Object Oriented Design

Lecture 4

Ruzanna Chitchyan, Jon Bird, Pete Bennett
TAs: Alex Elwood, Alex Cockrean, Casper Wang

Overview

- Why would we do OO design?
- Class Diagrams
  - Associations: Composition and Aggregation
  - Generalisation: Inheritance
  - Navigability
- Modelling behaviour
  - Communication diagrams
  - Sequence Diagrams

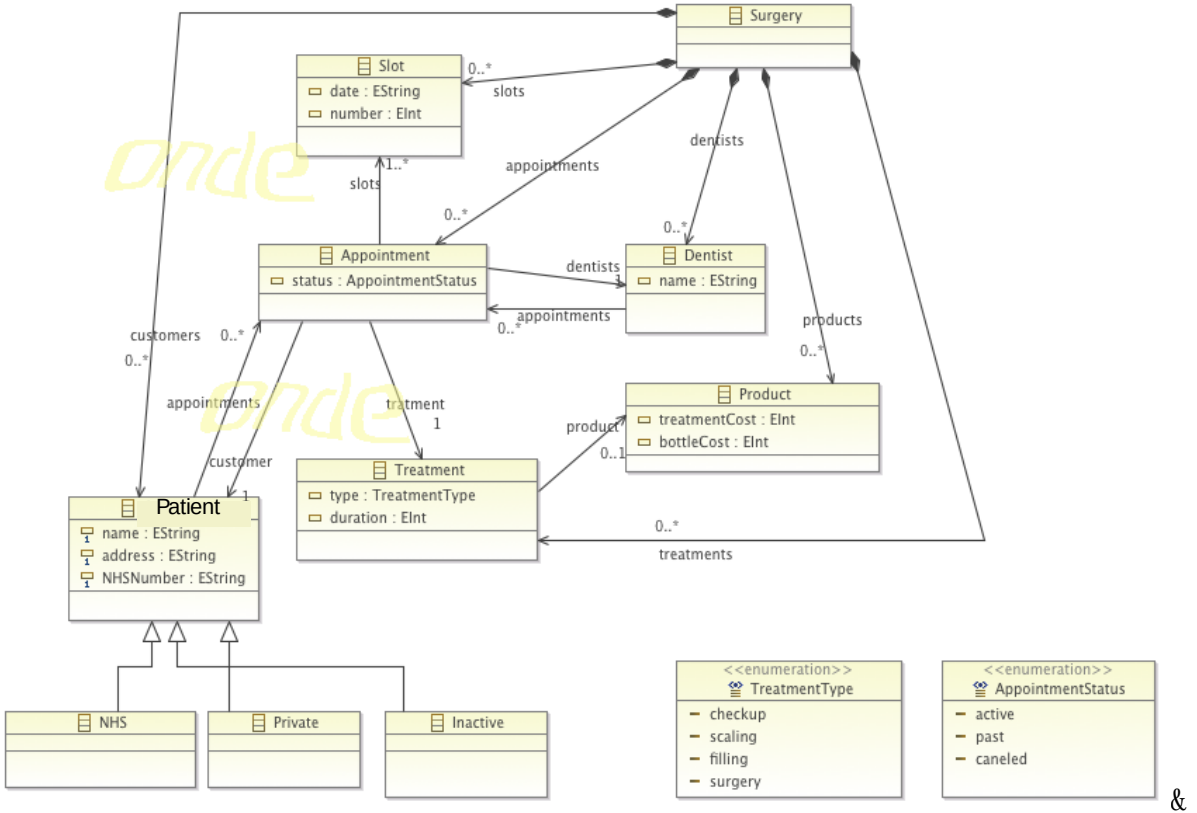
Why OO Design?

- Organise ideas
- Plan work
- Build understanding of the system structure and behavior
- Communicate with development team
- Help (future) maintenance team to understand

Class Diagrams

What Is a Class Diagram?

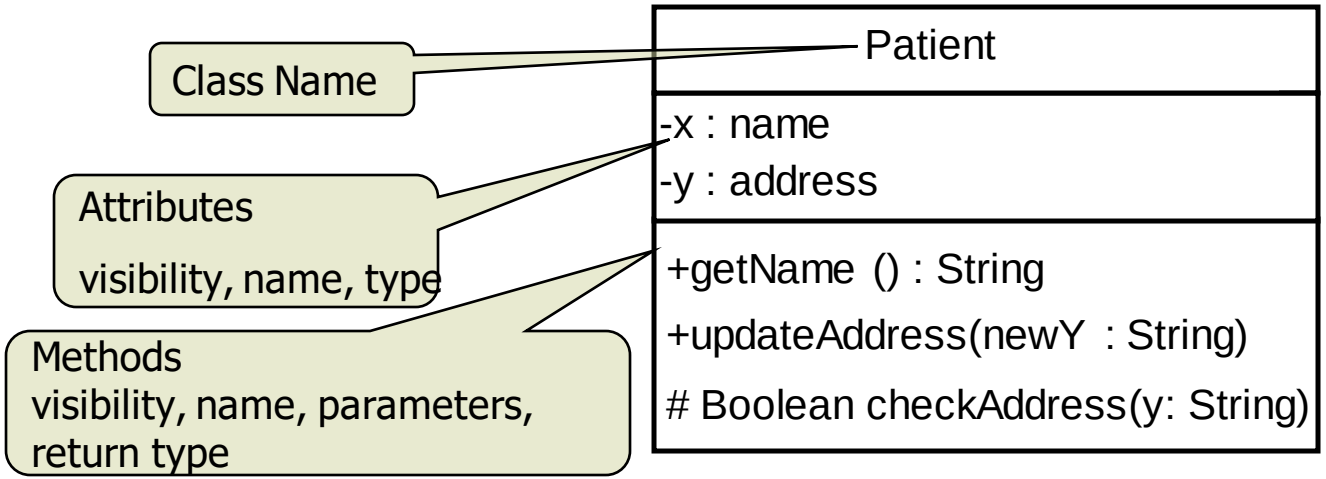
- Static view of a system



&

Class Diagrams

Class can be understood as a template for creating objects with own functionality



Visiblity:

+ public # protected - private

Notation for Attributes

[visibility] name [: type] [multiplicity] [=value] [{property}]

- visibility
  - other package classes
  - - private : available only within the class
  - + public: available for the world
  - # protected: available for subclasses and other package classes
  - ~ package: available only within the package
- [multiplicity], by default 1
- properties: readOnly, union, subsets<property-name>, redefined<property-name>, ordered, bag, seq, composite
- static attributes appear underlined

Notation for Operations

[visibility] name ([parameter-list]) : [{property}]

- visibility
- method name
- formal parameter list, separated by commas:
  - direction name : type [multiplicity] = value [{property}]
  - static operations are underlined
- Examples:
  - display()
  - - hide()
  - + toString() : String
  - createWindow (location: Coordinates, container: Container): Window

How do we find Classes: Grammatical Parse

Classes

- Identify nouns from existing text
- Narrow down to remove
    - Duplicates and variations (e.g., synonyms)
    - Irrelevant
    - Out of scope

Grammatical Parse: Dental Surgery Example

You are responsible for development of a software system for keeping track of the appointments and services of a dental surgery. This business employs several dentists, provides treatments to NHS and non NHS patients, and allows for patients to buy products (e.g., toothbrush, paste, etc.) when they pay for the received services (such as periodontal therapy with plague removal and scaling, and dental surgery).

Grammatical Parse: Dental Surgery Example

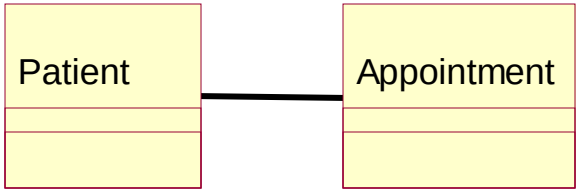
You are responsible for development of a software system for keeping track of the appointments and services of a dental surgery. This business employs several dentists, provides treatments to NHS and non NHS patients, and allows for patients to buy products (e.g., toothbrush, paste, etc.) when they pay for the received services (such as periodontal therapy with plague removal and scaling, and dental surgery).

Structural Relationships in Class Diagrams

What Is an Association?

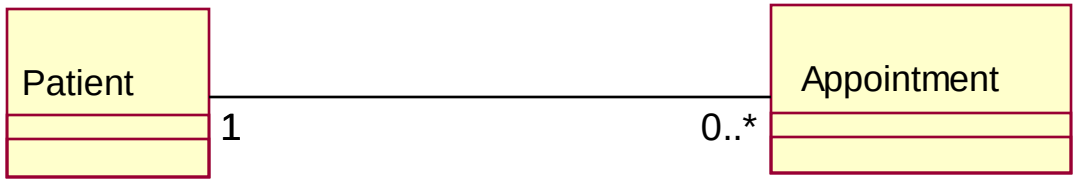
- The semantic relationship between two or more classifiers that specifies connections among their instances.

- A structural relationship specifying that objects of one thing are connected to objects of another thing.



What Is Multiplicity?

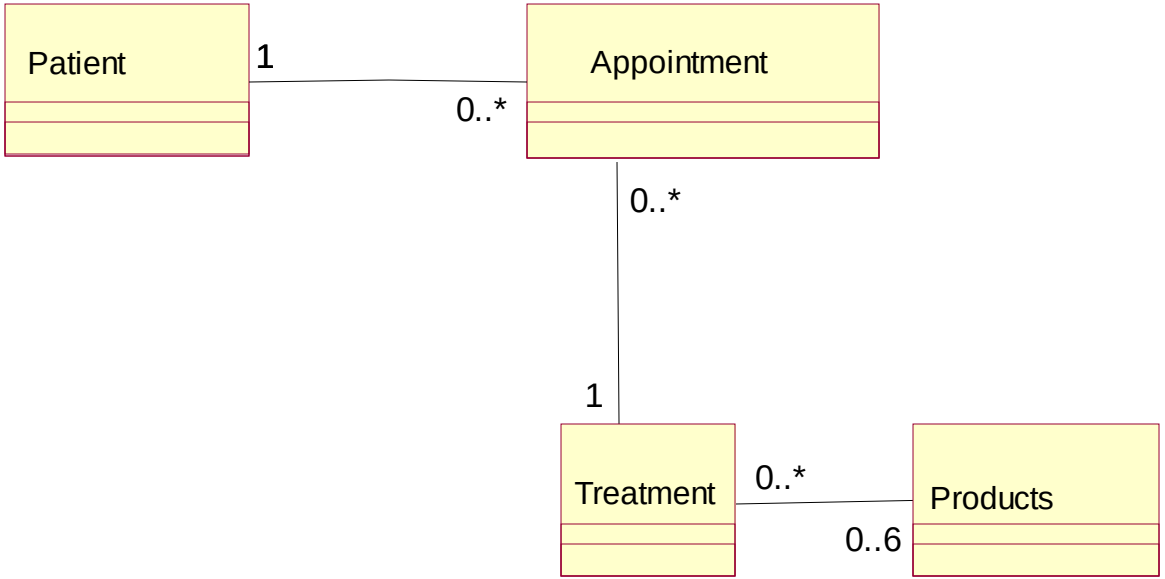
- Multiplicity is the number of instances one class relates to ONE instance of another class.
- For each association, there are two multiplicity decisions to make, one for each end of the association.
  - For each instance of Patient, many or no Appointments can be made.
  - For each instance of Appointment, there will be one Patient to see.



Multiplicity Indicators

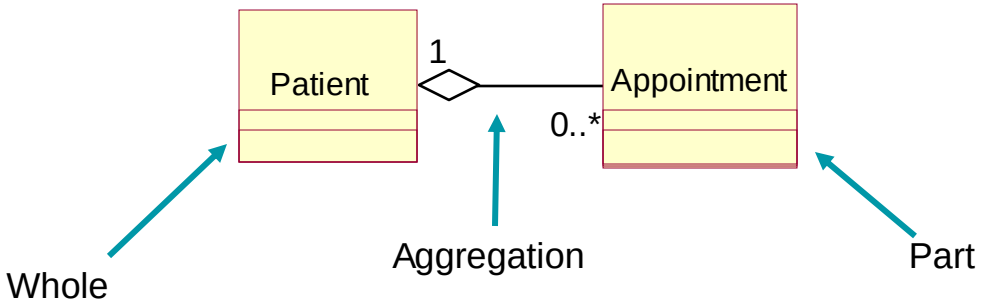
Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

Example: Multiplicity



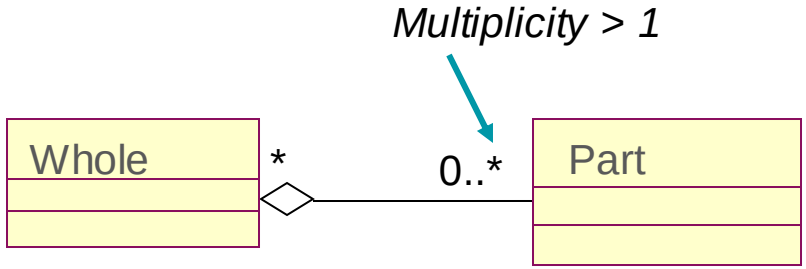
What Is an Aggregation?

- A special form of association that models a whole-part relationship between the aggregate (the whole) and its parts.
  - An aggregation is an “is a part-of” relationship.
- Multiplicity is represented like other associations.

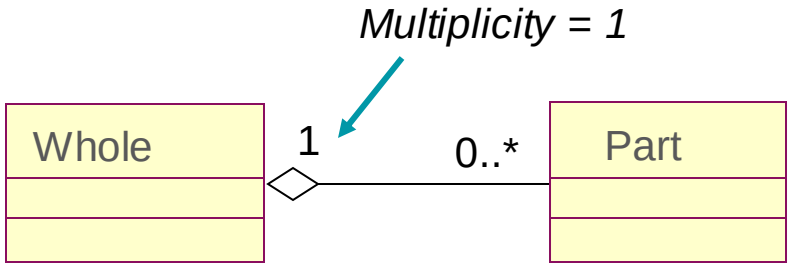


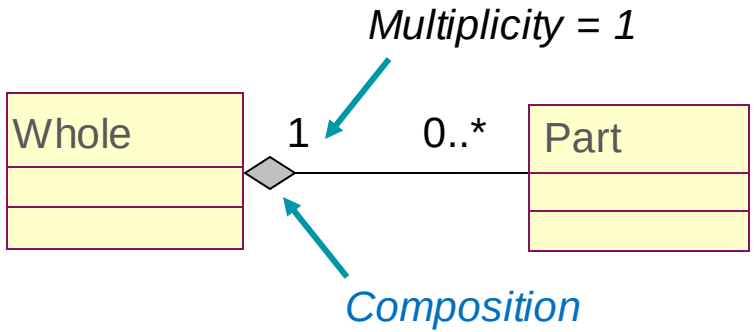
Aggregation: Shared vs. Non-shared

- Shared Aggregation



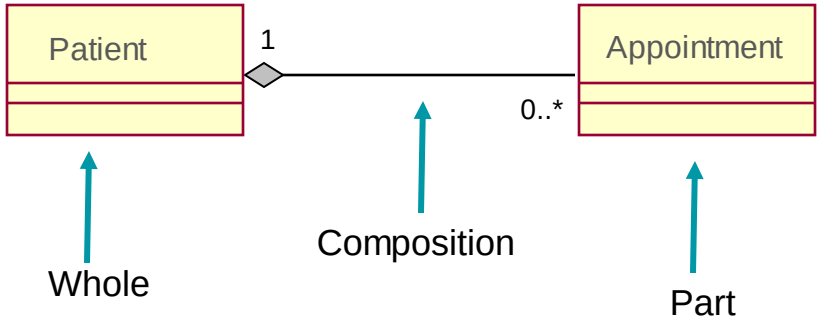
- Non-shared Aggregation





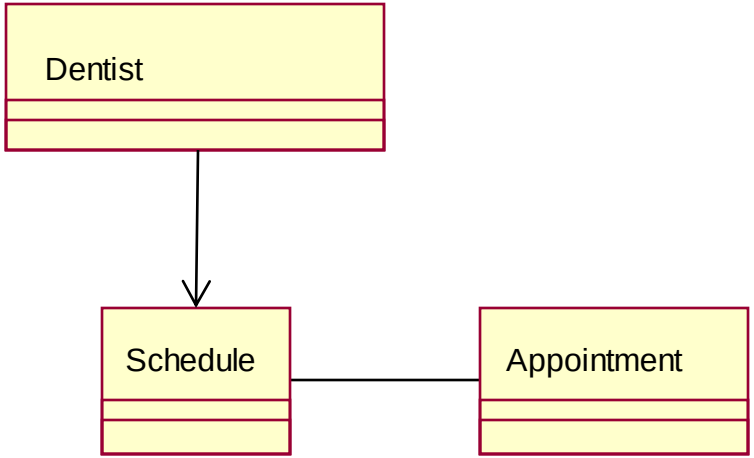
What Is Composition?

- A form of aggregation with strong ownership and coincident lifetimes
  - The parts cannot survive the whole/aggregate



What Is Navigability?

- Indicates that it is possible to navigate from an associating class to the target class using the association



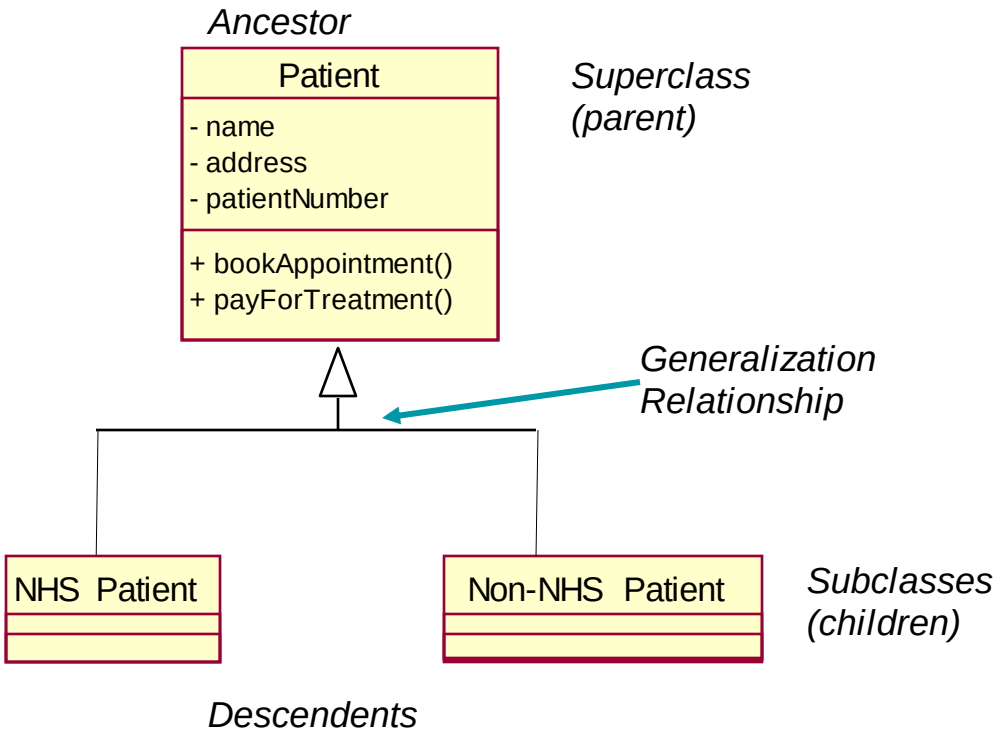
What Is Generalization?

- A relationship among classes where one class shares the *properties and/or behavior* of one or more classes.
- Defines a hierarchy of abstractions where a subclass inherits from one or more superclasses.

- Is an "**is a kind of**" relationship.

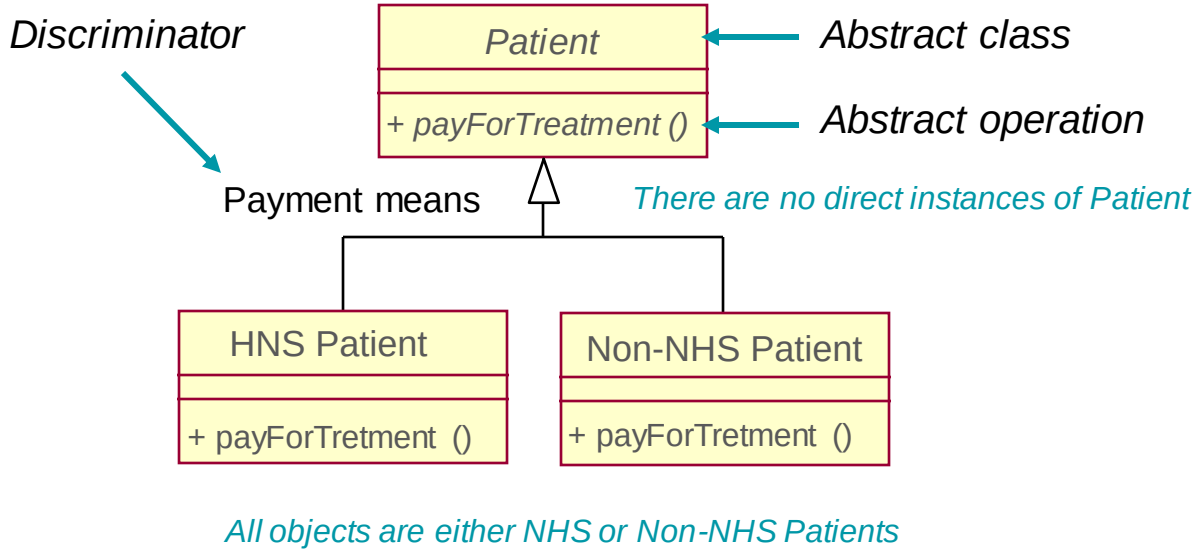
Example: Inheritance

- One class inherits from another
- Follows the “is a” style of programming
- Class substitutability



Abstract and Concrete Classes

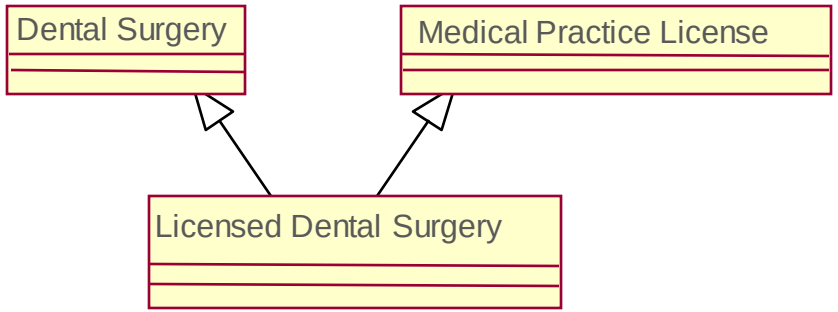
- Abstract classes cannot have any objects
- Concrete classes can have objects



Generalization vs. Aggregation

- Generalization and aggregation are often confused
  - Generalization represents an “is a” or “kind-of” relationship
  - Aggregation represents a “part-of” relationship

Is this correct?



Behaviour Modelling

Objects Need to Collaborate

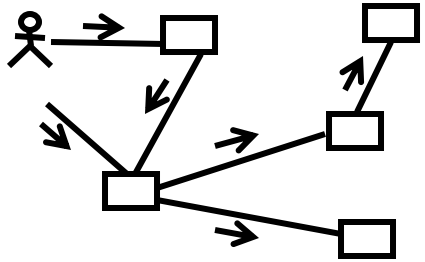
- Objects are useless unless they can collaborate to solve a problem.
  - Each object is responsible for its own behavior and status.
  - No one object can carry out every responsibility on its own.

- How do objects interact with each other?
  - They interact through messages.
  - Message shows how one object asks another object to perform some activity.

Communication Diagrams

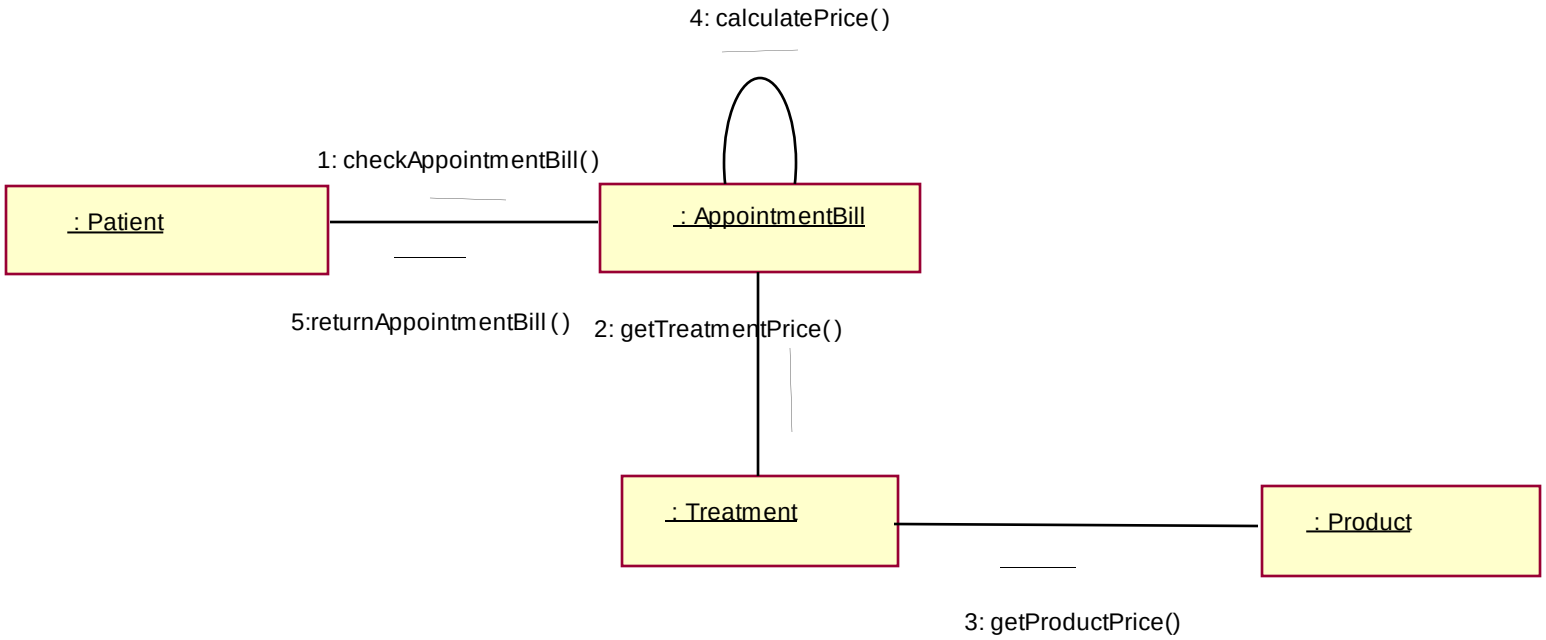
What Is a Communication Diagram?

- A communication diagram emphasizes the organisation of the objects that participate in an interaction.
- The communication diagram shows:
  - The objects participating in the interaction.
  - Links between the objects.
  - Messages passed between the objects.



Communication Diagrams

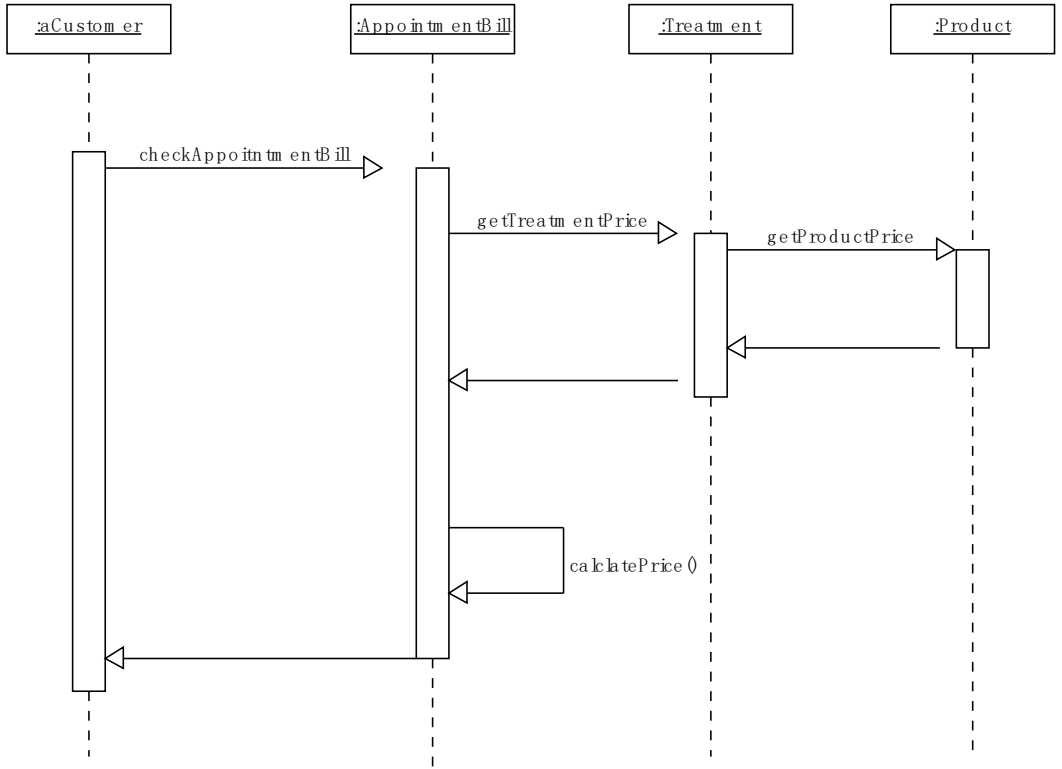
Example: Communication Diagram



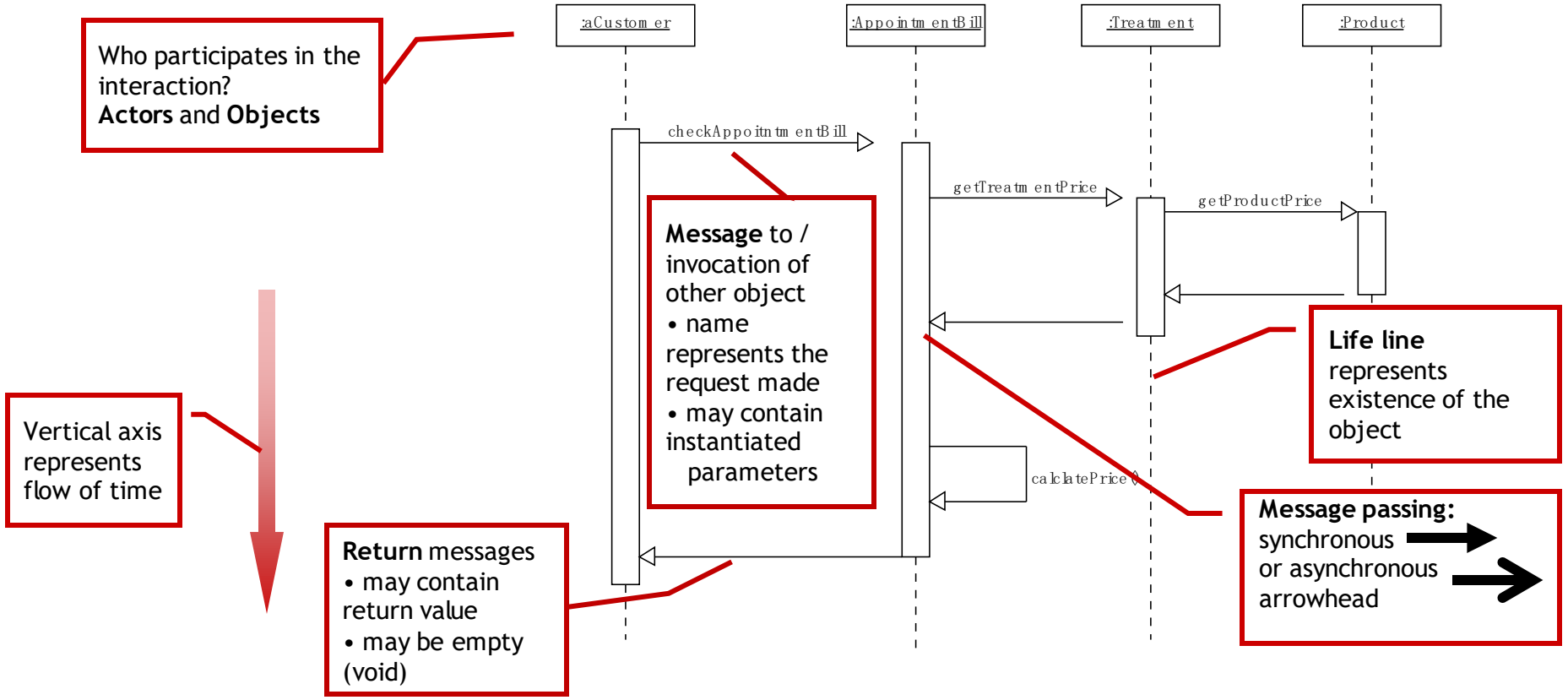
Sequence Diagrams

Sequence Diagrams: Basic Elements

- A set of participants arranged in time sequence
- Good for real-time specifications and complex scenarios



Sequence Diagrams: Basic Elements



Method for Analysis Sequence Diagrams

- for each scenario (high-level sequence diagram)
  - decompose to show what happens to objects inside the system

    ➔ objects and messages
  - Which tasks (operation) does the object perform?

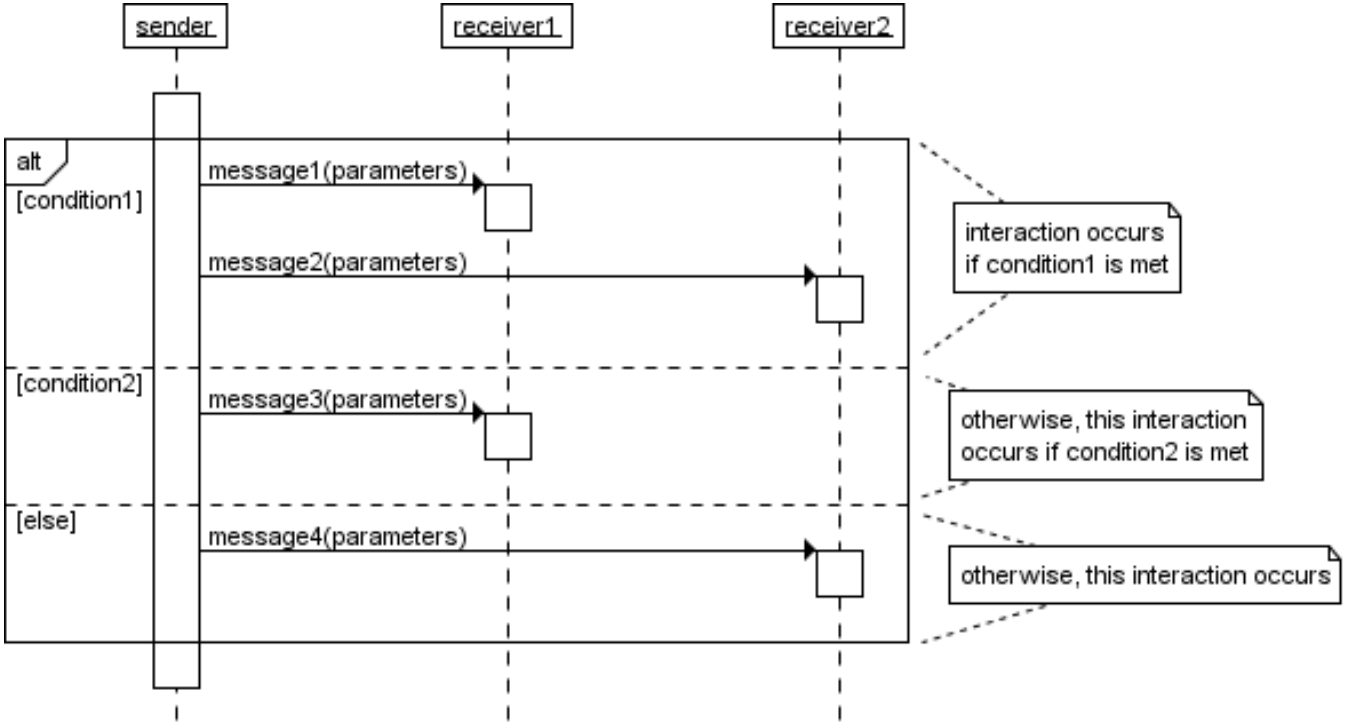
    ➔ label of message arrow
  - Who is to trigger the next step?

    ➔ return message or pass on control flow

Sequence Diagrams

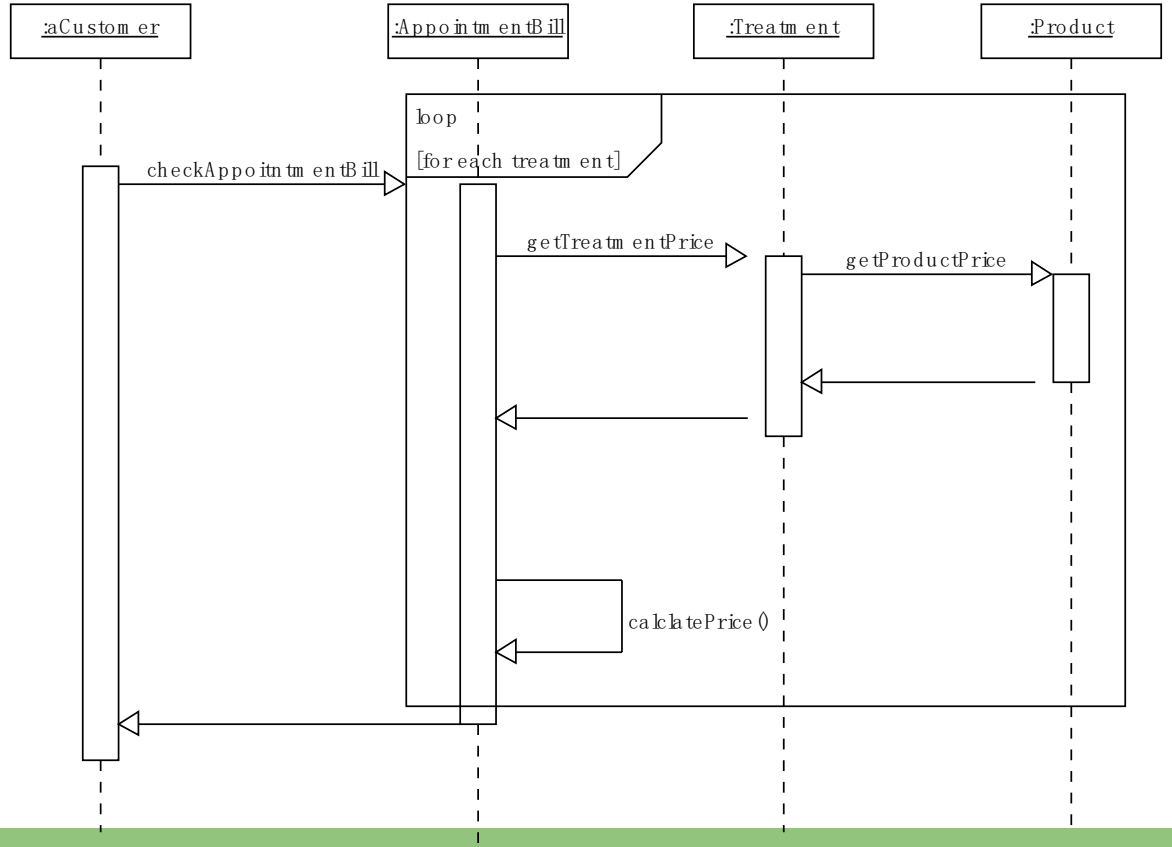
- Sequence Diagrams can model simple sequential flow, branching, iteration, recursion and concurrency
- They may specify different scenarios/runs
    - Primary
    - Variant
    - Exceptions

Interaction frames: alt



https://web.archive.org/web/20231018070441/http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

Interaction frames: loop



Comparison: Communication and Sequence Diagrams

Sequence and Communication Diagram Similarities

- Semantically equivalent
    - Can convert one diagram to the other without losing any information

- Model the dynamic aspects of a system
- Model a use-case scenario

Sequence and Communication Diagram Differences

Sequence diagrams	Communication diagrams
▪ Show the explicit sequence of messages ▪ Show execution occurrence ▪ Better for visualizing overall flow ▪ Better for real-time specifications and for complex scenarios	▪ Show relationships in addition to interactions ▪ Better for visualizing patterns of communication ▪ Better for visualizing all of the effects on a given object ▪ Easier to use for brainstorming sessions

Review

- What does a class diagram represent?
- Define association, aggregation, and generalization.
- How do you find associations?
- What information does multiplicity provide?

- What is the main purpose of a SD?
- What are the main concepts in a SD?
- What are the communication diagrams?
- What is the difference between SD and communication diagrams?