Chapter 5: Structural Modeling

Objectives

- Understand the rules and style guidelines for creating CRC cards, class diagrams, and object diagrams.
- Understand the processes used to create CRC cards, class diagrams, and object diagrams.
- Be able to create CRC cards, class diagrams, and object diagrams.
- Understand the relationship among structural models.
- Understand the relationship between structural and functional models.

Introduction

- Functional models represent system behavior
- Structural models represent system objects and their relationships:
  - People
  - Places
  - Things

Introduction

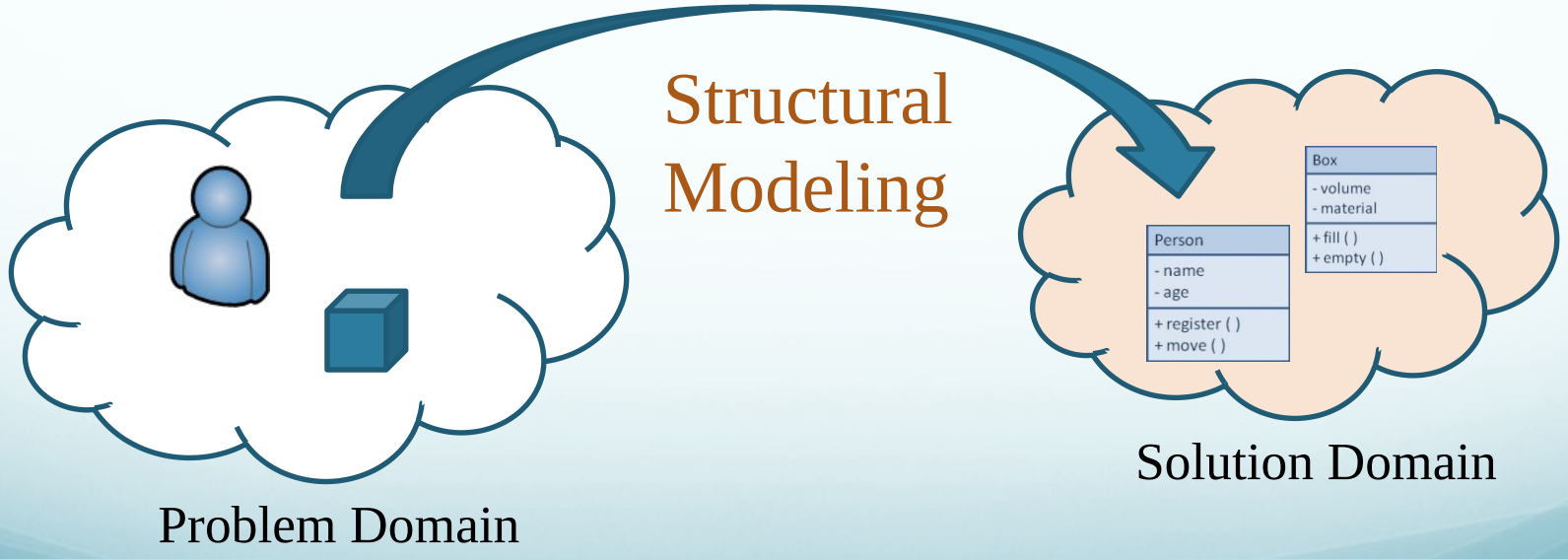
- People, places and things are
  - objects
  - which have attributes
  - and operations

Structural Models

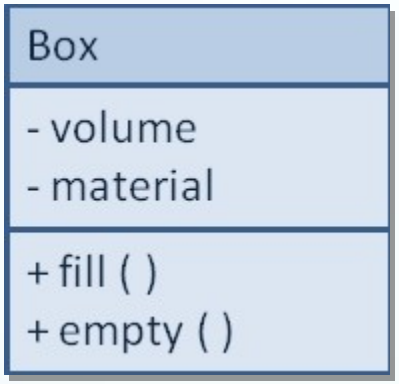
- Drawn using an iterative process
  - First drawn in a conceptual, business-centric way
  - Then refined in a technology-centric way describing the actual databases and files
  - More and more detail is added in each iteration
- Create a vocabulary for analysts & users
  - Allows effective communication between analysts & users

Structural Models

Main goal: to discover the key data contained in the problem domain and to build a structural model of the objects



Classes, Attributes, & Operations



- Classes
  - Templates for instances of people, places, or things
- Attributes
  - Properties that describe the state of an instance of a class (an object)
- Operations
  - Actions or functions that a class can perform

Relationships

- Describe how classes relate to one another
- Three basic types in UML
  - Generalization
    - Enables inheritance of attributes and operations
    - Represents relationships that are “a-kind-of”
    - Reverse is specialization
  - Aggregation
    - Relates parts to wholes or assemblies
    - Represents relationships that are “a-part-of” or “has-parts”
  - Association
    - Miscellaneous relationships between classes
    - Usually a weaker form of aggregation

Object Identification

- Textual analysis of use-case information
  - Nouns suggest classes
  - Verbs suggest operations
  - Creates a rough first cut to provide an object list
- Brainstorming—people offering ideas
  - Initial list of classes (objects) is developed
  - Attributes, operations and relationships to other classes can be assigned in a second round

Textual Object Identification

Use Case Name: Make Old Patient Appt	ID: 2	Importance Level: Low
Primary Actor: Old Patient	Use Case Type: Detail, Essential	

Stakeholders and Interests:
Old Patient – wants to make, change, or cancel an appointment
Doctor – wants to ensure patient's needs are met in a timely manner

Brief Description: This use case describes how we make an appointment as well as changing or canceling an appointment for a previously seen patient.

Trigger: Patient calls and asks for a new appointment or asks to cancel or change an existing appointment.

Type: External

Relationships:

Association: Old Patient
Include:
Extend: Update Patient Information
Generalization: Manage Appointments

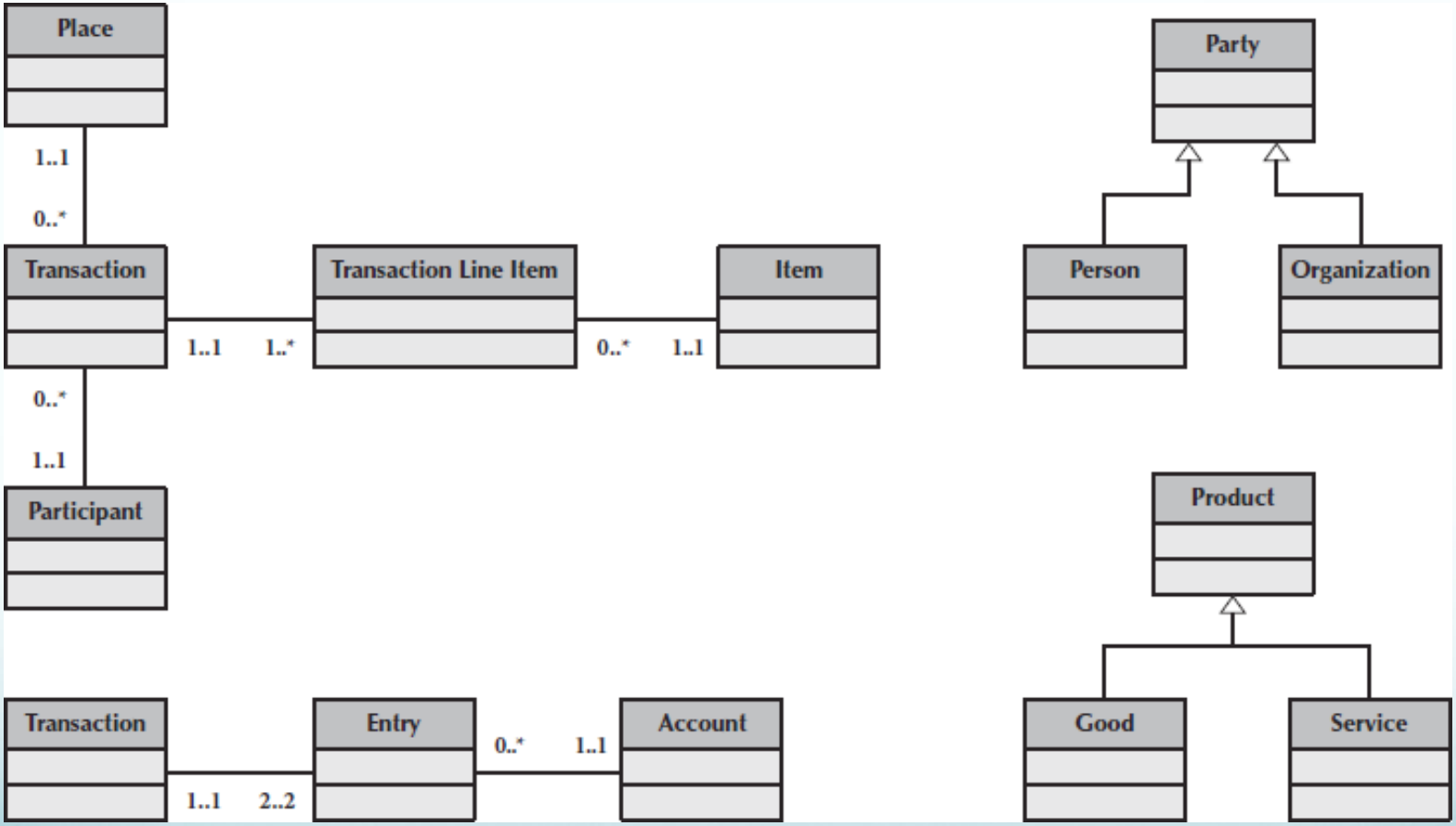
Normal Flow of Events:

1. The Patient contacts the office regarding an appointment.
2. The Patient provides the Receptionist with his or her name and address.
3. If the Patient's information has changed

Object Identification (cont.)

- Common Object Lists
  - Physical things
  - Incidents
  - Roles
  - Interactions
- Patterns
  - Useful groupings of collaborating classes that provide solutions to common problems (are reusable)
  - Developed patterns provide a starting point for work in similar domains

Some Patterns



CRC Cards

- **C**lass-**R**esponsibility-**C**ollaboration
- Index cards used to document the responsibilities and collaborations of a class
- Responsibilities
  - Knowing—what a class must know manifested as attributes
  - Doing—what a class must do manifested later as operations
- Collaboration
  - Objects working together to service a request:
    - Requestor (client)
    - Responder (server)
  - Bound by a contract

Front-Side of a CRC Card

Class Name: Old 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-Side of a CRC Card

Attributes:

Amount (double)

Insurance carrier (text)

Relationships:

Generalization (a-kind-of): Person

Aggregation (has-parts): Medical History

Other Associations: Appointment

CRC Cards & Role-Playing

- An exercise to help discover additional objects, attributes, relationships & operations
- Team members perform roles associated with the actors and objects previously identified
- Utilize activity diagrams to run through the steps in a scenario
  - Identify an important use-case
  - Assign roles based on actors and objects
  - Team members perform each step in the scenario
  - Discover and fix problems until a successful conclusion is reached
  - Repeat for remaining use-cases

Class Diagrams

- A static model that shows classes and their relationships to one another
- Elements
  - Classes
    - Objects within the system (a person, place or thing)
    - Stores and manages information in the system and contains:
      - Attributes—characteristics of the class
      - Operations—activities the class can perform
  - Relationships—the associations between classes
    - Depicted as lines between classes
    - Multiplicity indicates how many of one object is/are associated with other objects

Attributes

- Properties of a class
  - Person: last name, first name, address, etc.
  - Attributes can be derived
    - Preceded with a slash (/)
    - e.g., age is derived from date of birth
- Visibility of an attribute:
  - Restricts access to attributes to ensure consistency
  - Public attributes (+): visible to all classes
  - Private attributes (-): visible only to an instance of the class in which they are defined
  - Protected attributes (#): visible only to an instance of the class in which they are defined and its descendants

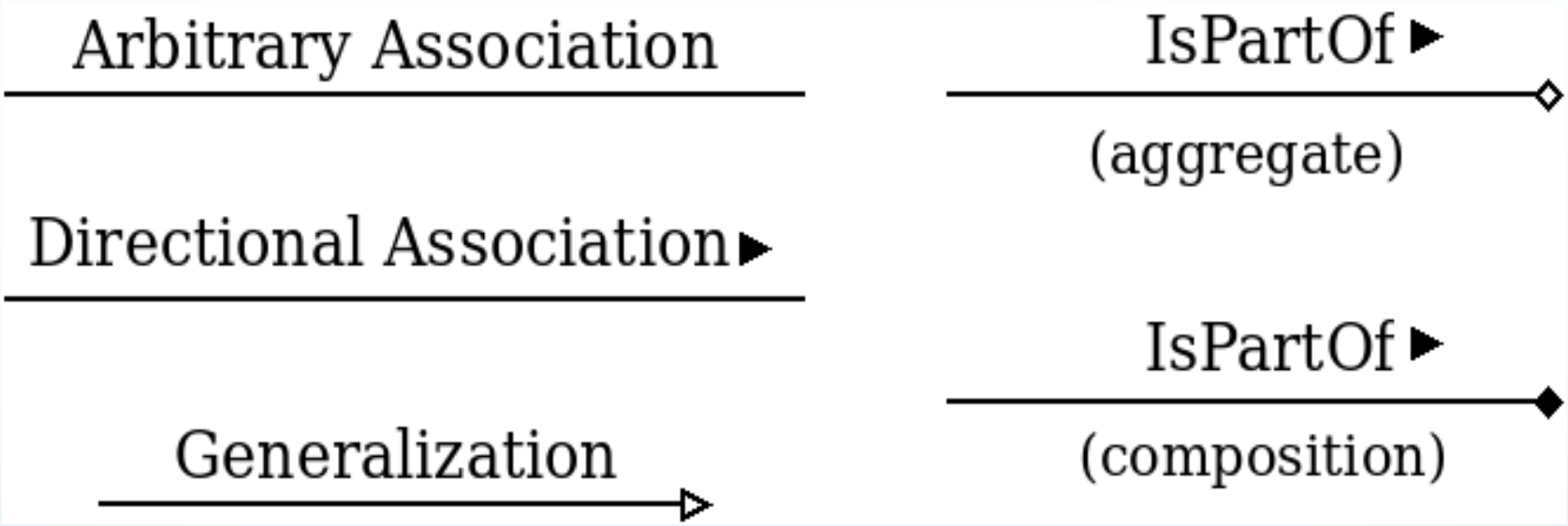
Operations

- Common operations are not shown
  - Create or delete an instance
  - Return or set a value
- Types of operations:
  - Constructor—creates an object
  - Query—makes information about the state of an object available
  - Update—changes values of some or all of an object's attributes
  - Destructor—deletes or removes an object

Relationships

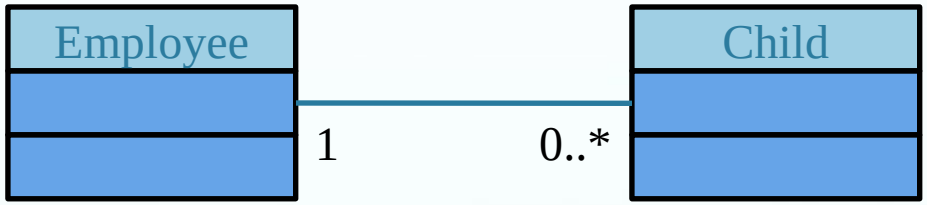
- Denotes associations between classes
  - Depicted with a line labeled with the name of the relationship
  - May be directional (depicted with a triangle; e.g., a patient schedules an appointment)
- Classes may be related to themselves (e.g., employees and managers who may be members of the same class)
- Multiplicity indicates how many of one class are related to another class

Relationships

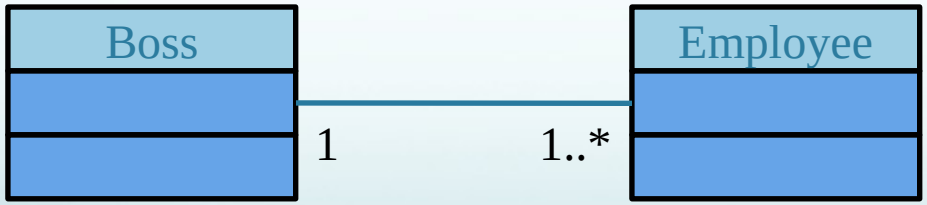


Multiplicities

Exactly one:
A department has one and only one boss



Zero or more:
An employee has zero to many children



One or more:
A boss is responsible for one or more employees

May include ranges or lists, such as 1..2 or 1,3,5

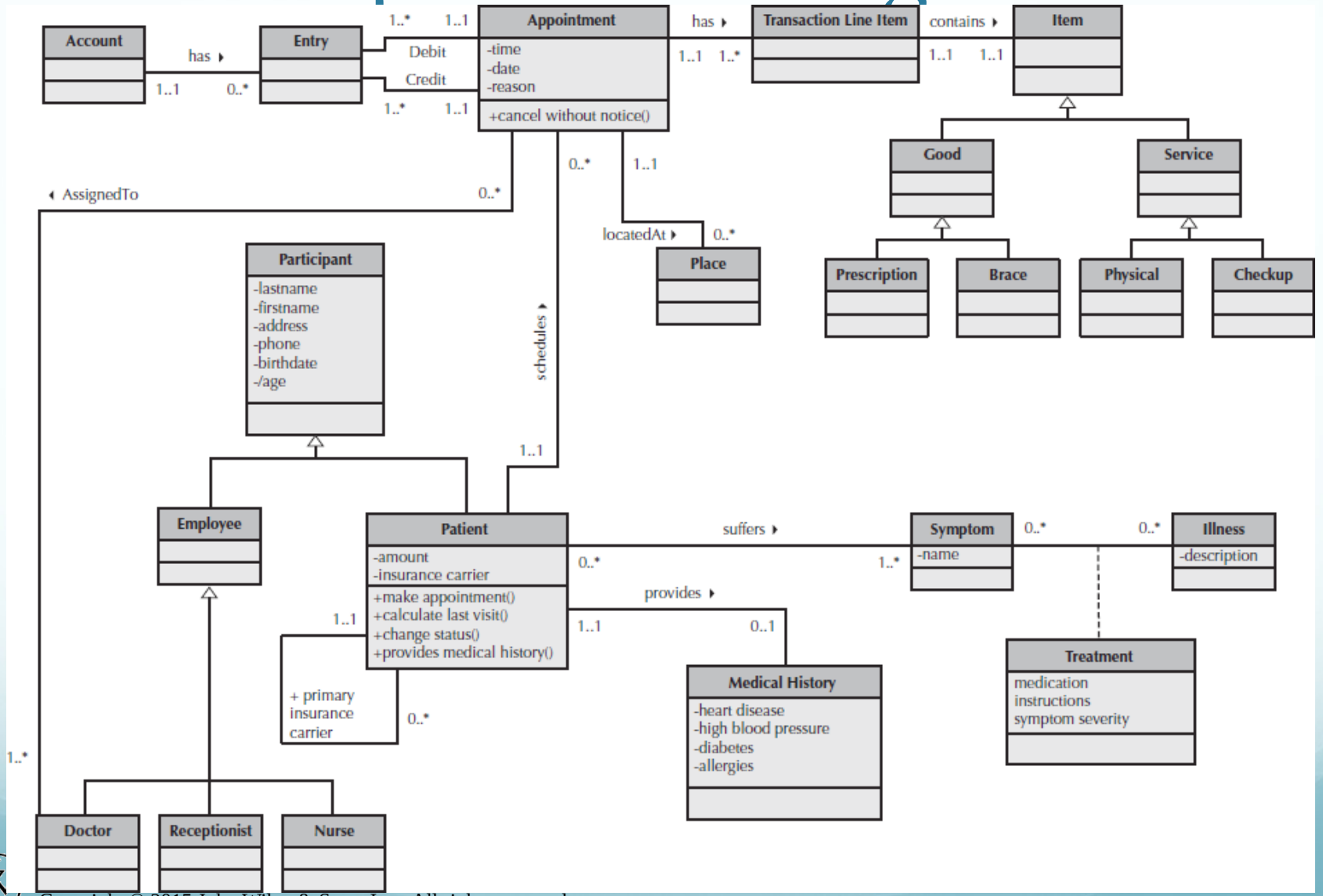
Association Classes

- Common in many-to-many relationships
- Used when attributes about the relationship between two classes needs to be recorded
  - Students are related to courses; a Grade class provides an attribute to describe this relationship
  - Illnesses are related to symptoms; a Treatment class provides an attribute to describe this relationship

Generalization & Aggregation Associations

- Generalization denotes inheritance
  - Properties and operations of the superclass are valid for the sub-class
  - Depicted as a solid line with a hollow arrow pointing at the superclass
- Aggregation denotes a logical “a-part-of” relationship
- Composition denotes a physical “a-part-of” relationship

Sample Class Diagram



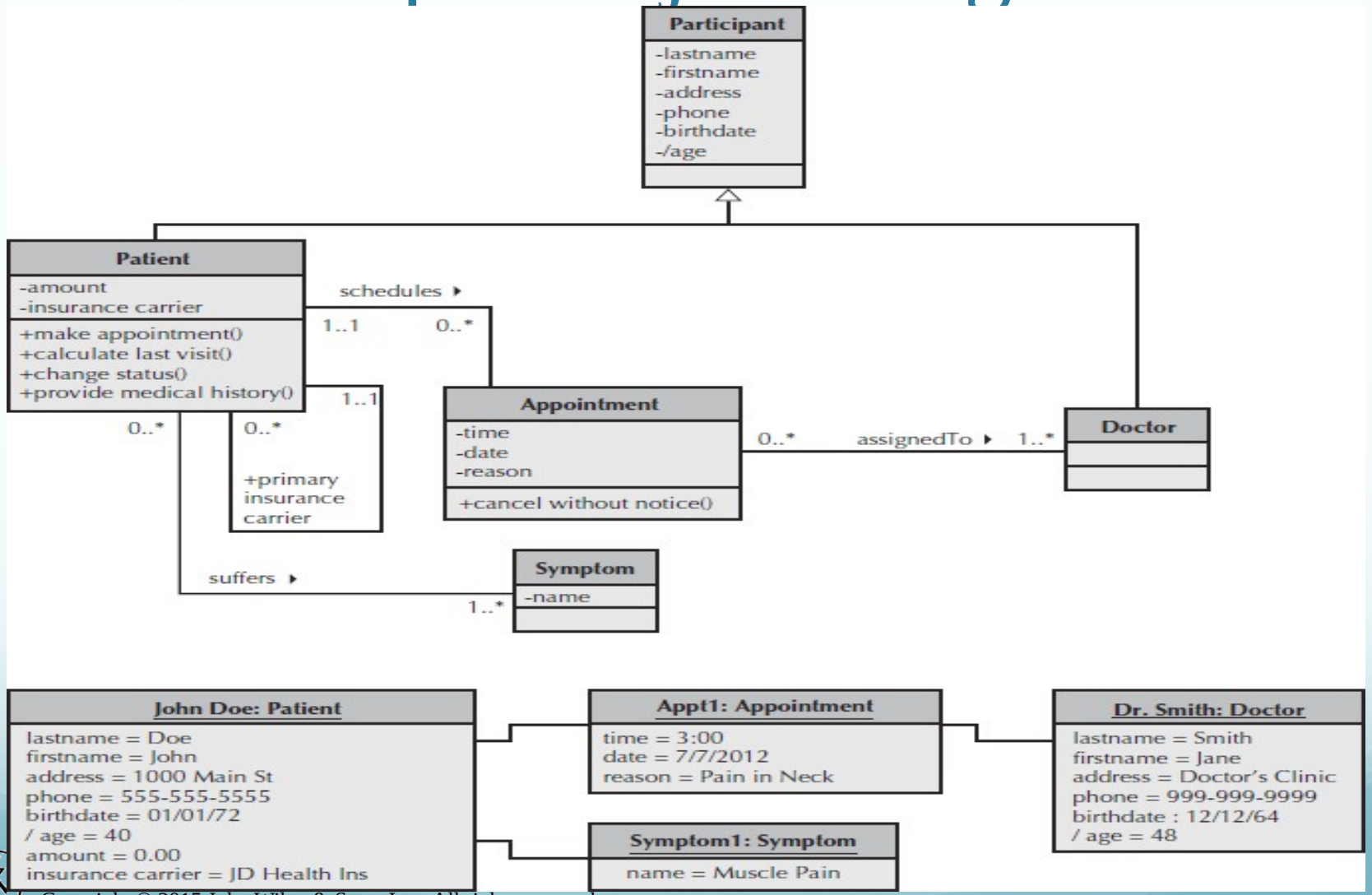
Simplifying Class Diagrams

- Fully populated class diagrams of real-world system can be difficult to understand
- Common ways of simplifying class diagrams:
  - Show only concrete classes
  - The view mechanism shows a subset of classes
  - Packages show aggregations of classes (or any elements in UML)

Object Diagrams

- Class diagrams with instantiated classes
  - Example: instead of a Doctor class, create an actual doctor, say Dr. Smith
  - Place values into each attribute
- Used to discover additional attributes, relationships and/or operations or those that are misplaced

Example Object Diagram



7 Steps to Structural Models

1. Create CRC Cards
2. Review CRC Cards & identify missing objects, attributes, operations and/or relationships
3. Role-play the CRC cards—look for breakdowns & correct; create new cards as necessary
4. Create the class diagram
5. Review the class diagram—remove unnecessary classes, attributes, operations and/or relationships
6. Incorporate patterns
7. Review and validate the model

Verifying & Validating the Model

- Analyst presents to developers & users
  - Walks through the model
  - Provides explanations & reasoning behind each class
- Rules
  1. Each CRC card is associated with a class
  2. Responsibilities on the front of the card are included as operations on the class diagram
  3. Collaborators on the front of the card imply a relationship on the back of the card
  4. Attributes on the back of the card are listed as attributes on the class diagram

Rules for Validating & Verifying the Model (cont.)

5. Attributes on the back of the CRC card each have a data type (e.g., salary implies a number format)
6. Relationships on the back of the card must be properly depicted on the class diagram
   a) Aggregation/Association
   b) Multiplicity
7. Association classes are used only to include attributes that describe a relationship

Summary

- Structural Models
- CRC Cards
- Class Diagrams
- Creating CRC Cards and Class Diagrams