OBJECT-ORIENTED ANALYSIS

EGCO343 SOFTWARE DESIGN

KANAT POOLSAWASD
DEPARTMENT OF COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

DATA MODELING CONCEPTS

- Analysis modeling often begins with data modeling.
- The data object is a representation of any composite information that is processed by software.
- Data attributes define the properties of a data object and take on one of three different characteristics.
  - Name an instance of the data object
  - Describe the instance
  - Make reference to another instance in another table
- Relationship indicate the manner in which data objects are connected to one another.

OBJECT-ORIENTED ANALYSIS

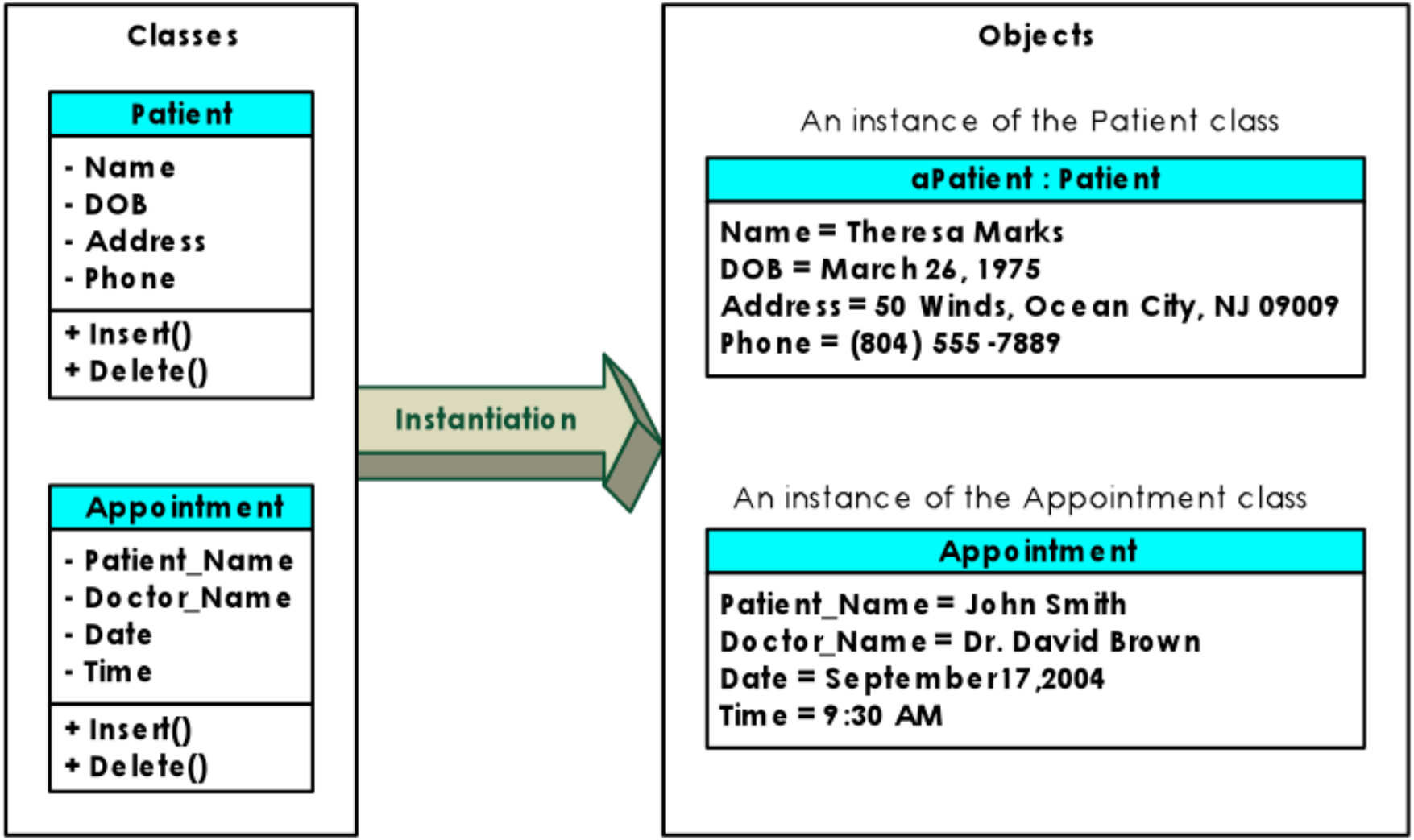
- The intent of object-oriented analysis (OOA) is to define all classes (and the relationships and behavior associated with them) that are relative to the problem to be solved.
- To accomplish this, a number of tasks must occur:
  - Basic user requirements must be communicated between the customer and the software engineer.
  - Classes must be identified.
  - A class hierarchy is defined.
  - Object-to-object relationships should be represented.
  - Object behavior must be modeled.
  - All tasks are reapplied iteratively until the model is complete.

BASIC CHARACTERISTICS

- **Classes and Objects**
  A class is the general template we use to define and create specific instances, or objects

"Every object is associated with a class"

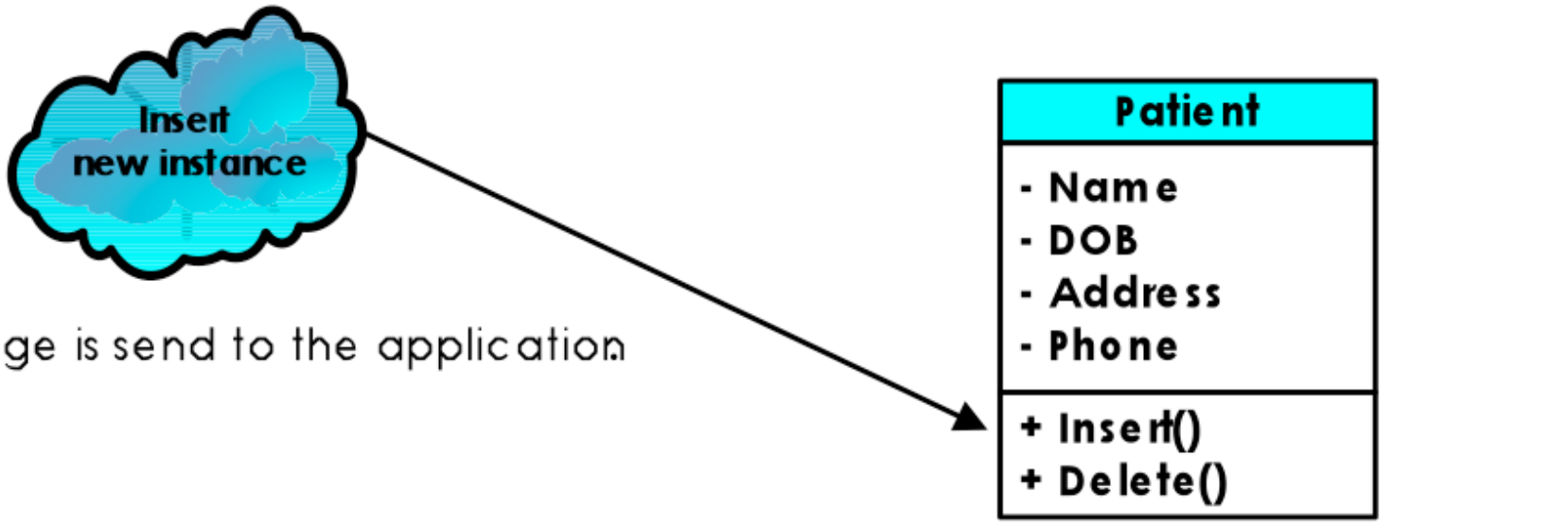
CLASSES AND OBJECTS



METHODS AND MESSAGES (1)

- **Methods** implement an object's behavior. A method is nothing more than an action that an object can perform.
- **Messages** are information sent to objects to trigger methods.

METHODS AND MESSAGES (2)



A message is send to the application

The object's insert method will respond to the message and insert a new patient instance

MESSAGE EXAMPLES

- Call a method associated with buffer, object that returns the value in the buffer.

```
v = getBuffer();
```

- Call a method associated with the object(s)

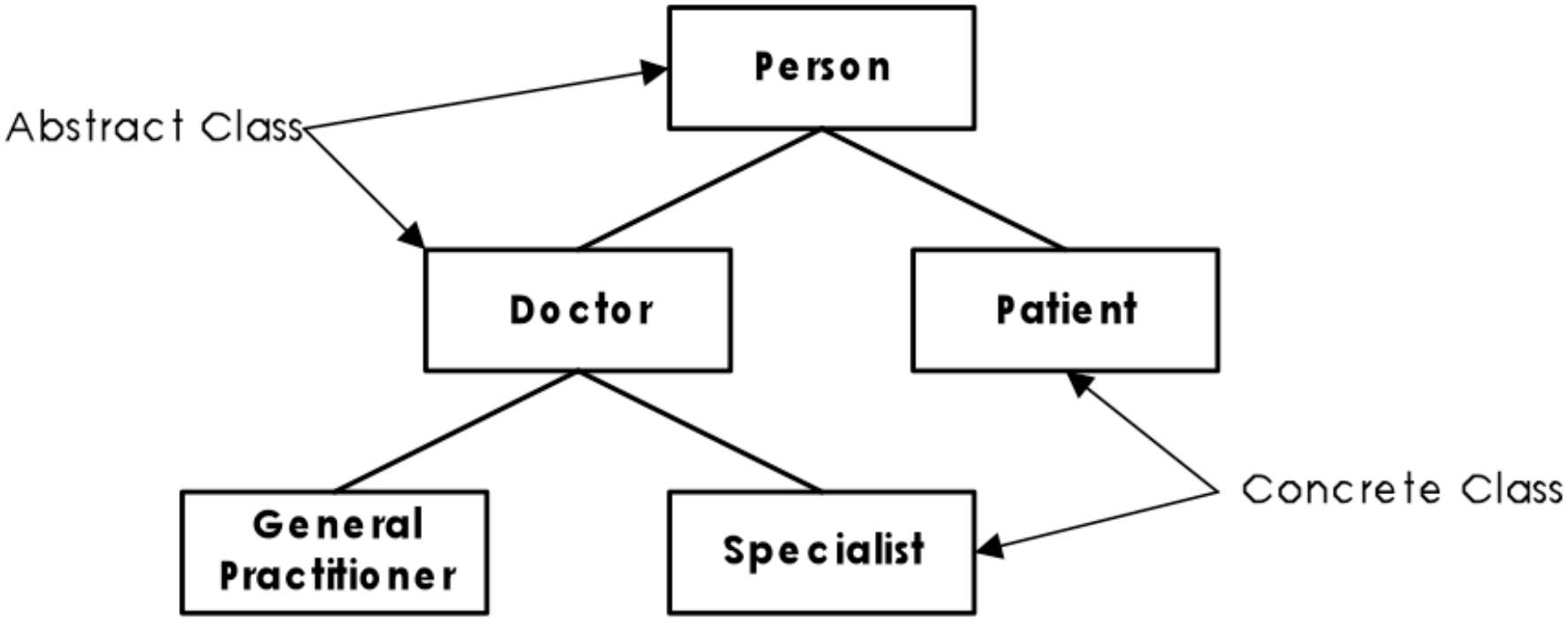
```
thermostat.setTemp(25);
```

ENCAPSULATION & INFORMATION HIDING

- Encapsulation is the practice of bundling the data (fields/attributes) and the methods (functions) that operate on that data together into a single entity (usually a class).
- Information hiding means restricting direct access to the internal details of an object, exposing only what's necessary through a controlled interface.

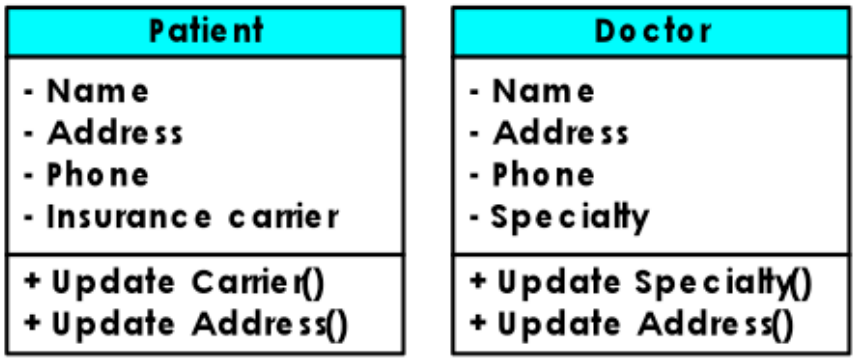
INHERITANCE (1)

Class Hierarchy

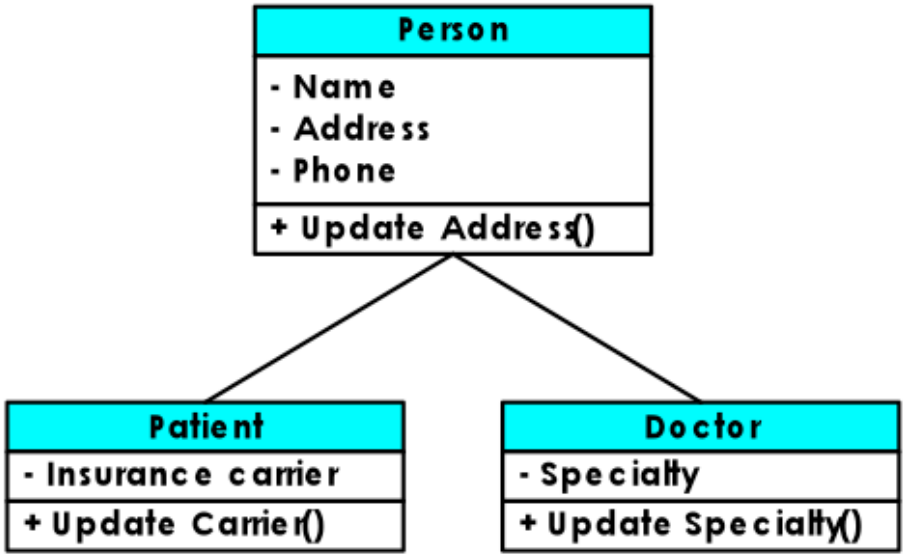


INHERITANCE (2)

Without Inheritance



With Inheritance



POLYMORPHISM & DYNAMIC BINDING

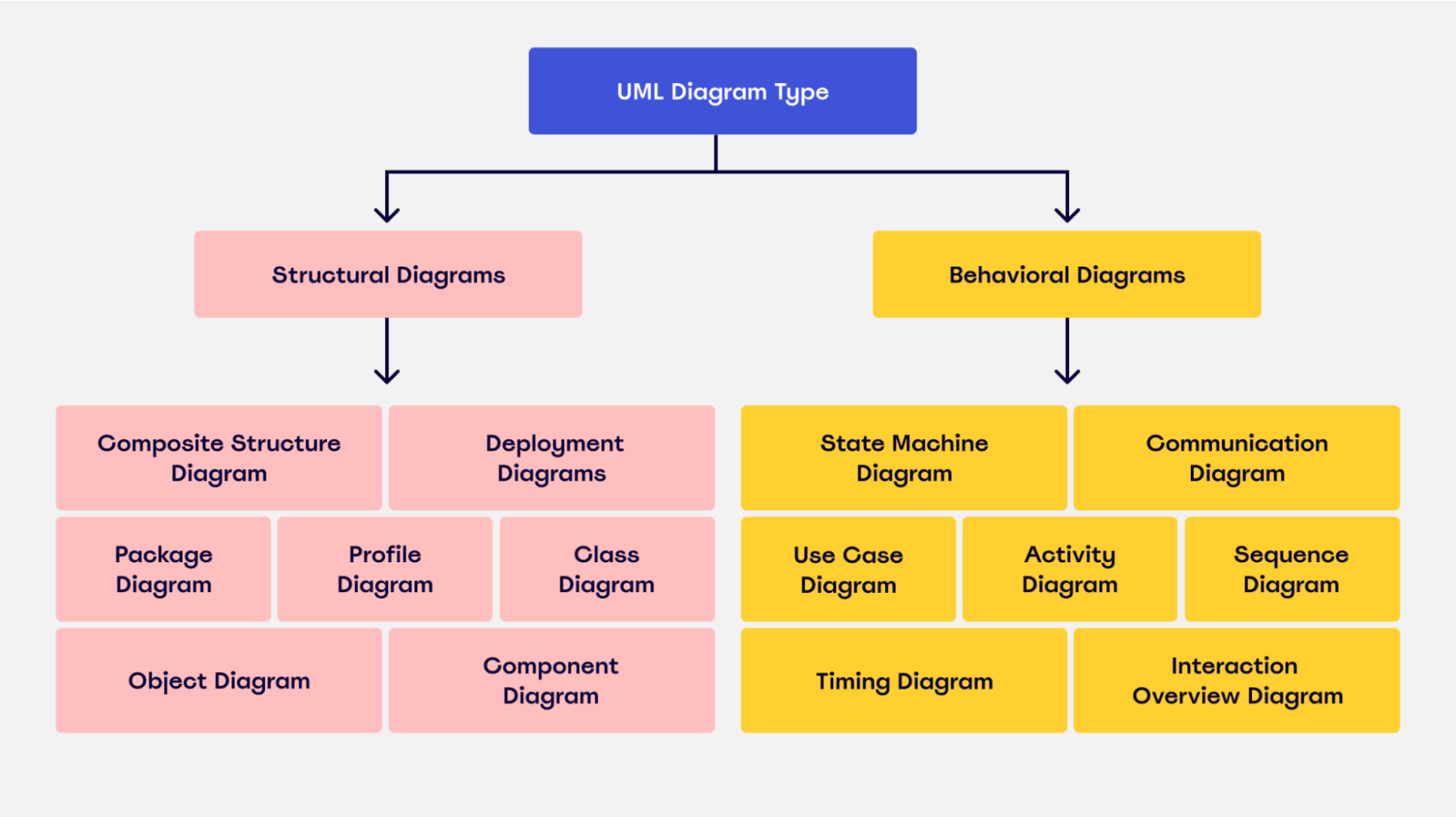
- Polymorphism means "many forms" — the ability for different classes to be treated as instances of the same superclass, with each class providing its own behavior for the same method.
- Dynamic binding is the process where the method that will be executed is determined at runtime based on the actual object type, not the reference type.
- Polymorphism + Dynamic Binding Together
  - Polymorphism = You can call the same method on different object types.
  - Dynamic Binding = The machine (such as JVM) decides at runtime which version of the method to run.

THE UNIFIED MODELING LANGUAGE (1)

- In 1995, Rational Software brought three industry leaders together to create a single approach to object-oriented systems development. Grandy Booch, Ivar Jacobson, and James Rumbaugh worked with others to create a standard set of diagramming techniques known as the Unified Modeling Language (UML).
- The version 2.0 of the UML defines a set of 14 diagramming techniques used to model a system.

THE UNIFIED MODELING LANGUAGE (2)

- UML diagrams are broken into two major grouping



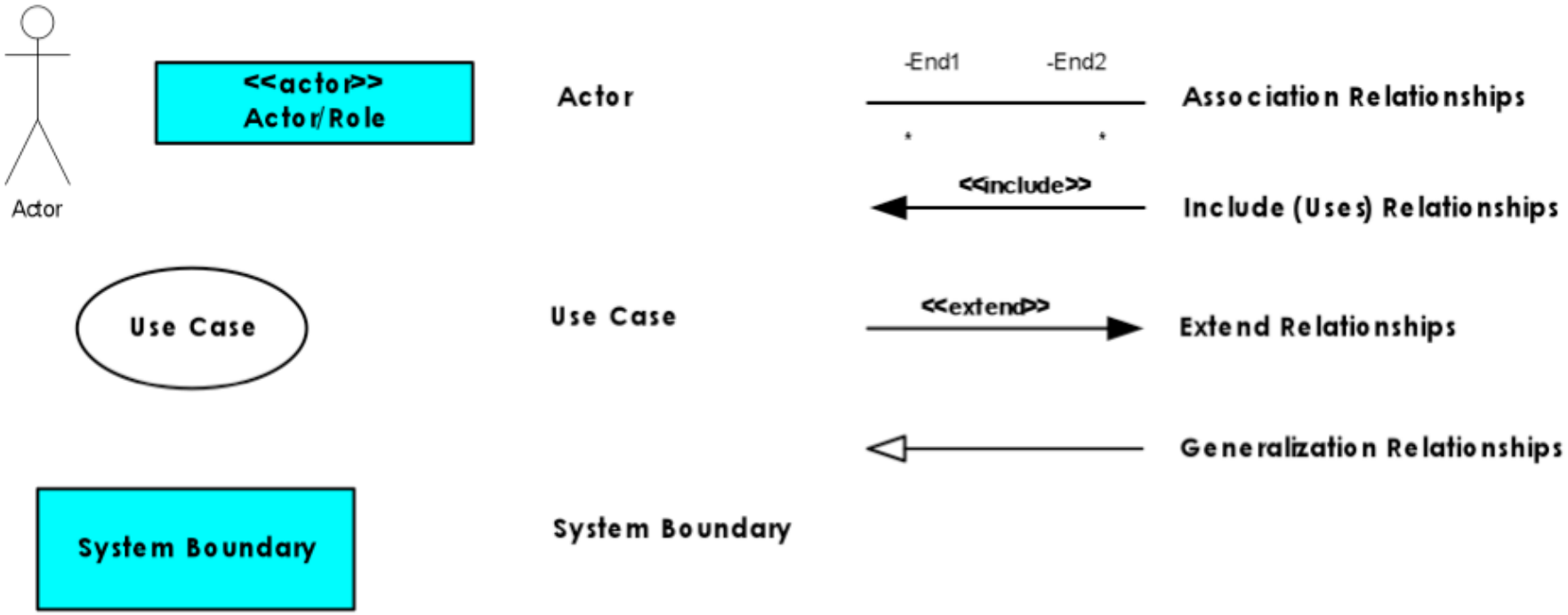
USE CASES

- Use-case diagrams are functional diagrams in that they portray the basic function of the system – that is. What the users can do and how the system should respond to the user's action.
- Creating use case diagrams is a two-step: First, the users work with the project team to write text-based use case description, and second, the project team translate the use case descriptions into formal use case diagrams.
- Use cases are the primary drivers for all of the UML diagramming techniques.

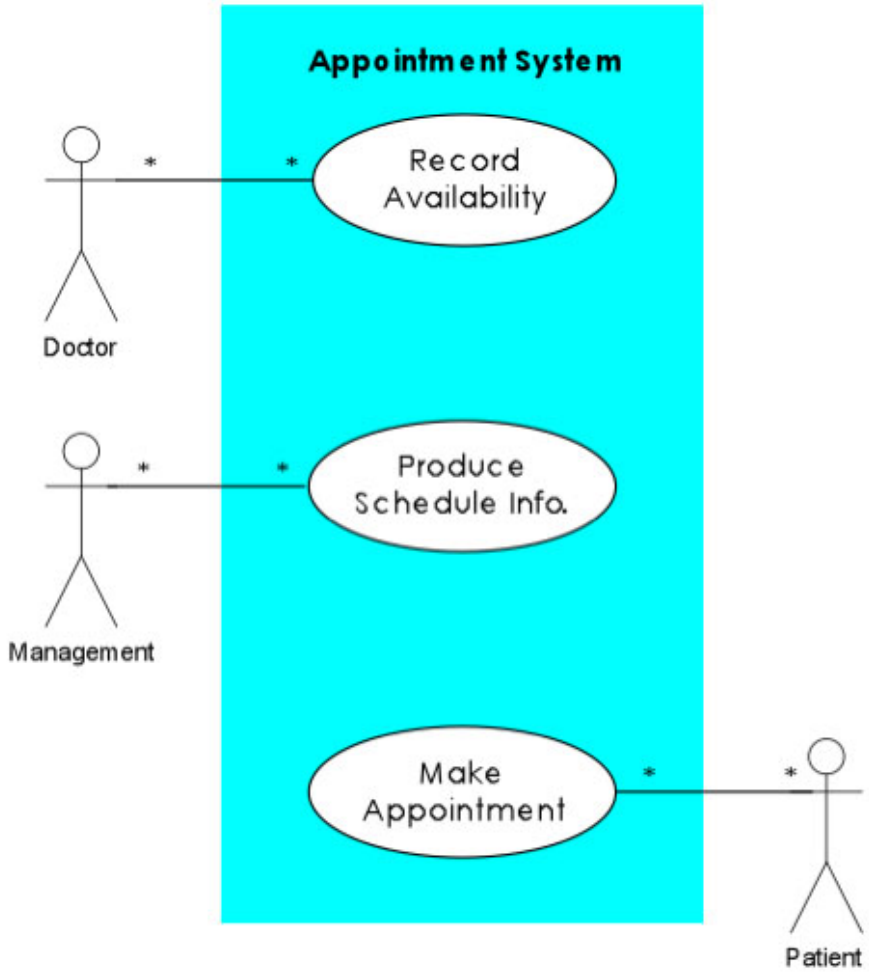
ELEMENTS OF A USE CASE DESCRIPTION (NARRATIVE)

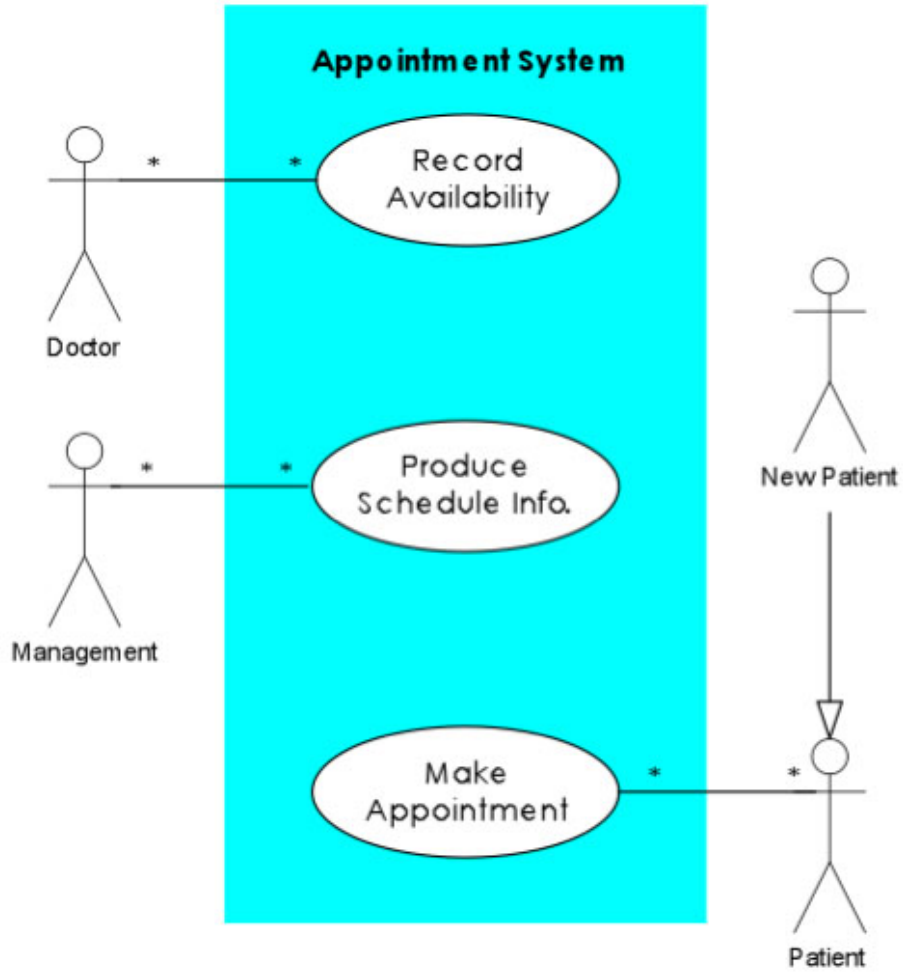
- Overview Information
- Relationships
  - Association
  - Extend
  - Include
  - Generalization
- Flow of Events
  - Normal Flow of Events
  - Sub-flows
  - Alternate or Exceptional Flows
- Optional Characteristics

USE-CASE DIAGRAMS

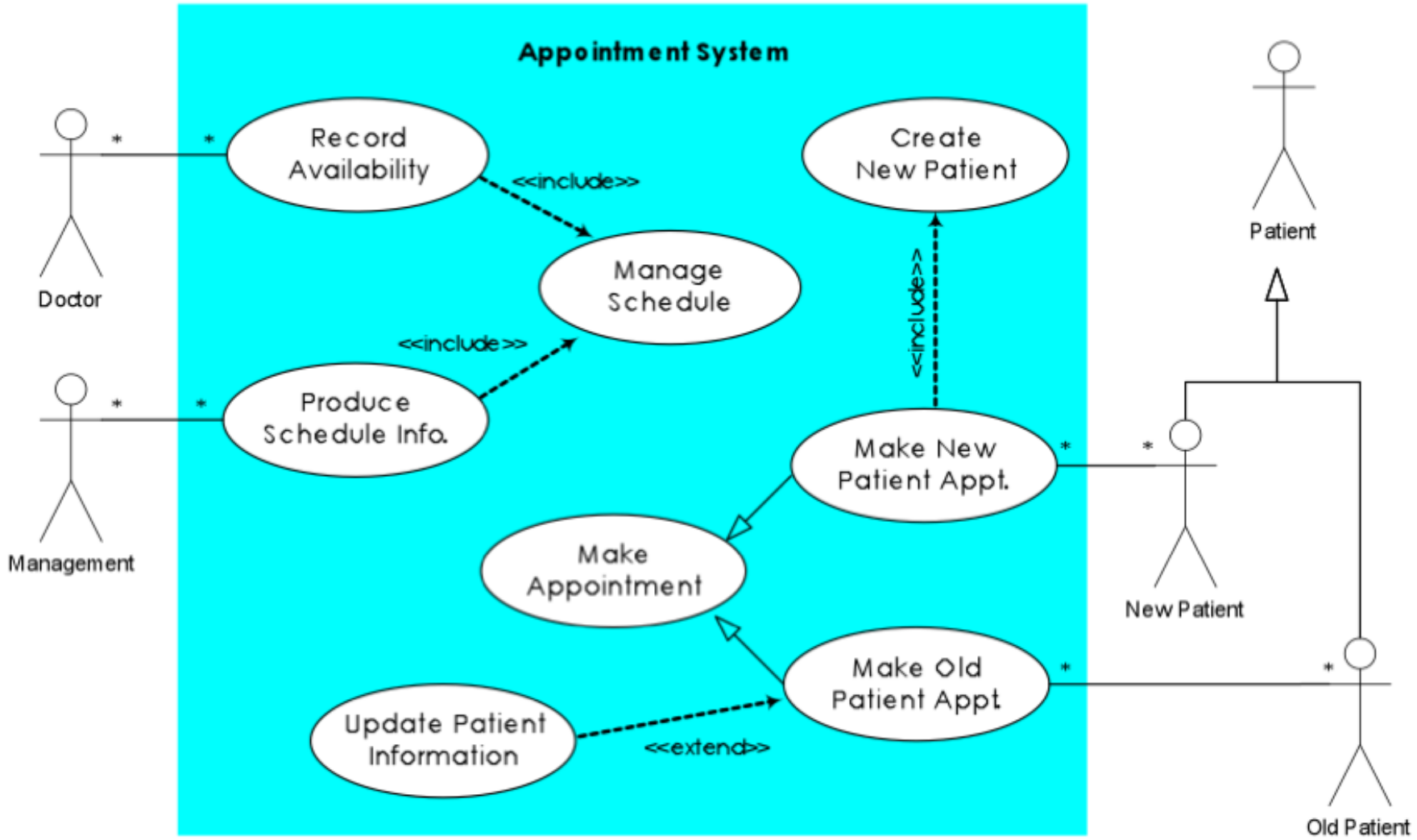


EXAMPLE 1: (1)
USE CASE DIAGRAM FOR APPOINTMENT SYSTEM





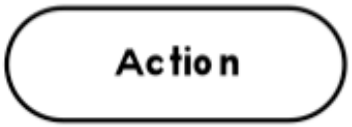
EXAMPLE 1: (2)
USE CASE DIAGRAM FOR APPOINTMENT SYSTEM



WORKSHOP (1)

- Each group create a Use-Case Diagram of the “Room Reservation System” using draw.io.

ACTIVITY DIAGRAM



Action or Activity

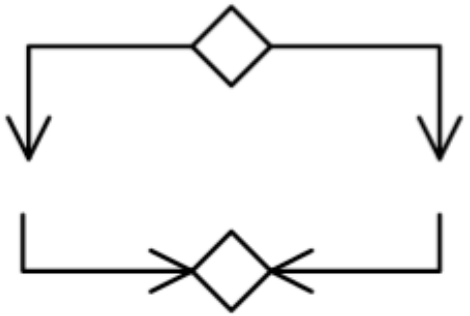
Object Node

Control Flow

Object Flow

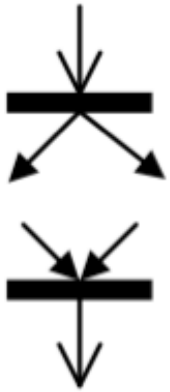
Initial Node

Final-Activity Node



Decision Node

Merge Node

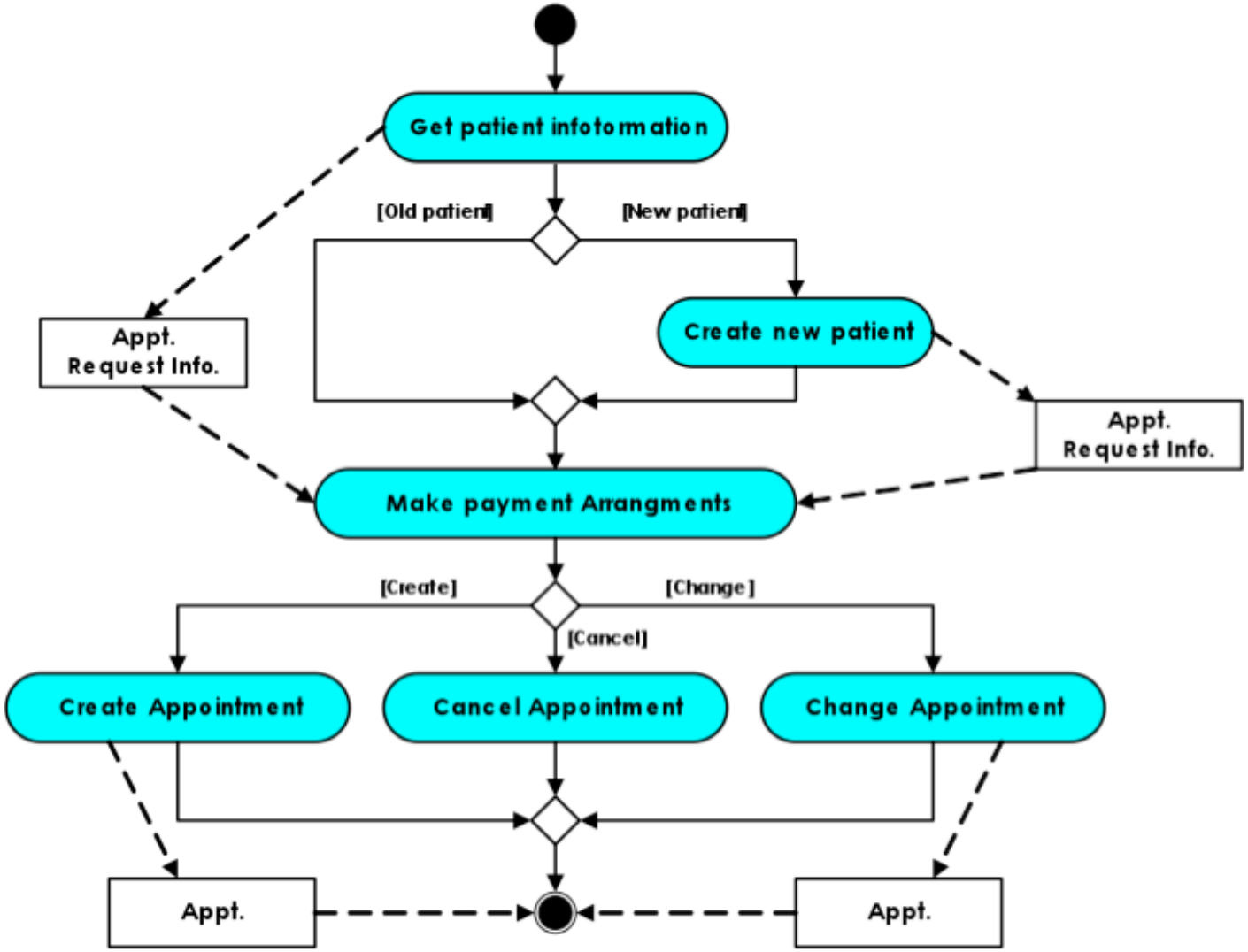


Fork Node

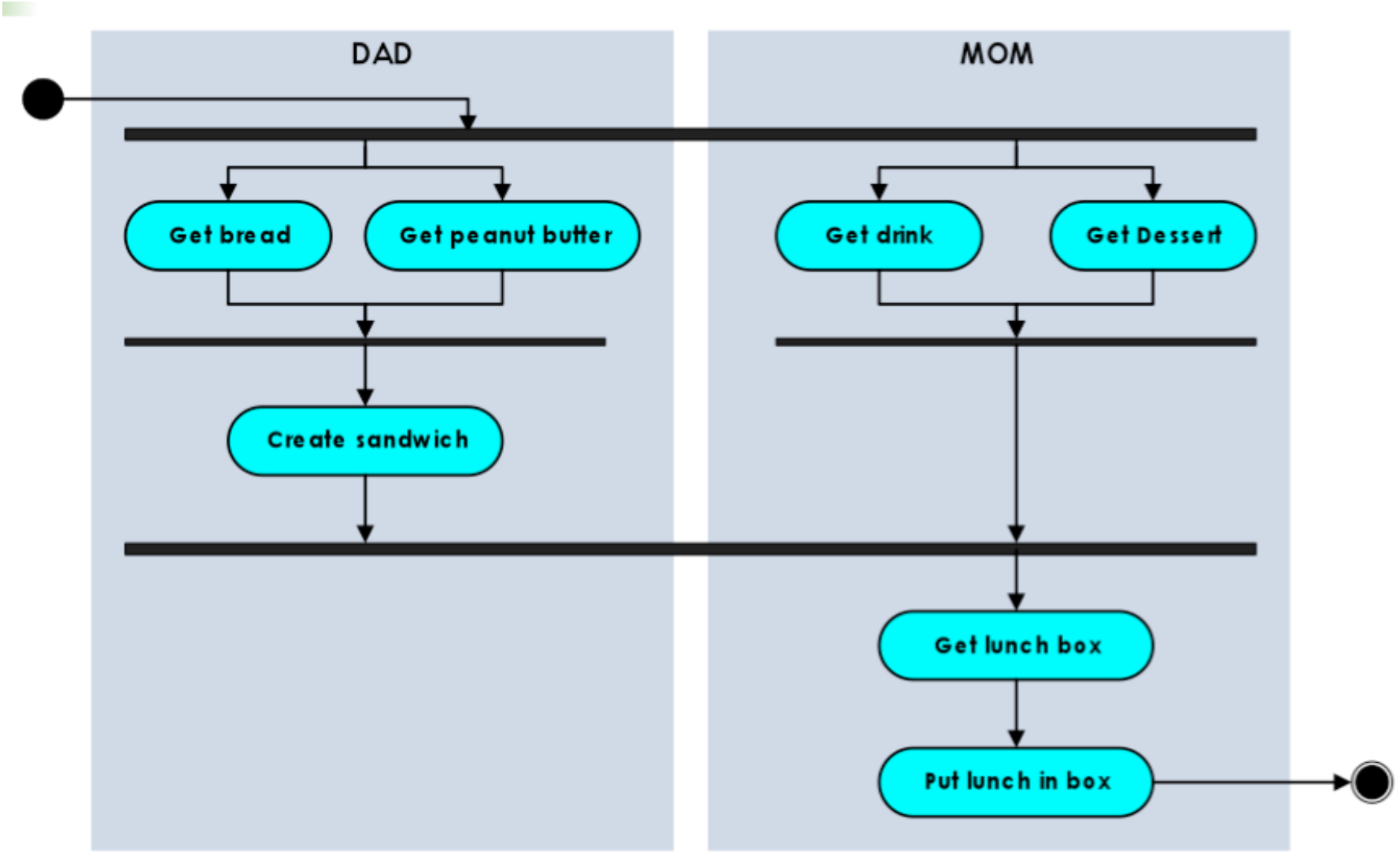
Join Node

Swimlane

EXAMPLE 2:
ACTIVITY DIAGRAM FOR APPOINTMENT SYSTEM



SAMPLE 3:
ACTIVITY DIAGRAM FOR SCHOOL BOX LUNCH



WORKSHOP (2)

- Each group creates an Activity Diagram of the (current) room reservation process, dividing the swim lane into three sections corresponding to the three stakeholders.

STRUCTURAL MODELS

- A structural model is a formal way of representing the objects that are used and created by a business system. It illustrates people, places, or things about which information is captured, and how they are related to each other.
- Classes, Attributes, and Operations
- Relationship
  - Generalisation (a-kind-of)
  - Aggregation (a-part-of)
  - Association Relationships

CLASS-RESPONSIBILITY-COLLABORATION CARDS (CRC CARD)

- Class-Responsibility-Collaboration cards, most commonly called CRC cards, are used to document the responsibilities and collaborations of a class.
- The responsibilities of a class can be divided into two separate types: knowing and doing.
  - **Knowing** Responsibilities are those things that an instance of class must be capable of knowing
  - **Doing** Responsibilities are those things that an instance of a class must be capable of doing

CLASS-RESPONSIBILITY-COLLABORATION CARDS (CRC CARD)

Knowing

Doing

```
class Person {
    String name;
    String surname;
    Date birthdate;

    int calculateAge() {
        // calculating age from DOB
        return ...;
    }
}
```

EXAMPLE CRC CARD (FRONT)

Class Name: Patient	**ID:** 3	**Type:** Concrete
Description: An individual that needs to receive or has received medical attention		**Associated Use Cases:** 2
Responsibilities Make appointment Calculate last visit Change status Provide medical history		**Collaborators** Appointment Medical history

EXAMPLE CRC CARD (BACK)

Attributes: Amount (double) Insurance carrier (text)
Relationships: **Generalization (a-kind-of):** Person **Aggregation (has-parts):** Medical History **Other Associations:** Appointment

CLASS DIAGRAMS

- The class diagram is a static model that shows the classes and the relationships among classes that remain constant in the system over time.

ELEMENTS OF A CLASS DIAGRAM

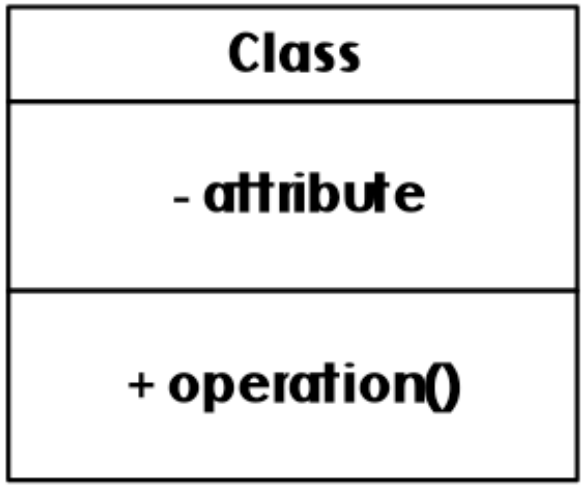
- Attributes are properties of the class about which we want to capture information.
  - Derived: slash (/)
  - Public: (+)
  - Protected: (#)
  - Private: (-)
- Operations are functions or actions that a class can perform

OPERATIONS

- A class can contain three kind of operations:
  - Constructor Operation (create)
  - Query Operation (read)
  - Update Operation (update or delete)

CLASS DIAGRAM SYNTAX (1)

- **CLASS**
  - Represent a kind of person, place, or thing about which the system will need to capture and store information
  - Has a name typed in bold and centered in its top compartment
  - Has a list of attributes in its middle compartment
  - Has a list of operations in its bottom compartment



CLASS DIAGRAM SYNTAX (2)

- **ATTRIBUTE**
  - Represents properties that describe the state of an object
  - Can be derived from other attributes, show by placing a slash before the attribute's name
  - The visibility of an attribute can be public (+), protect (#), or private (-).

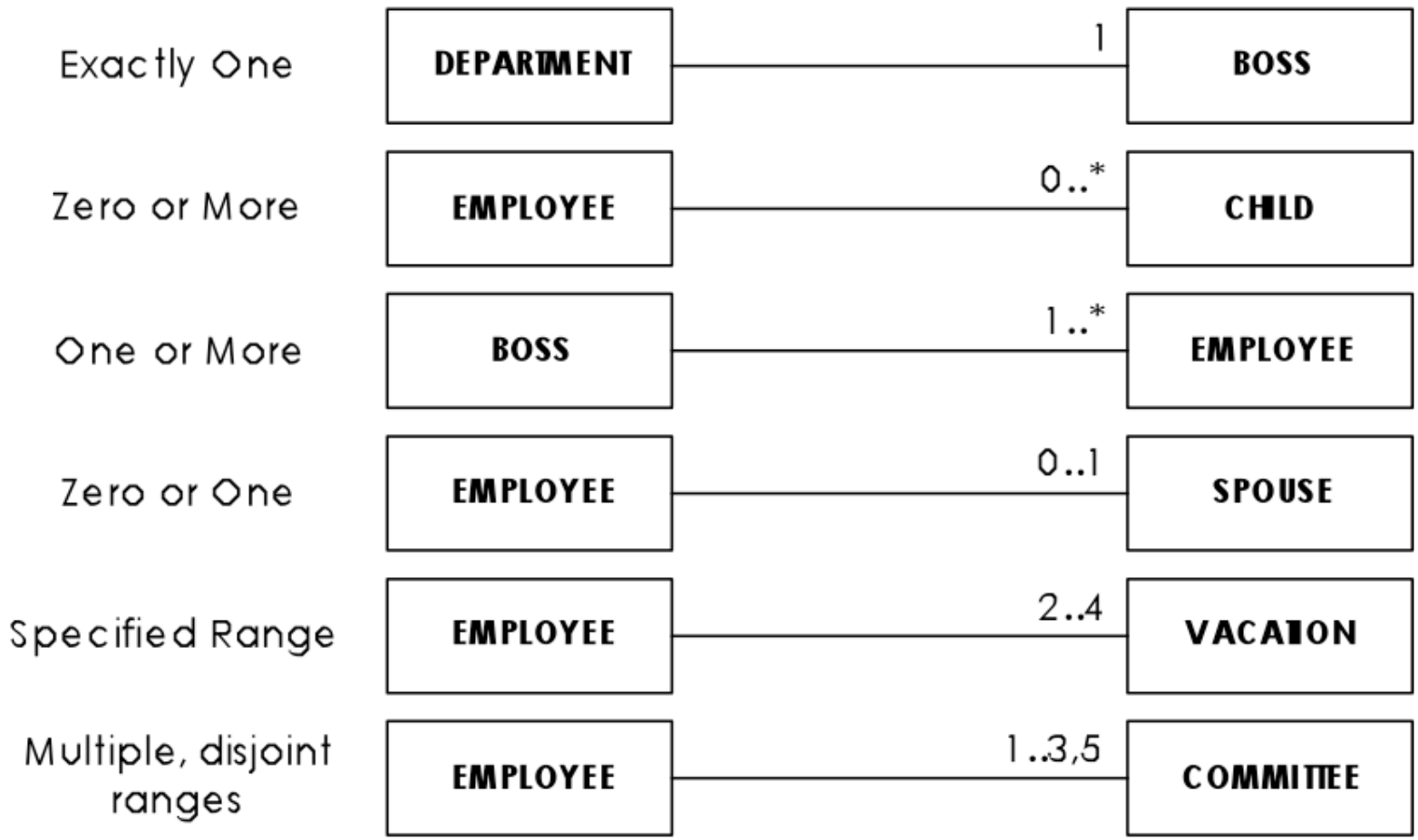
CLASS DIAGRAM SYNTAX (3)

- **OPERATION**
  - Represents the actions or functions that a class can perform.
  - Includes parentheses that may contain parameters or information needed to perform the operation.

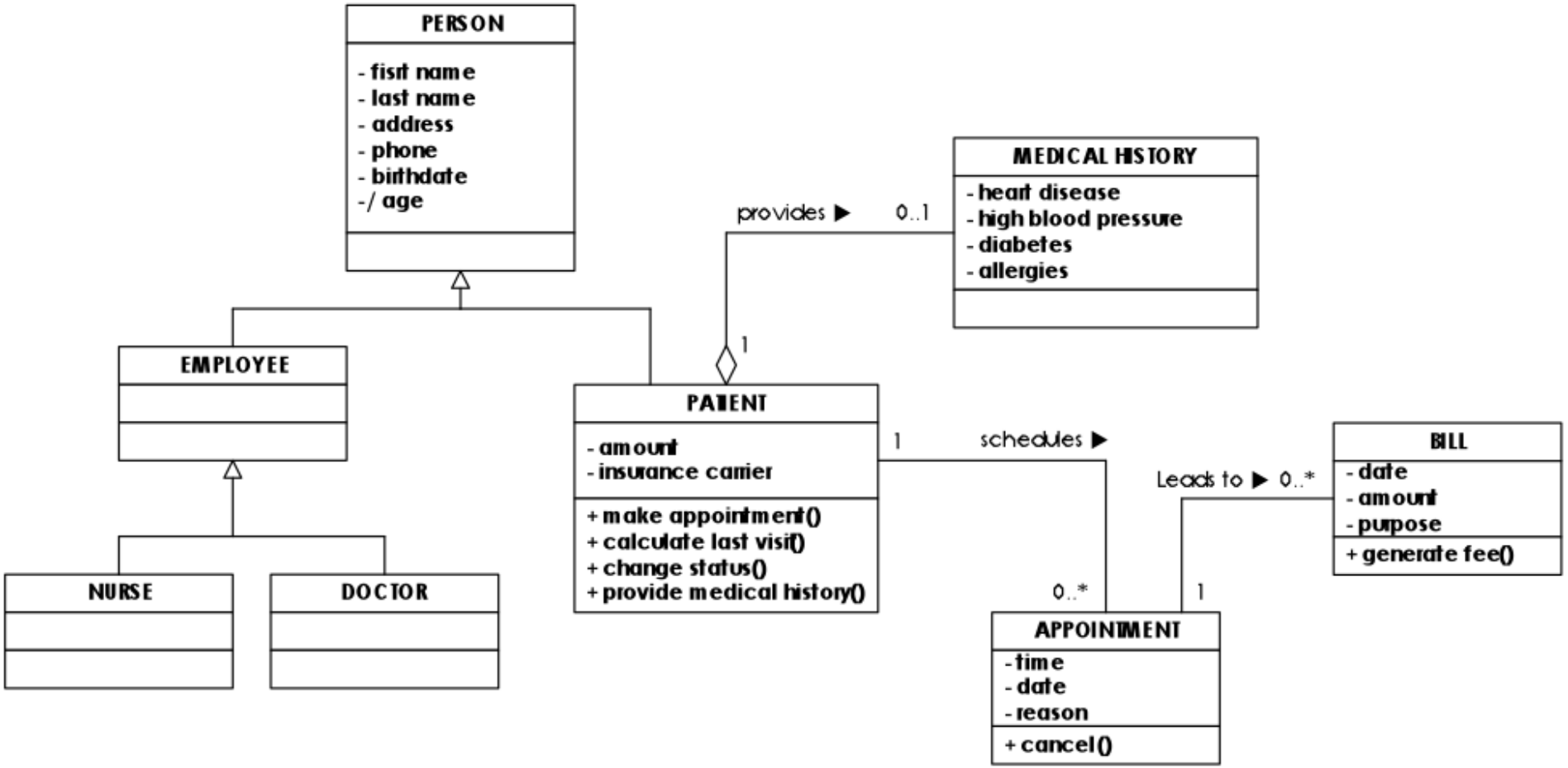
CLASS DIAGRAM SYNTAX (4)

- **ASSOCIATION**
  - Represents a relationship between multiple classes, or a class and itself
  - Is labeled using a verb phrase or a role name, whichever better represents the relationship
  - Can exist between one and more classes
  - Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance

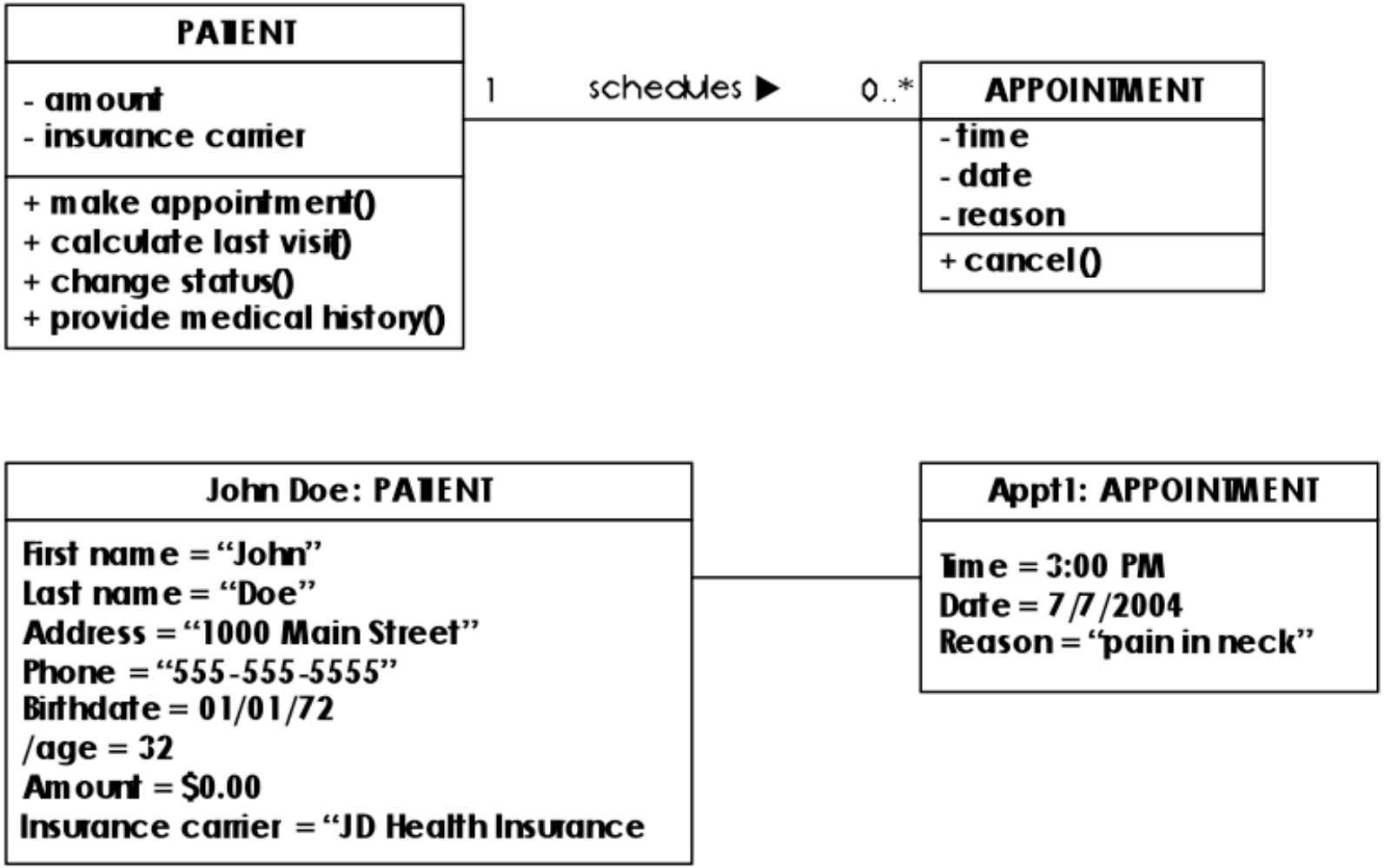
MULTIPLICITY



EXAMPLE 4: CLASS DIAGRAMS



OBJECT DIAGRAMS



WORKSHOP (3)

- Each group create a Class Diagram of the “Room Reservation System” using draw.io

BEHAVIORAL MODELS

- Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.

BEHAVIORAL STATE MACHINES

- A behavioral state machine is a dynamic model that shows the different states that a single object passes through during its life in response to events, along with its responses and actions.

SEQUENCE DIAGRAM SYNTAX (1)

- **STATE**
  - Is show as a rectangle with rounded corners
  - Has a name that represents the state of an object
- **INITIAL STATE**
  - Is show as a small filled-in-circle
  - Represents the point at which an object begins to exist

SEQUENCE DIAGRAM SYNTAX (2)

- **FINAL STATE**
  - Is show as a circle surrounding a small solid filled circle
  - Represents the completion of activity

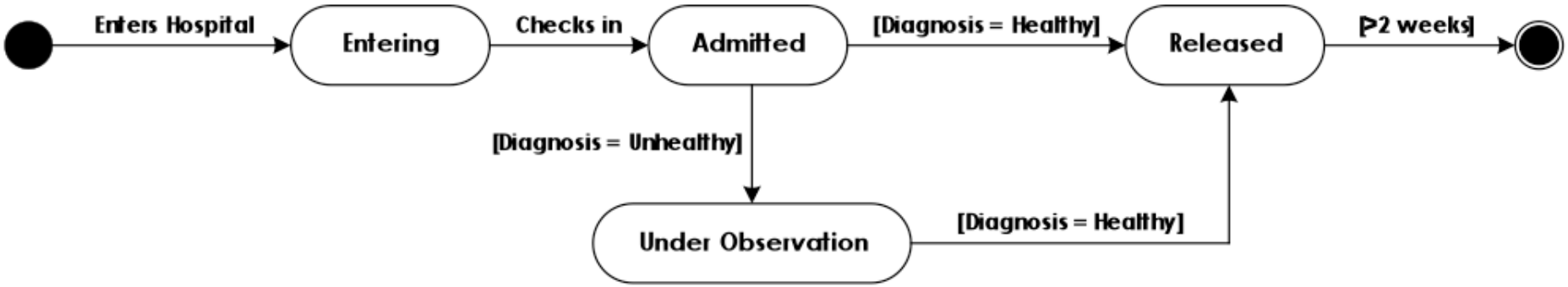
- **AN EVENT**
  - Is a noteworthy occurrence that triggers a change in state
  - Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time
  - Is used to label a transaction

SEQUENCE DIAGRAM SYNTAX (3)

- **TRANSACTION**

  - Indicates that an object in the first state will enter the second state
  - Is triggered by the occurrence of the event labeling the transaction
  - Is show as a solid arrow from one state to another, labeled by the event name

EXAMPLE 5: STATE DIAGRAM



WORKSHOP (4)

- Each group create a State Diagram for the **Room** in the “Room Reservation System” using draw.io.

SEQUENCE DIAGRAMS

- Sequence diagrams is a behavioral model to describe the internal dynamic aspects of an information system that supports the business processes in an organization
- Sequence diagrams are one of two types of interaction diagrams.
- They illustrate the objects that participate in a use case and the messages that pass between them over time for one use case.

SEQUENCE DIAGRAM SYNTAX (1)

- **ACTOR**
  - Is a person or system that derives benefit from and is external to the system
  - Participates in a sequence by sending and/or receiving messages
  - Are placed across the top of the diagram
  - Is depicted as either a stick figure (default) or if a non-human actor is involved, as a rectangle with <<actor>> in it (alternative)

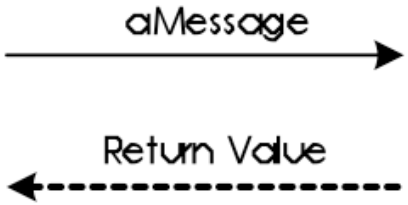
anActor

SEQUENCE DIAGRAM SYNTAX (2)

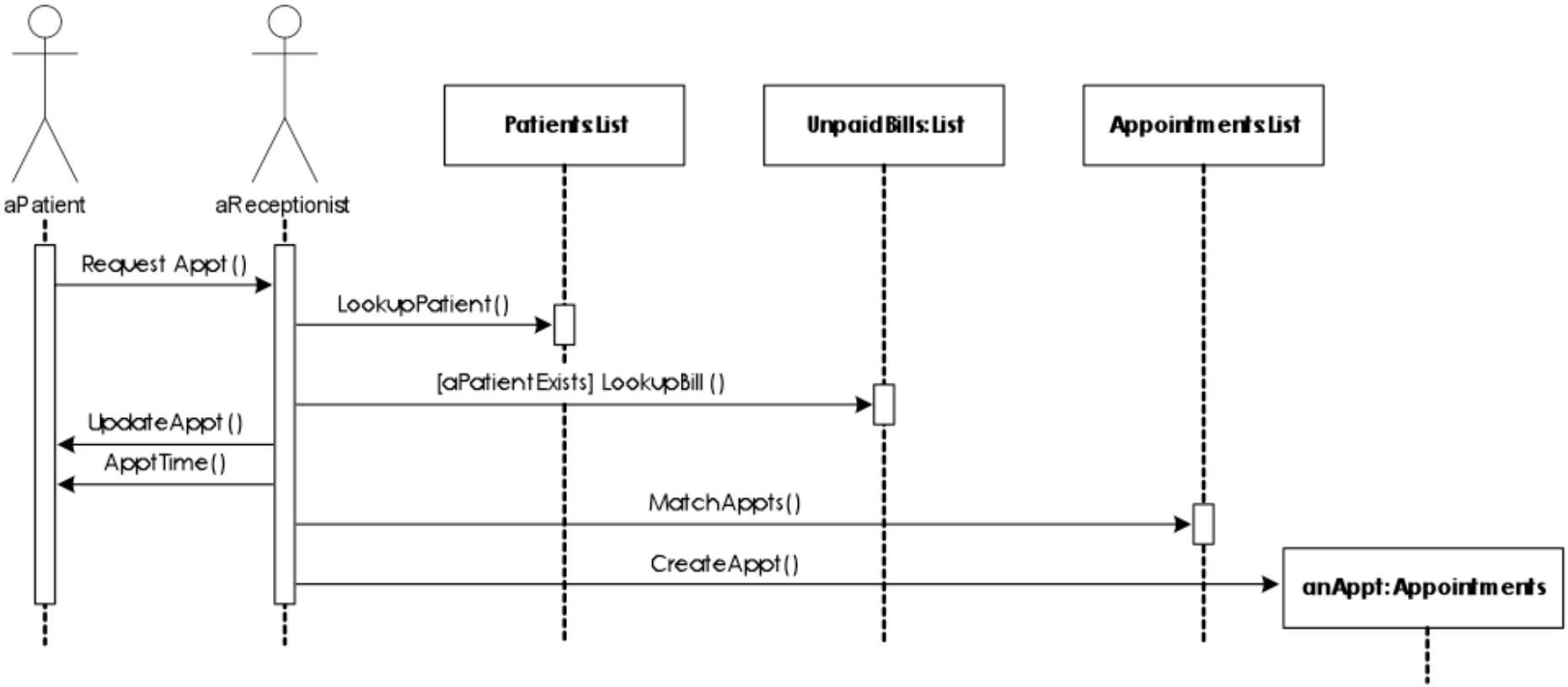
- **OBJECT**
  - Participates in a sequence by sending and/or receiving messages
  - Are placed across the top of the diagram
- **LIFELINE**
  - Denotes the life of an object during a sequence

SEQUENCE DIAGRAM SYNTAX (3)

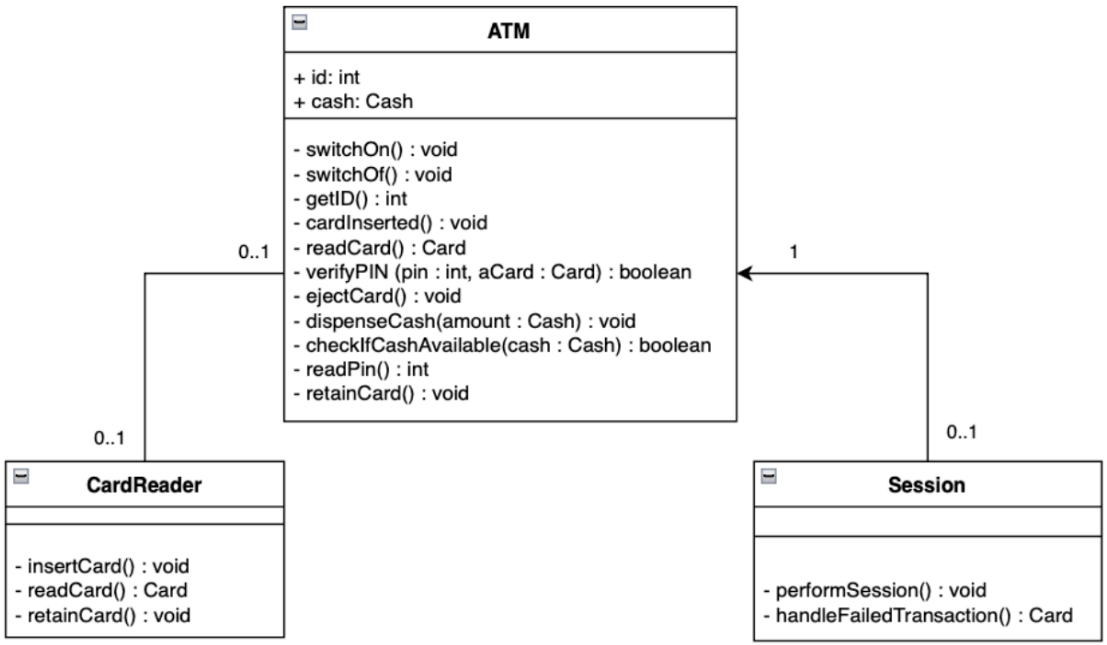
- **EXECUTION OCCURRENCE**
  - Is a long narrow rectangle placed atop a lifeline
  - Denotes when an object is sending or receiving message
- **MESSAGE**
  - Convey information from one object to another one
  - An operation call is labeled with the message being sent and a solid arrow, which a return is labeled with the value being returned and show as a dash arrow

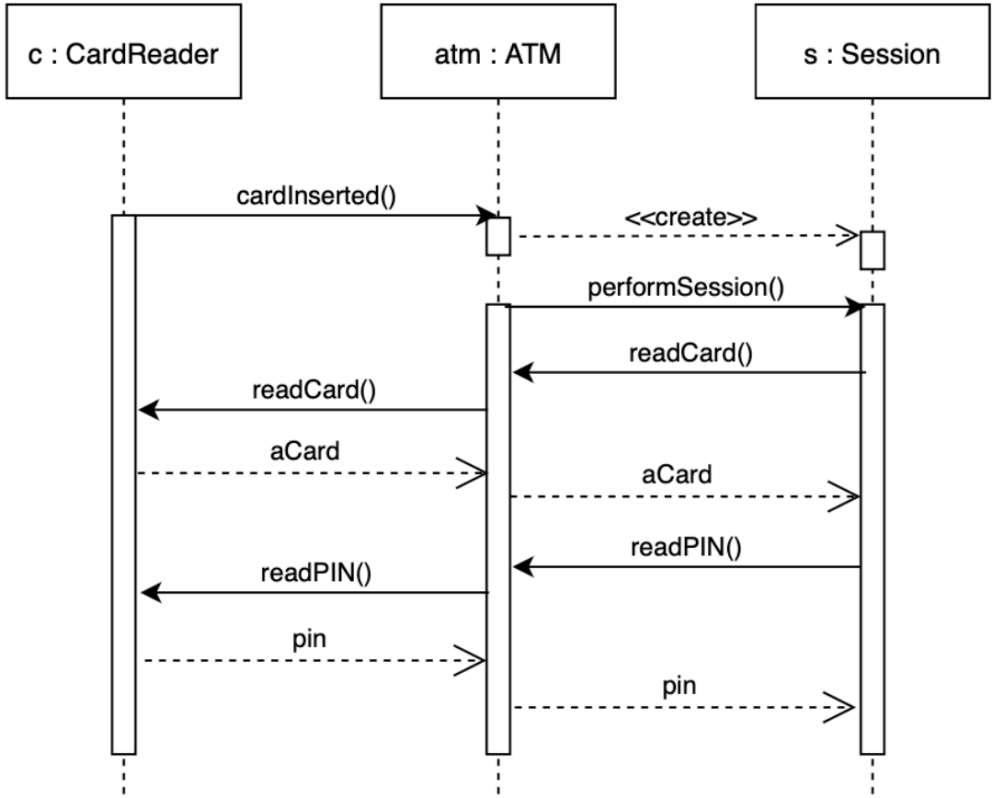


EXAMPLE 6: SEQUENCE DIAGRAM



RELATIONSHIP BETWEEN CLASS AND SEQUENCE DIAGRAM

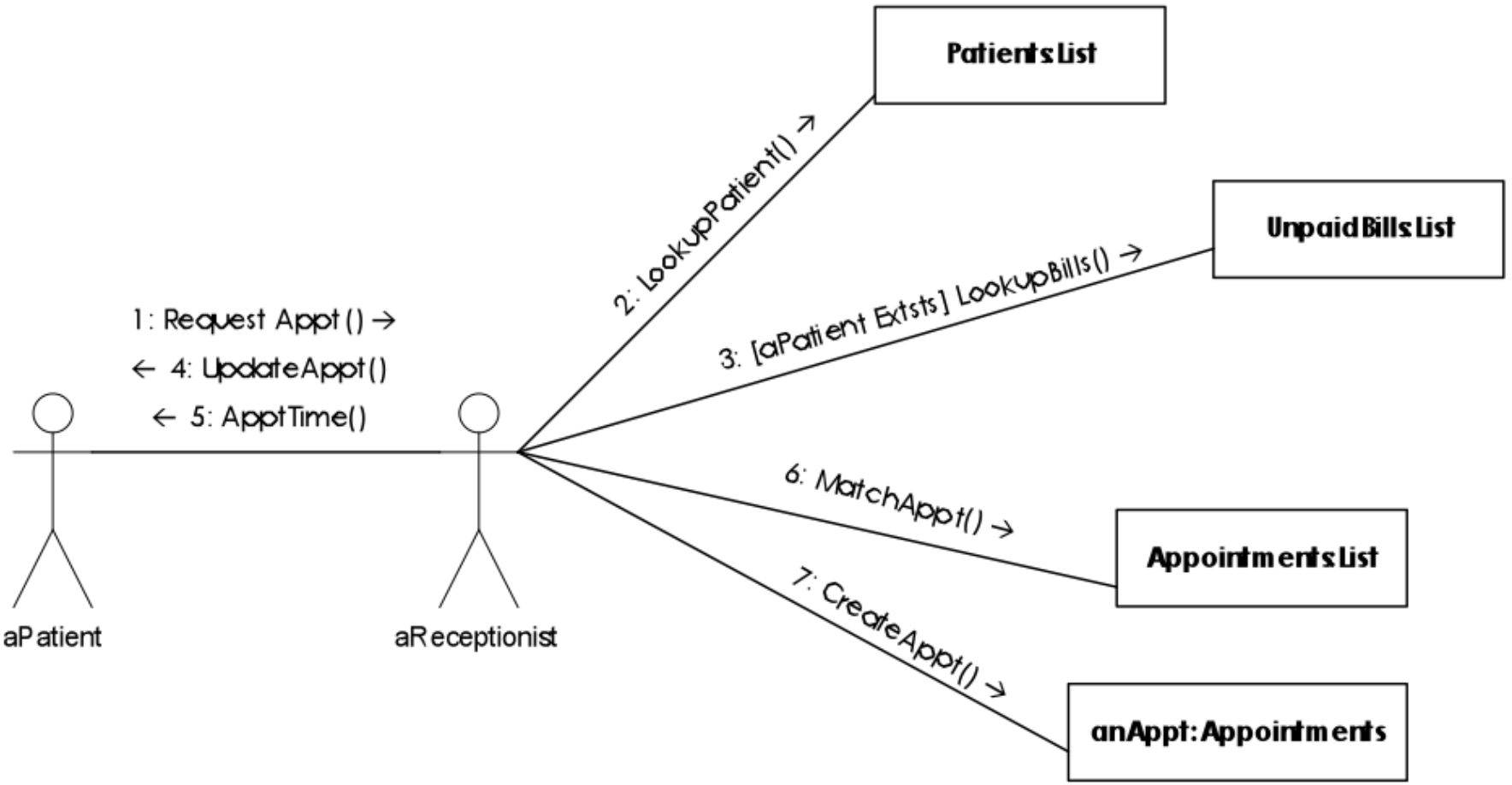




STEPS FOR BUILDING SEQUENCE DIAGRAMS

- Set the context.
- Identify which objects will participate.
- Set the lifeline for each object.
- Layout the message from the top to the bottom of the diagram based on the order in which they are sent.
- Add the execution occurrence to each object's lifeline.
- Validate the sequence diagram.

COMMUNICATION DIAGRAMS



ASSIGNMENT 4

- Use UML to analyze the requirements of a room reservation system.
  - Use-Case Diagram*
  - Activity Diagram*
  - Class Diagram*
  - State Diagram (Reservation State)
  - Sequence Diagram