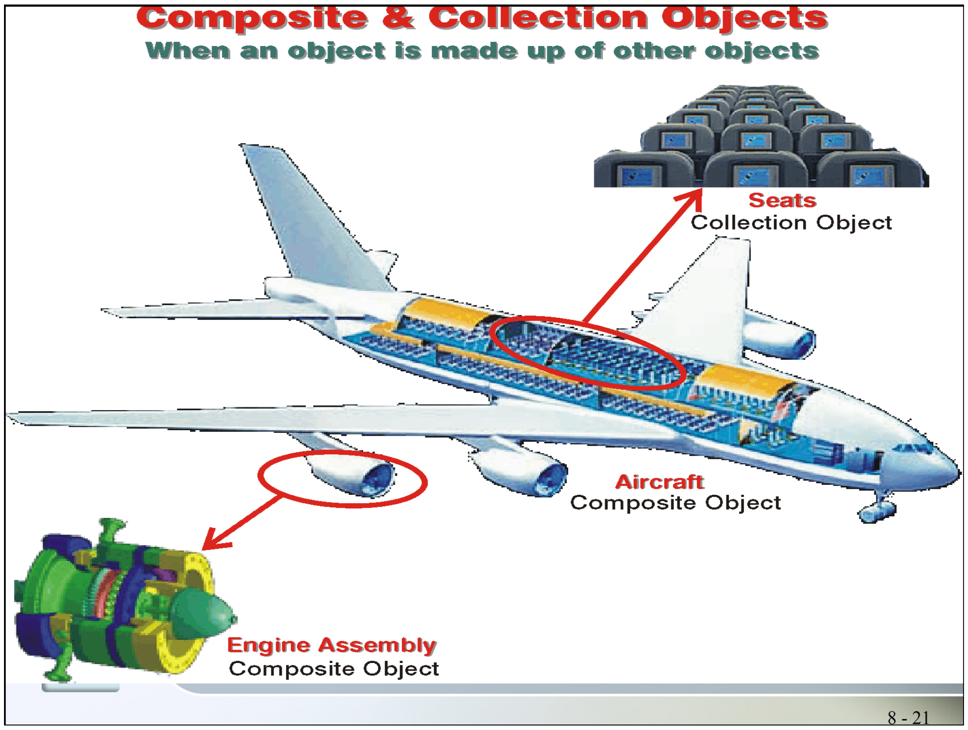
8 - 19



## Composite and Collection Objects

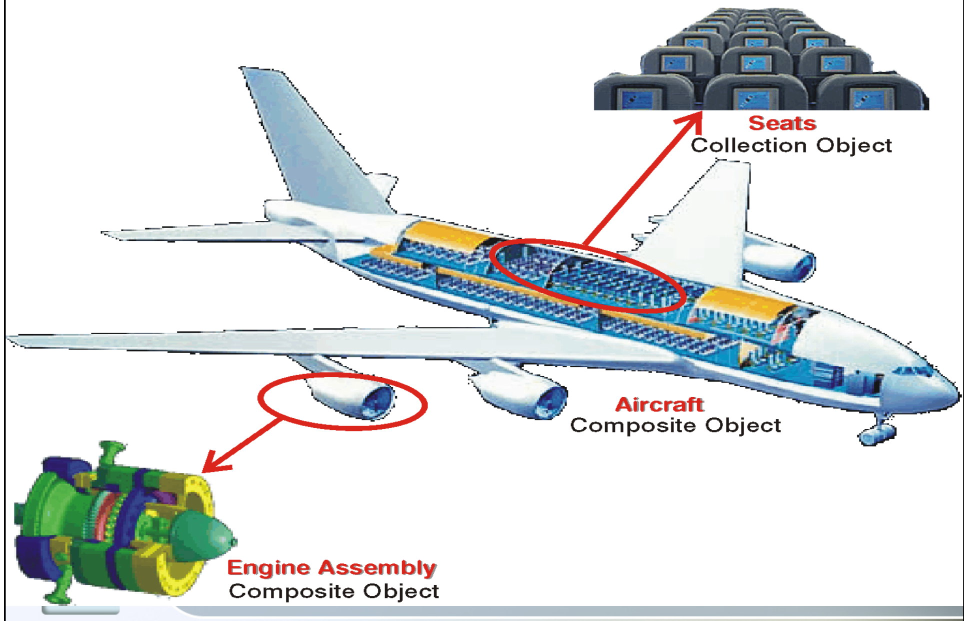
- A composite object is one that is composed of other objects.
- A collection object is a composite object that manages a set of objects instantiated from the *same* class.

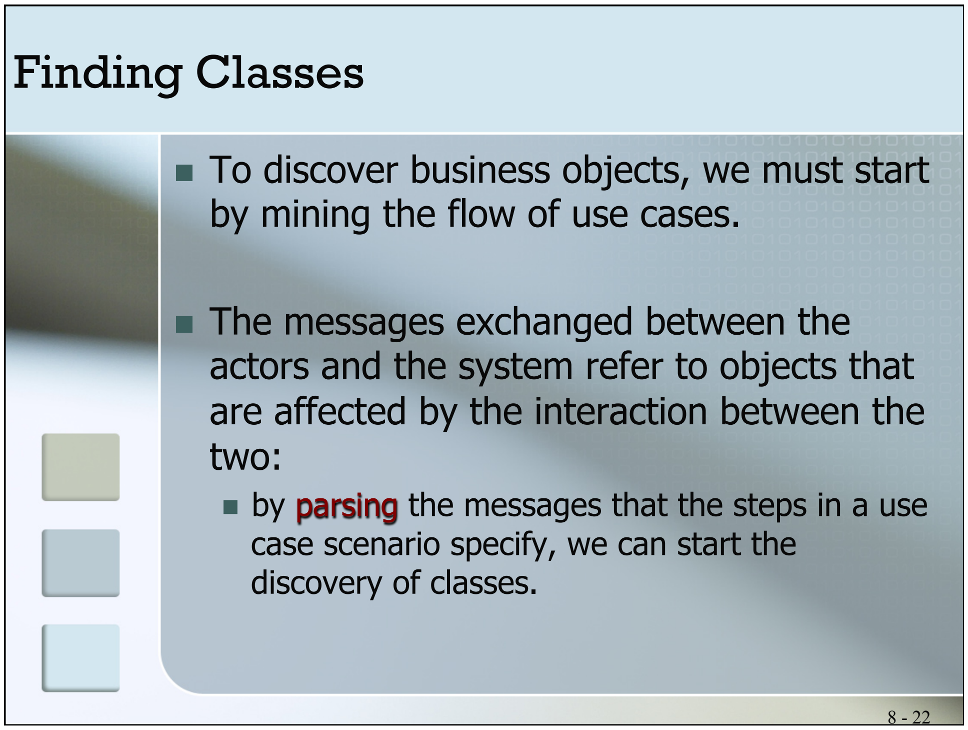
8 - 20



## Composite & Collection Objects

When an object is made up of other objects





## Finding Classes

- To discover business objects, we must start by mining the flow of use cases.
- The messages exchanged between the actors and the system refer to objects that are affected by the interaction between the two:
  - by parsing the messages that the steps in a use case scenario specify, we can start the discovery of classes.

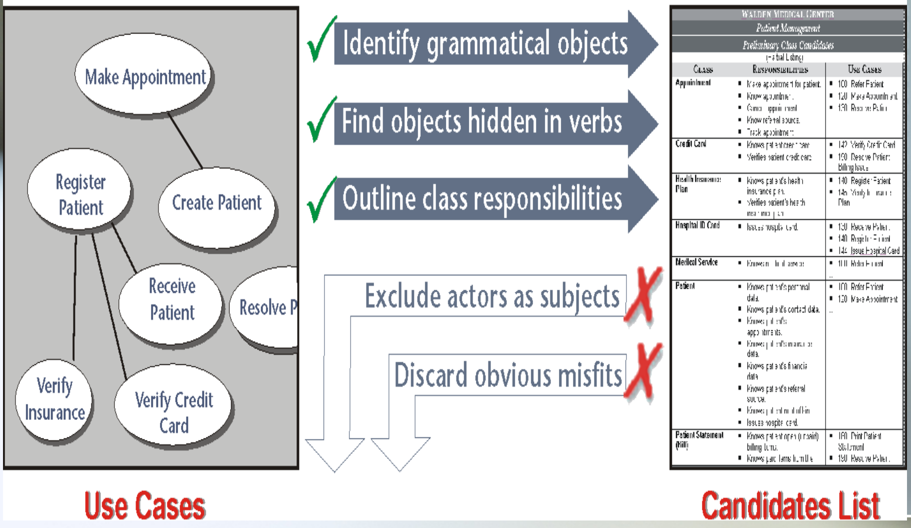
## Responsibilities

- An object's responsibilities consist of what it does and what it knows; in other words, its operations and its attributes.
- Before we can define classes in detail, we must discover class candidates and outline their tentative responsibilities.

8 - 23

## Finding Class Candidates

### Parsing the Behavior for Structure


Make Appointment
Register Patient
Create Patient
Receive Patient
Resolve P
Verify Insurance
Verify Credit Card
✓ Identify grammatical objects
✓ Find objects hidden in verbs
✓ Outline class responsibilities
Exclude actors as subjects ✗
Discard obvious misfits ✗
Use Cases
Candidates List

WALDEN MEDICAL CENTER

*Patient Management*

*Preliminary Class Candidates*

(Partial Listing)

| CLASS | RESPONSIBILITIES | USE CASES |
|---|---|---|
| **Appointment** | ▪ Make appointment for patient.<br>▪ Know appointment.<br>▪ Cancel appointment.<br>▪ Know referral source.<br>▪ Track appointment. | ▪ 100 Refer Patient<br>▪ 120 Make Appointment<br>▪ 130 Receive Patient |
| **Credit Card** | ▪ Knows patient credit card.<br>▪ Verifies patient credit card. | ▪ 142 Verify Credit Card<br>▪ 190 Resolve Patient Billing Issue |
| **Health Insurance Plan** | ▪ Knows patient's health insurance plan.<br>▪ Verifies patient's health insurance plan. | ▪ 140 Register Patient<br>▪ 145 Verify Insurance Plan |
| **Hospital ID Card** | ▪ Issues hospital card. | ▪ 130 Receive Patient<br>▪ 140 Register Patient<br>▪ 144 Issue Hospital Card |
| **Medical Service** | ▪ Knows medical service. | ▪ 100 Refer Patient<br>... |
| **Patient** | ▪ Knows patient's personal data.<br>▪ Knows patient's contact data.<br>▪ Knows patient's appointments.<br>▪ Knows patient's insurance data.<br>▪ Knows patient's financial data.<br>▪ Knows patient's referral source.<br>▪ Knows patient next of kin.<br>▪ Issues hospital card. | ▪ 100 Refer Patient<br>▪ 120 Make Appointment<br>... |
| **Patient Statement (Bill)** | ▪ Knows patient open (unpaid) billing items.<br>▪ Knows paid items from the | ▪ 160 Print Patient Statement<br>▪ 190 Resolve Patient |

8 - 24

## Finding Class Candidates

**Make Appointment**

**Normal Flow:**

1. Appointment clerk verifies that the needed medical service is provided by the hospital.
2. Appointment clerk records patient's personal and contact data.
3. Appointment clerk records information about the referral source.
4. Appointment clerk consults hospital's schedule to find a free slot for the required medical service.
5. Appointment clerk verifies that the patient is available for the appointment.

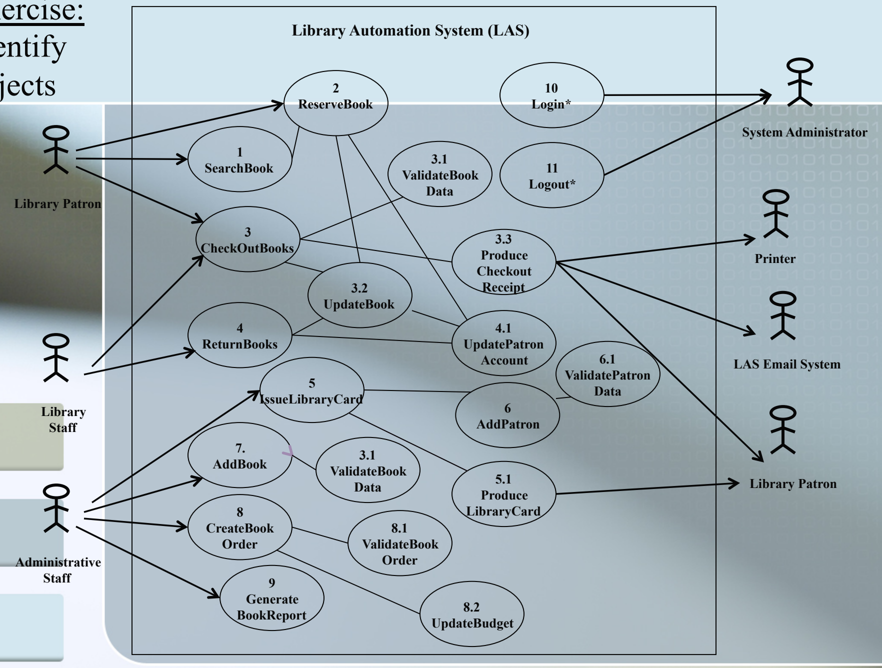
➔ Loop 1: **Repeat** steps 4-5 until hospital's schedule matches patient's availability.

6. Appointment clerk makes the appointment.

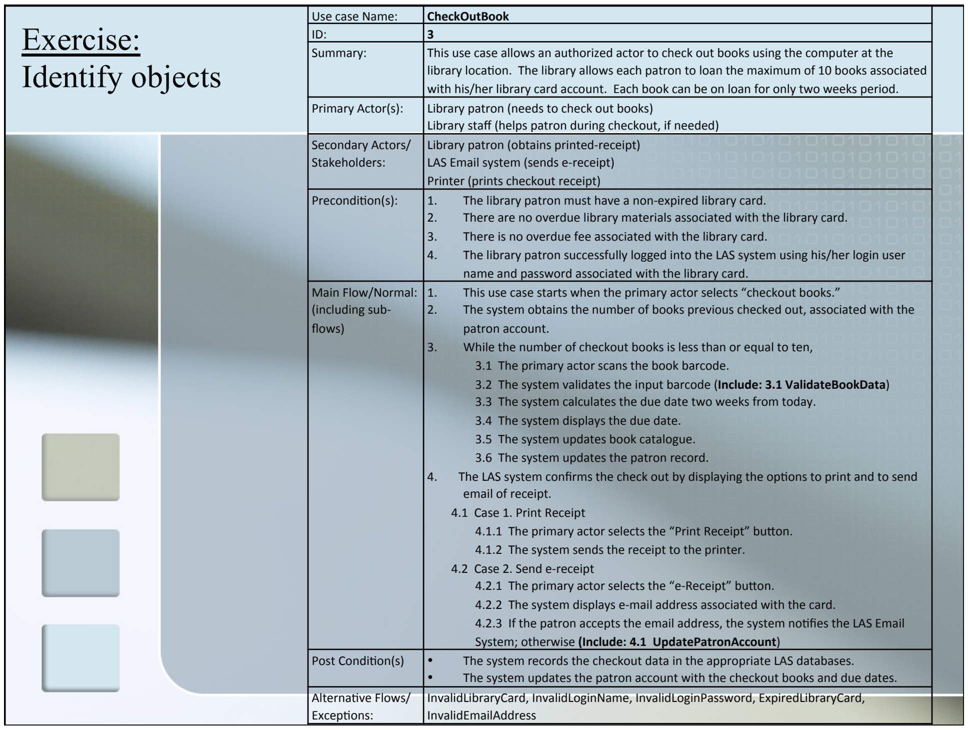
➔ Loop 2: **Repeat** steps 4-6 for each appointment.

8 - 25

## Exercise: Identify objects



* Login and Logout are triggered by these actors: Library Patron, Library Staff and Library Administrative Staff. Updated by K. Rush, SP13

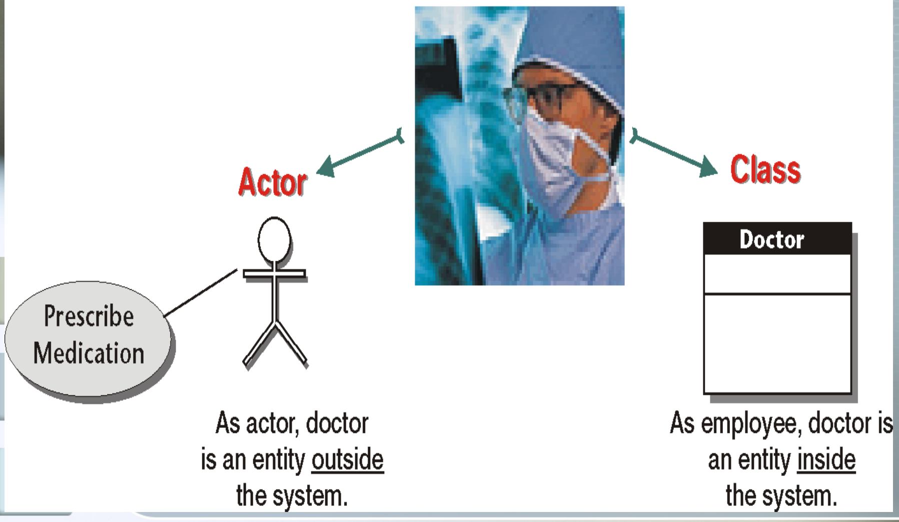


## Exercise: Identify objects

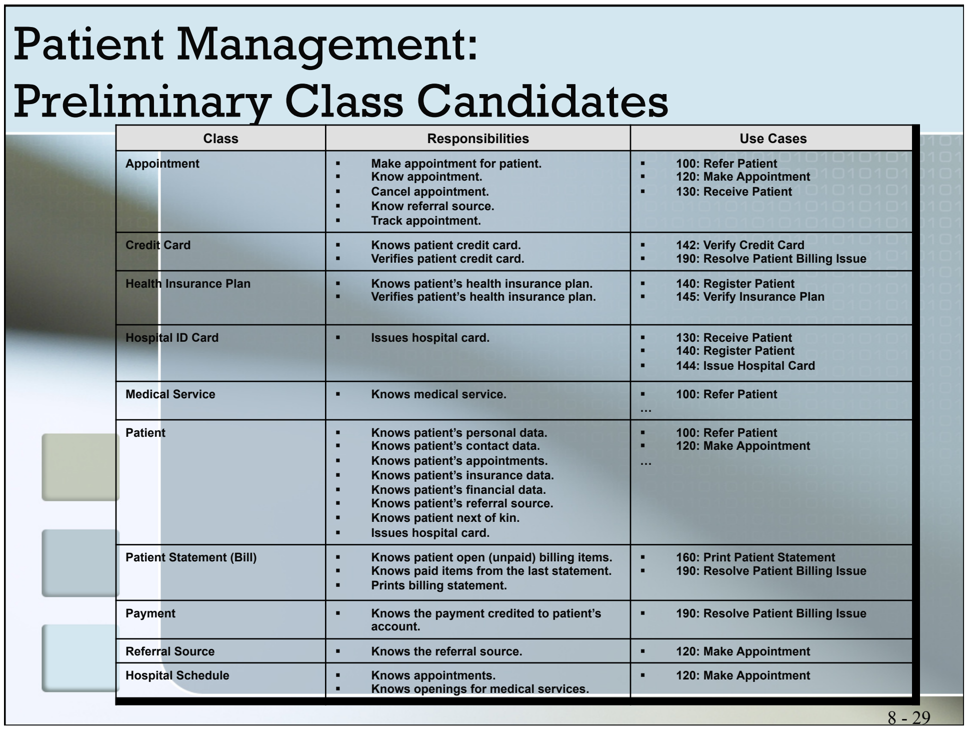
| | |
|---|---|
| Use case Name: | **CheckOutBook** |
| ID: | **3** |
| Summary: | This use case allows an authorized actor to check out books using the computer at the library location. The library allows each patron to loan the maximum of 10 books associated with his/her library card account. Each book can be on loan for only two weeks period. |
| Primary Actor(s): | Library patron (needs to check out books)<br>Library staff (helps patron during checkout, if needed) |
| Secondary Actors/ Stakeholders: | Library patron (obtains printed-receipt)<br>LAS Email system (sends e-receipt)<br>Printer (prints checkout receipt) |
| Precondition(s): | 1. The library patron must have a non-expired library card.<br>2. There are no overdue library materials associated with the library card.<br>3. There is no overdue fee associated with the library card.<br>4. The library patron successfully logged into the LAS system using his/her login user name and password associated with the library card. |
| Main Flow/Normal: (including sub-flows) | 1. This use case starts when the primary actor selects “checkout books.”<br>2. The system obtains the number of books previous checked out, associated with the patron account.<br>3. While the number of checkout books is less than or equal to ten,<br>3.1 The primary actor scans the book barcode.<br>3.2 The system validates the input barcode (**Include: 3.1 ValidateBookData**)<br>3.3 The system calculates the due date two weeks from today.<br>3.4 The system displays the due date.<br>3.5 The system updates book catalogue.<br>3.6 The system updates the patron record.<br>4. The LAS system confirms the check out by displaying the options to print and to send email of receipt.<br>4.1 Case 1. Print Receipt<br>4.1.1 The primary actor selects the “Print Receipt” button.<br>4.1.2 The system sends the receipt to the printer.<br>4.2 Case 2. Send e-receipt<br>4.2.1 The primary actor selects the “e-Receipt” button.<br>4.2.2 The system displays e-mail address associated with the card.<br>4.2.3 If the patron accepts the email address, the system notifies the LAS Email System; otherwise (**Include: 4.1 UpdatePatronAccount**) |
| Post Condition(s) | • The system records the checkout data in the appropriate LAS databases.<br>• The system updates the patron account with the checkout books and due dates. |
| Alternative Flows/ Exceptions: | InvalidLibraryCard, InvalidLoginName, InvalidLoginPassword, ExpiredLibraryCard, InvalidEmailAddress |

## Actor vs. Class

An entity can be both
But the two functions are not the same


Actor
Class
Doctor
Prescribe Medication
As actor, doctor is an entity outside the system.
As employee, doctor is an entity inside the system.

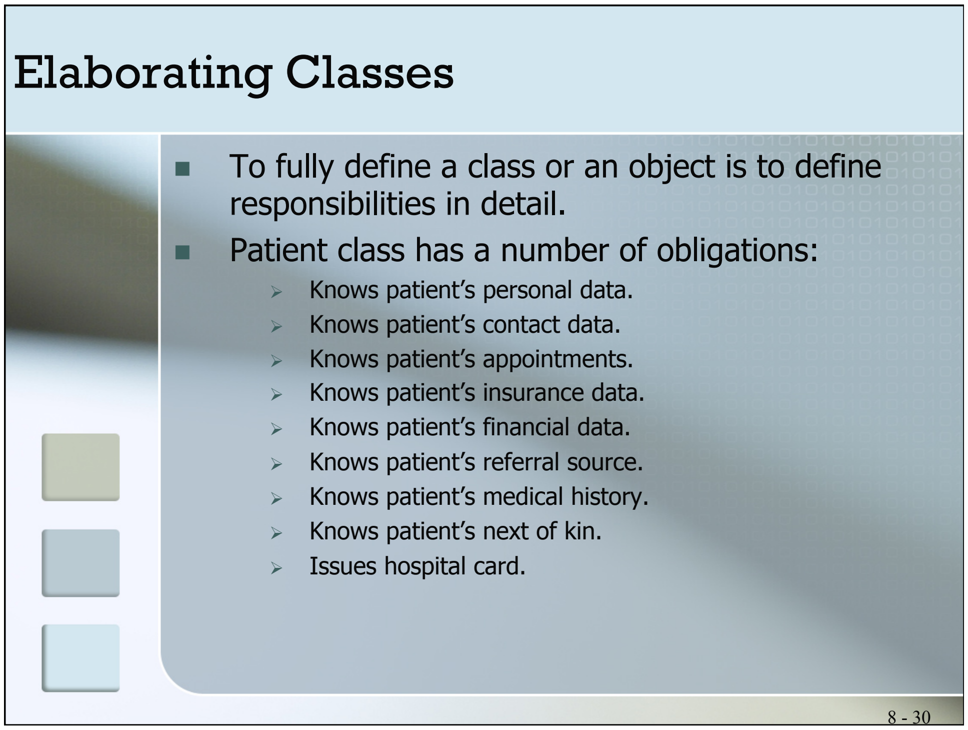
8 - 28



## Patient Management: Preliminary Class Candidates

| Class | Responsibilities | Use Cases |
|---|---|---|
| Appointment | ▪ Make appointment for patient.<br>▪ Know appointment.<br>▪ Cancel appointment.<br>▪ Know referral source.<br>▪ Track appointment. | ▪ 100: Refer Patient<br>▪ 120: Make Appointment<br>▪ 130: Receive Patient |
| Credit Card | ▪ Knows patient credit card.<br>▪ Verifies patient credit card. | ▪ 142: Verify Credit Card<br>▪ 190: Resolve Patient Billing Issue |
| Health Insurance Plan | ▪ Knows patient's health insurance plan.<br>▪ Verifies patient's health insurance plan. | ▪ 140: Register Patient<br>▪ 145: Verify Insurance Plan |
| Hospital ID Card | ▪ Issues hospital card. | ▪ 130: Receive Patient<br>▪ 140: Register Patient<br>▪ 144: Issue Hospital Card |
| Medical Service | ▪ Knows medical service. | ▪ 100: Refer Patient<br>… |
| Patient | ▪ Knows patient's personal data.<br>▪ Knows patient's contact data.<br>▪ Knows patient's appointments.<br>▪ Knows patient's insurance data.<br>▪ Knows patient's financial data.<br>▪ Knows patient's referral source.<br>▪ Knows patient next of kin.<br>▪ Issues hospital card. | ▪ 100: Refer Patient<br>▪ 120: Make Appointment<br>… |
| Patient Statement (Bill) | ▪ Knows patient open (unpaid) billing items.<br>▪ Knows paid items from the last statement.<br>▪ Prints billing statement. | ▪ 160: Print Patient Statement<br>▪ 190: Resolve Patient Billing Issue |
| Payment | ▪ Knows the payment credited to patient's account. | ▪ 190: Resolve Patient Billing Issue |
| Referral Source | ▪ Knows the referral source. | ▪ 120: Make Appointment |
| Hospital Schedule | ▪ Knows appointments.<br>▪ Knows openings for medical services. | ▪ 120: Make Appointment |

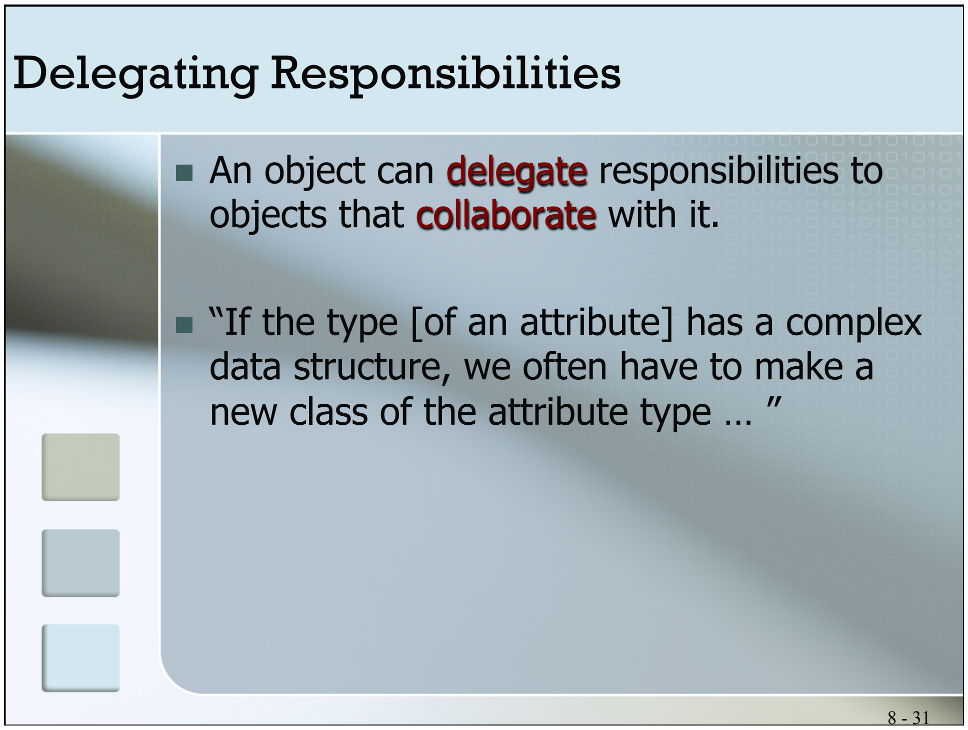
8 - 29



## Elaborating Classes

- To fully define a class or an object is to define responsibilities in detail.
- Patient class has a number of obligations:
  - Knows patient's personal data.
  - Knows patient's contact data.
  - Knows patient's appointments.
  - Knows patient's insurance data.
  - Knows patient's financial data.
  - Knows patient's referral source.
  - Knows patient's medical history.
  - Knows patient's next of kin.
  - Issues hospital card.

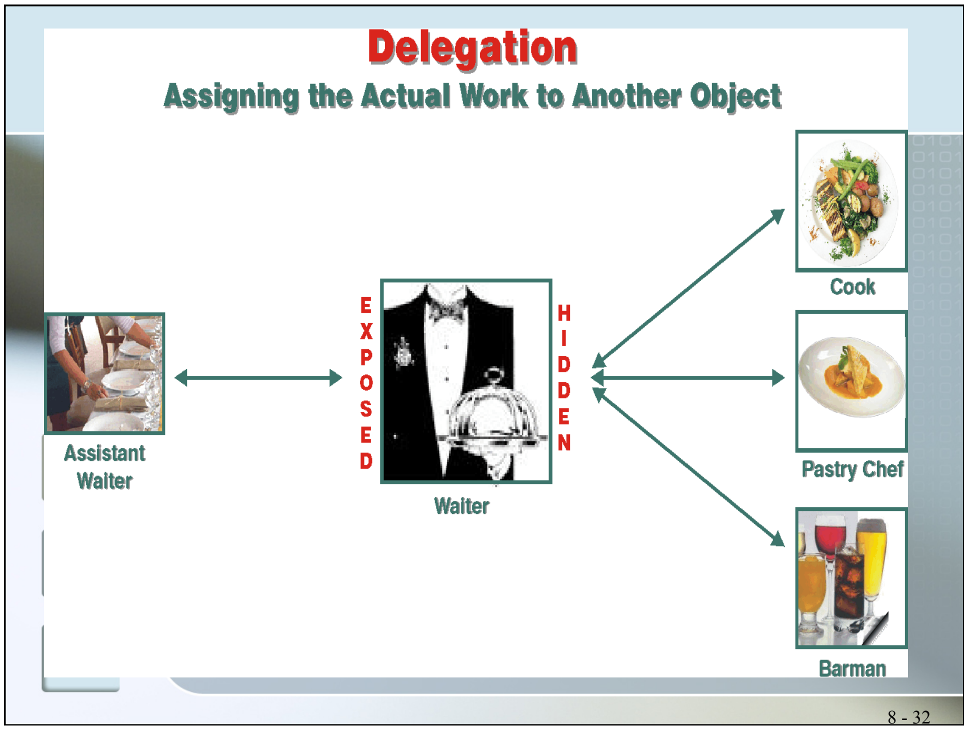
8 - 30



## Delegating Responsibilities

- An object can **delegate** responsibilities to objects that **collaborate** with it.
- "If the type [of an attribute] has a complex data structure, we often have to make a new class of the attribute type … "

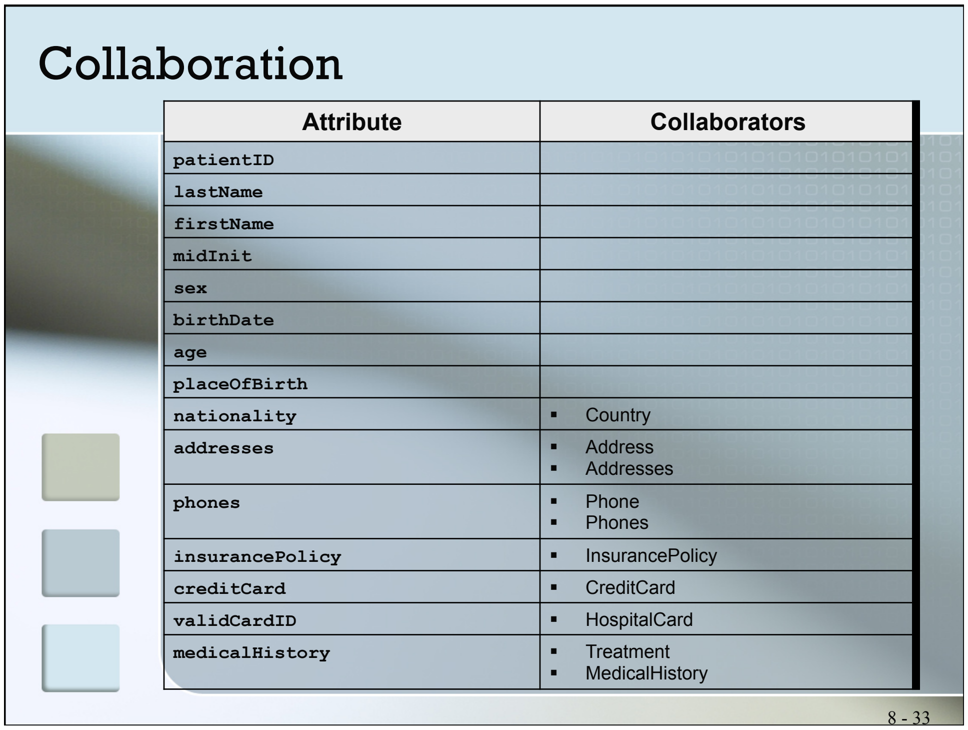
8 - 31



## Delegation

**Assigning the Actual Work to Another Object**

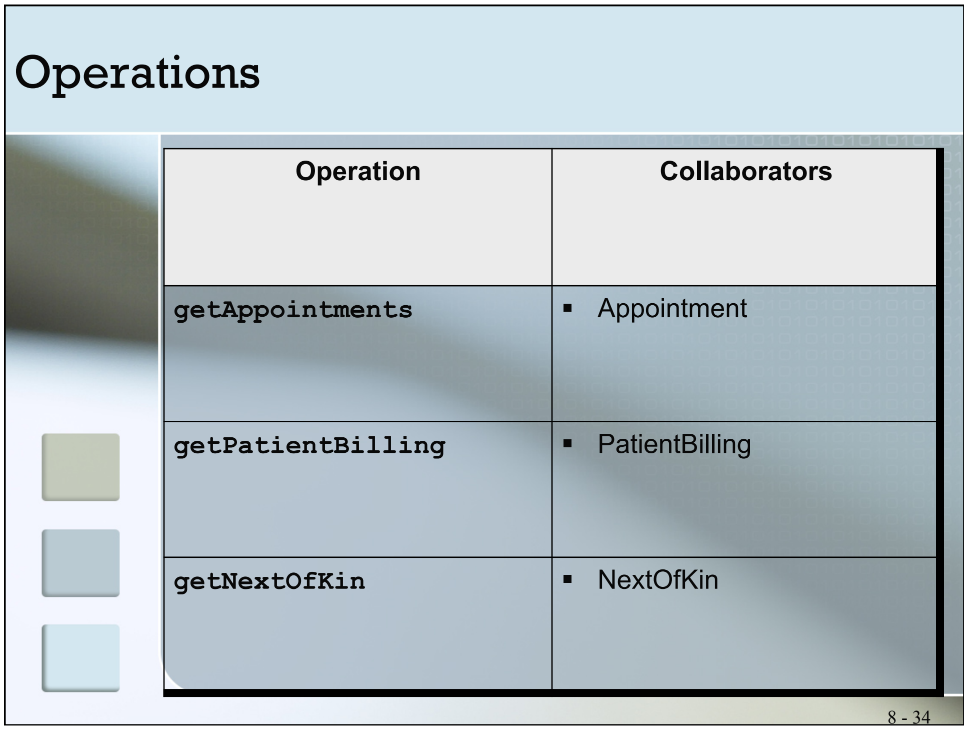
8 - 32



## Collaboration

| Attribute | Collaborators |
|---|---|
| `patientID` | |
| `lastName` | |
| `firstName` | |
| `midInit` | |
| `sex` | |
| `birthDate` | |
| `age` | |
| `placeOfBirth` | |
| `nationality` | ▪ Country |
| `addresses` | ▪ Address<br>▪ Addresses |
| `phones` | ▪ Phone<br>▪ Phones |
| `insurancePolicy` | ▪ InsurancePolicy |
| `creditCard` | ▪ CreditCard |
| `validCardID` | ▪ HospitalCard |
| `medicalHistory` | ▪ Treatment<br>▪ MedicalHistory |

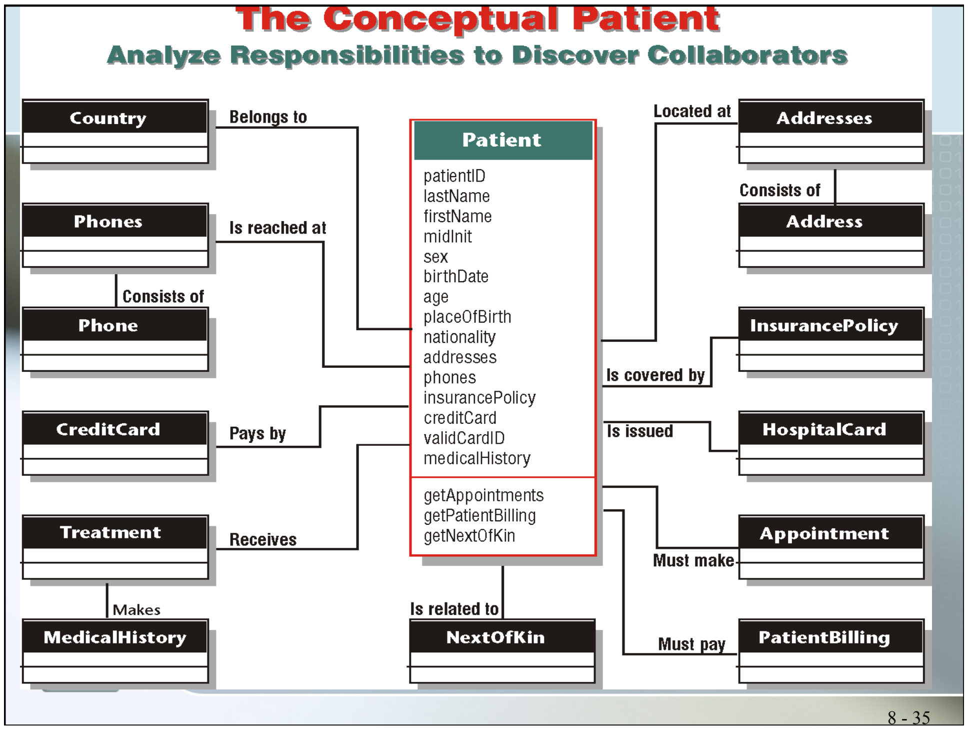
8 - 33



## Operations

| Operation | Collaborators |
|---|---|
| `getAppointments` | ▪ Appointment |
| `getPatientBilling` | ▪ PatientBilling |
| `getNextOfKin` | ▪ NextOfKin |

8 - 34



## The Conceptual Patient

### Analyze Responsibilities to Discover Collaborators

8 - 35

## Relationships

- Association
- Aggregation
- Composition

8 - 36

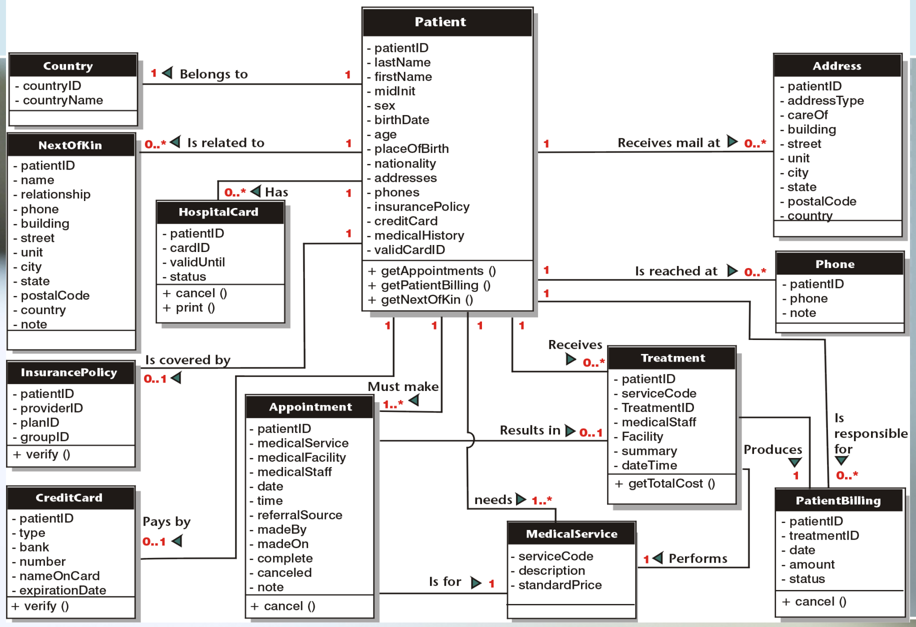
## Association

- Association is a structural relationship that defines the link between objects of one class with the objects of another class

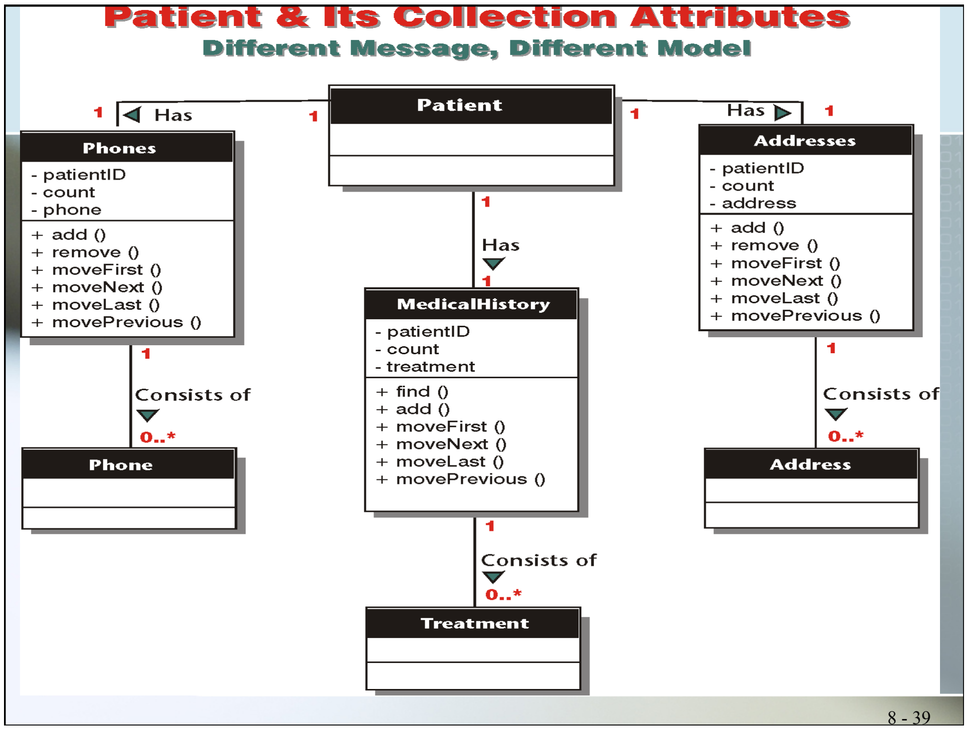
8 - 37

## Patient & Its Associations

### A Diagram Must Have a Point



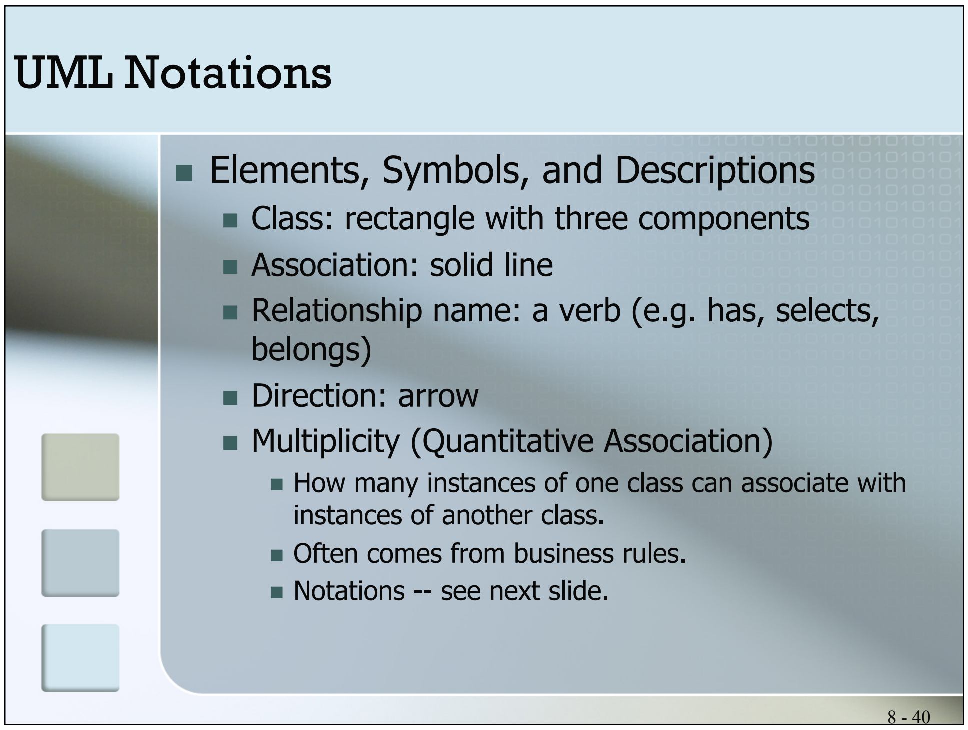
8 - 38



## Patient & Its Collection Attributes

### Different Message, Different Model

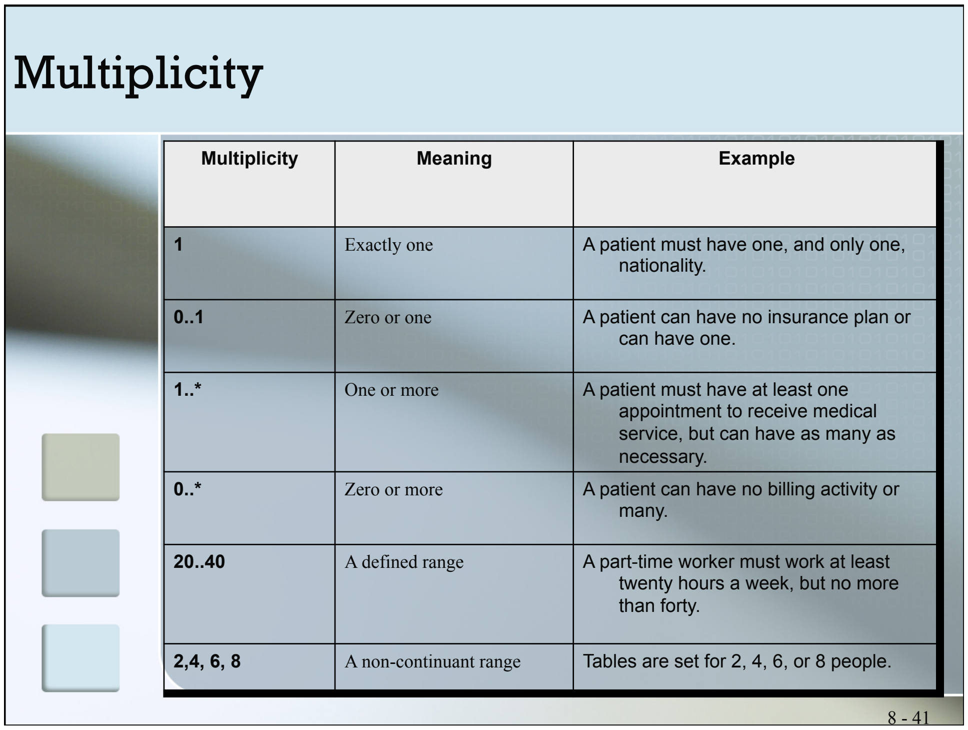
8 - 39



## UML Notations

- Elements, Symbols, and Descriptions
  - Class: rectangle with three components
  - Association: solid line
  - Relationship name: a verb (e.g. has, selects, belongs)
  - Direction: arrow
  - Multiplicity (Quantitative Association)
    - How many instances of one class can associate with instances of another class.
    - Often comes from business rules.
    - Notations -- see next slide.

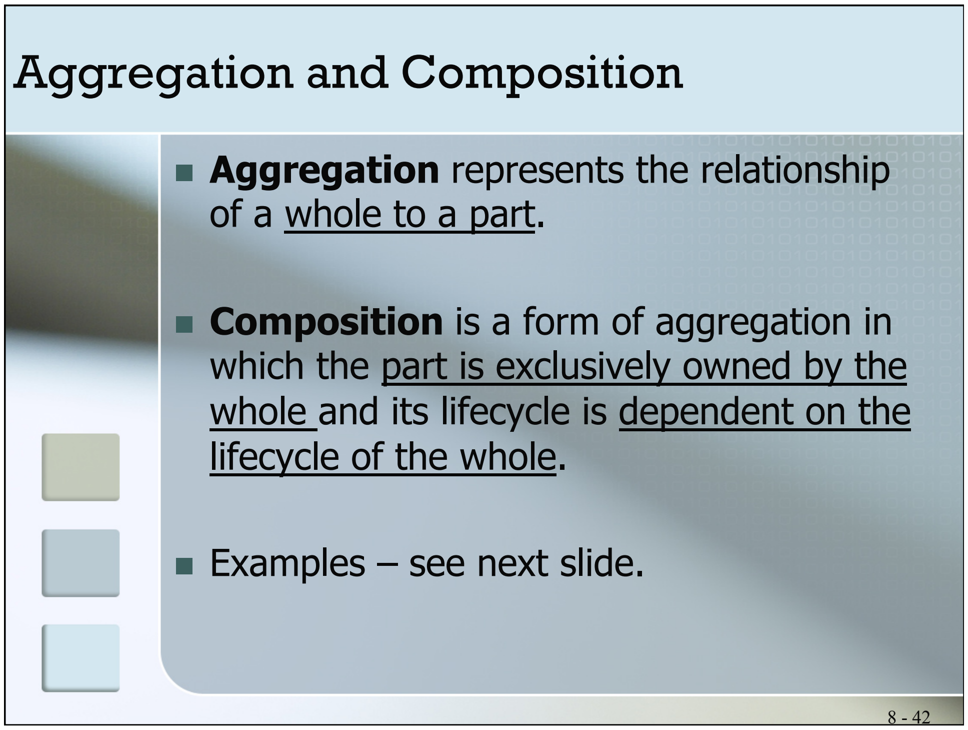
8 - 40



## Multiplicity

| Multiplicity | Meaning | Example |
|---|---|---|
| 1 | Exactly one | A patient must have one, and only one, nationality. |
| 0..1 | Zero or one | A patient can have no insurance plan or can have one. |
| 1..* | One or more | A patient must have at least one appointment to receive medical service, but can have as many as necessary. |
| 0..* | Zero or more | A patient can have no billing activity or many. |
| 20..40 | A defined range | A part-time worker must work at least twenty hours a week, but no more than forty. |
| 2,4, 6, 8 | A non-continuant range | Tables are set for 2, 4, 6, or 8 people. |

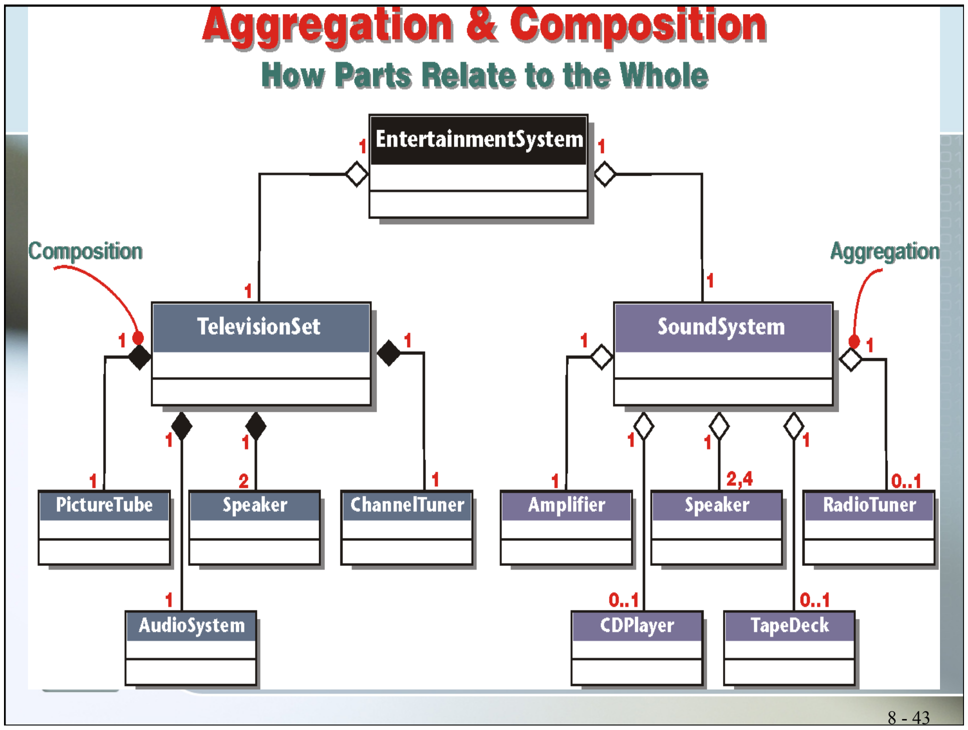
8 - 41



## Aggregation and Composition

- **Aggregation** represents the relationship of a whole to a part.
- **Composition** is a form of aggregation in which the part is exclusively owned by the whole and its lifecycle is dependent on the lifecycle of the whole.
- Examples – see next slide.

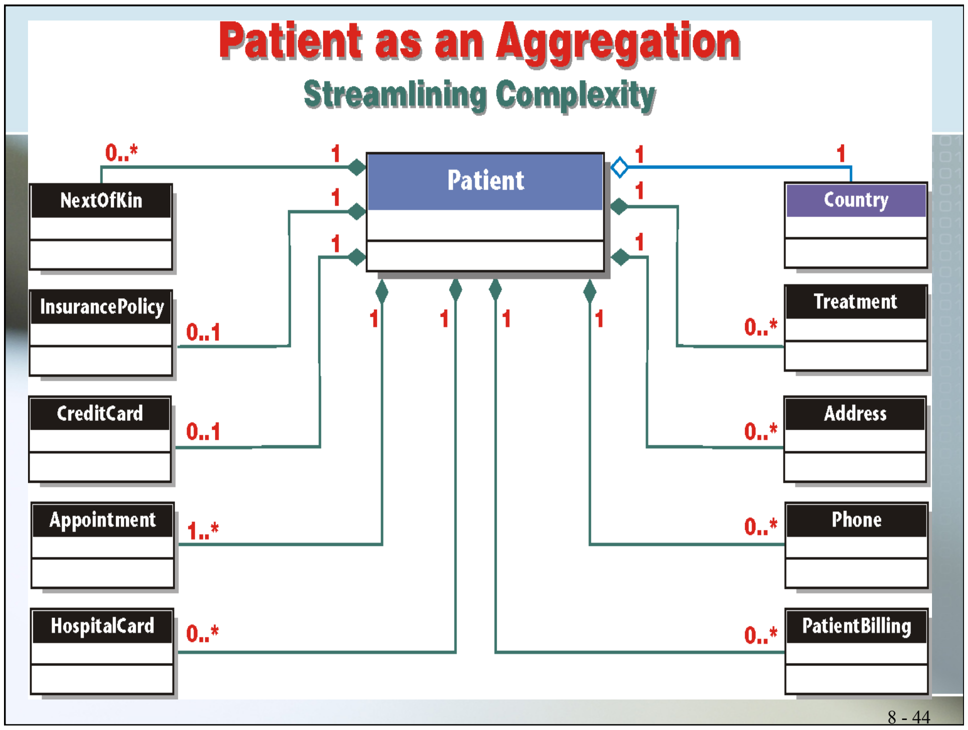
8 - 42



## Aggregation & Composition

### How Parts Relate to the Whole

8 - 43

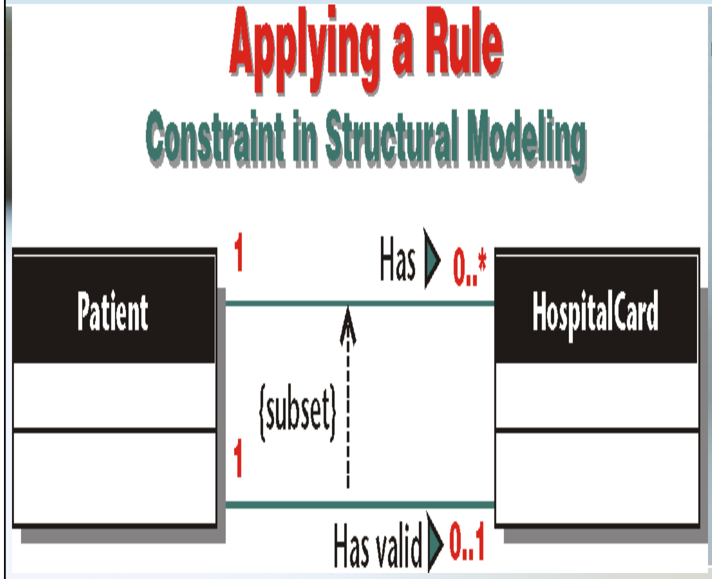


## Patient as an Aggregation

### Streamlining Complexity

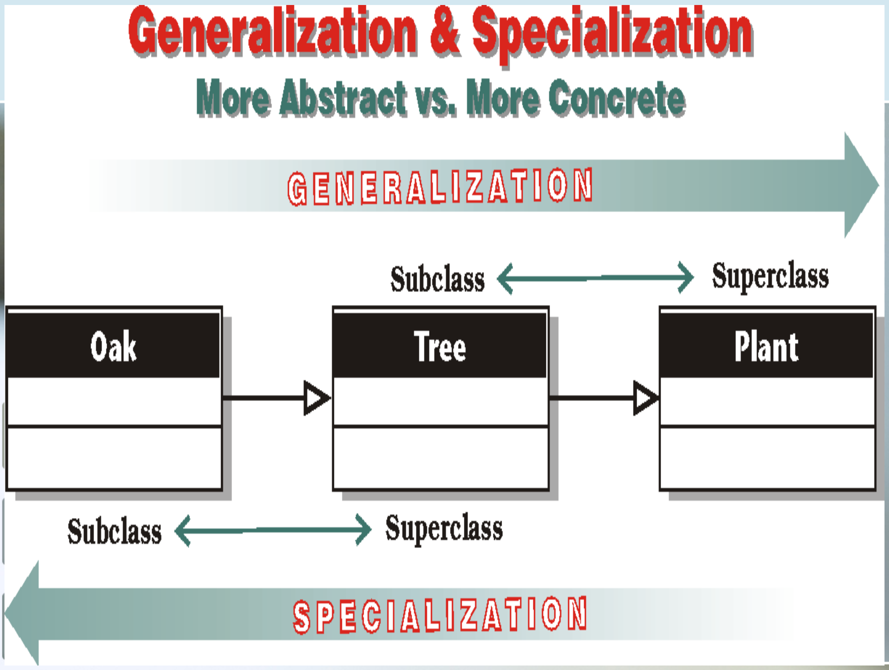
8 - 44

## Constraints

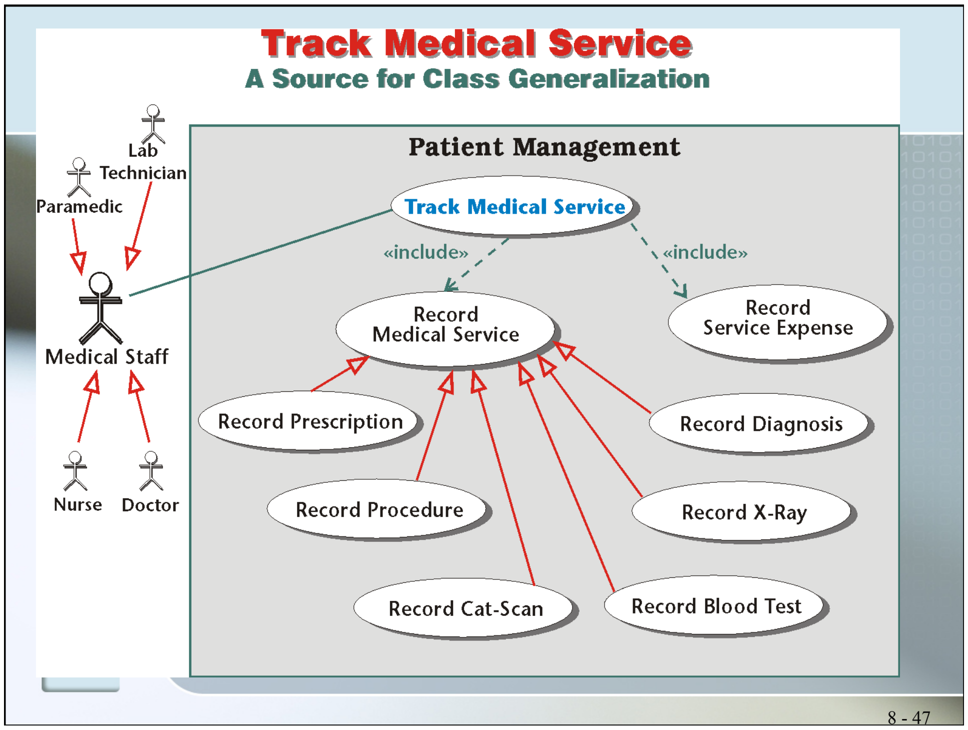


- In structural modeling, constraints are **rules that apply to associations**.

8 - 45



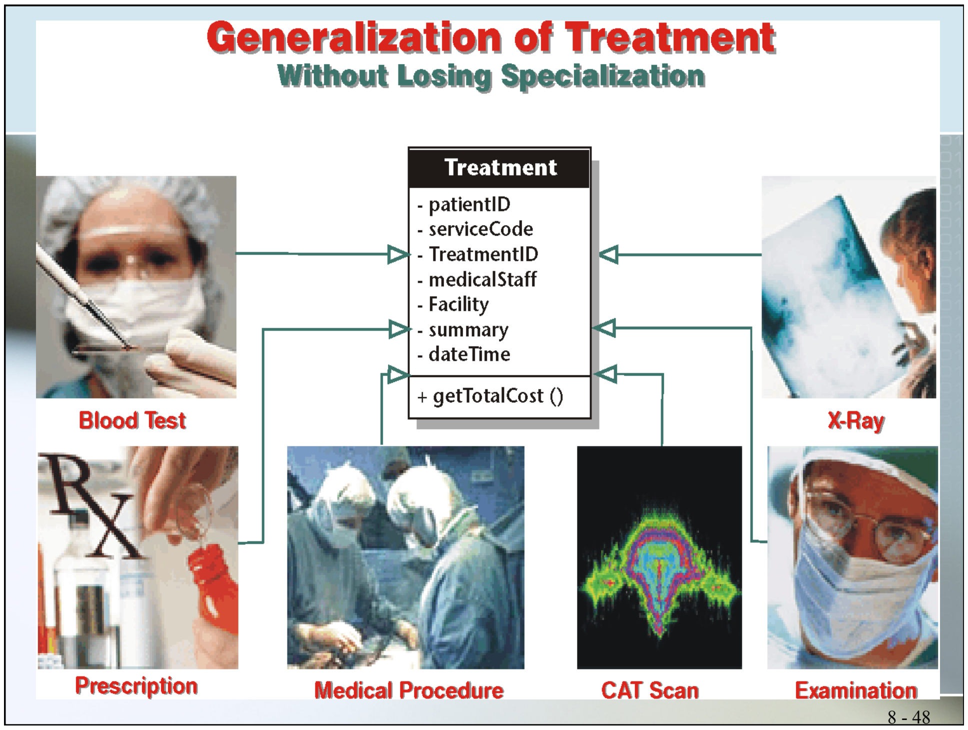
8 - 46



## Track Medical Service

### A Source for Class Generalization

8 - 47



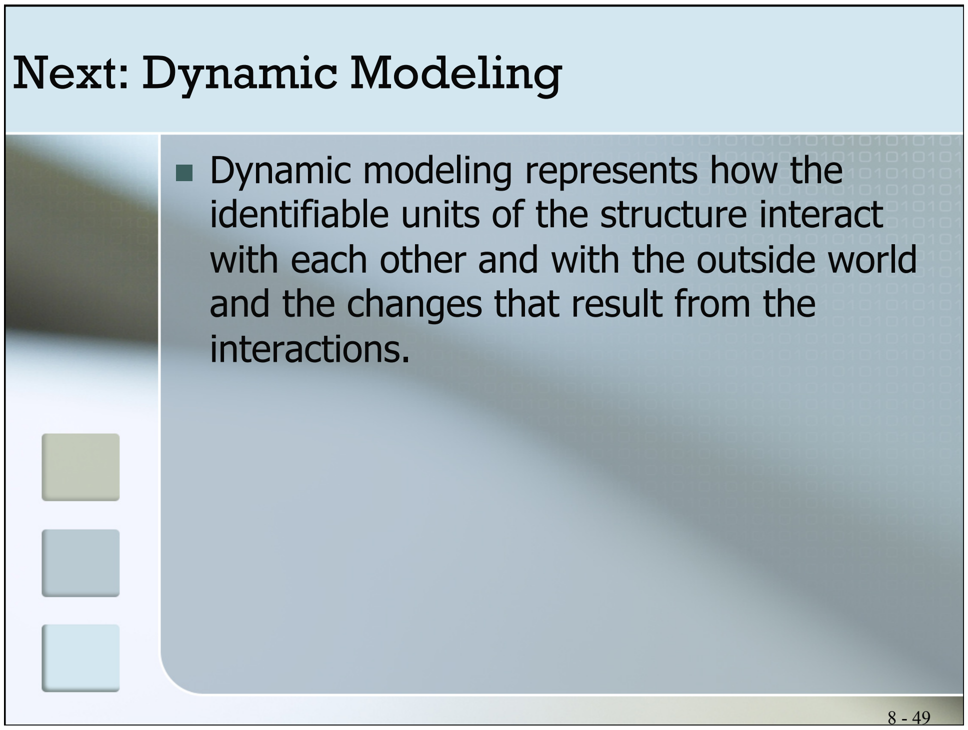
## Generalization of Treatment

### Without Losing Specialization

Treatment
- patientID
- serviceCode
- TreatmentID
- medicalStaff
- Facility
- summary
- dateTime

+ getTotalCost ()

8 - 48



## Next: Dynamic Modeling

- Dynamic modeling represents how the identifiable units of the structure interact with each other and with the outside world and the changes that result from the interactions.

8 - 49