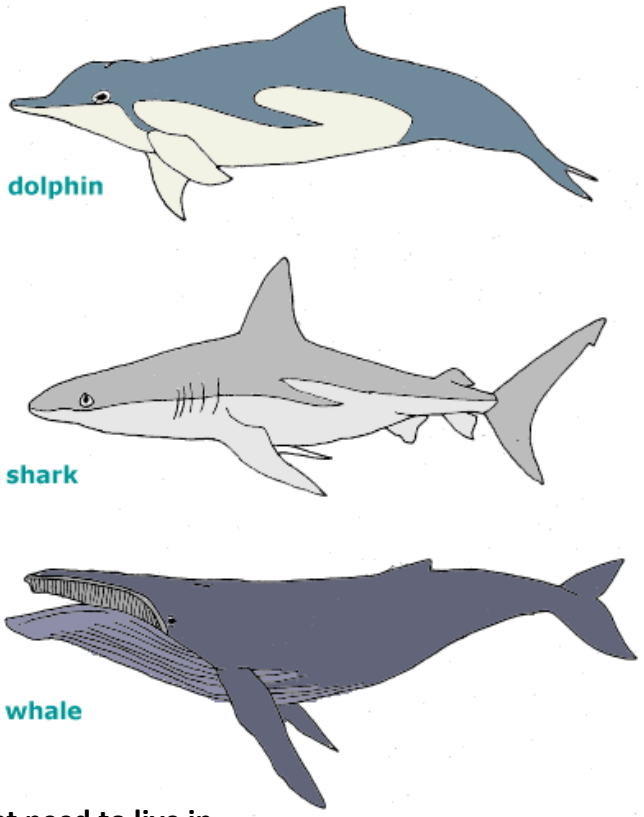
UML: Object-Oriented Approach and Class Diagram

Dr Alireza Nili

IFB103

Queensland University of Technology



Classes of animals that need to live in water/oceans [example]

Accessed on 06/May/2022 from https://www.pinterest.com.au/pin/446911962991827989/

QUT Science and Engineering

Housekeeping

We introduced UML and worked on a UML Use Case Diagram last week.

This week's lecture and tutorial focus on the object-oriented approach and UML Class Diagram.

Next week, we learn UML Activity Diagram and UML Sequence Diagram. Unlike the Use Case and Class diagrams which focus on features of a system, Activity and Sequence diagrams focus on the process of using the system.

QUT Science and Engineering

Week 10 assessment structure

Before your week 10 online session:

1. Ensure you have watched Week 10 lecture recording.
2. Have another look at DC2 assessment specification (the final assessment, available on Blackboard).

During week 10:

1. Read the Week 10 tutorial slides. Think on items 4a and 4b in the assessment specification file. Draw the first draft of your class diagram (item 4a) and answer item 4b **[Assessed items]**
2. Show the first draft of your diagram to your tutor during your tutorial. You need to use the screenshare feature if you are attending an online tutorial. Your tutor will give you feedback on your work.

This is important, so I repeat:

Every week:

1. Read/review DC2 assessment specification, particularly the parts of the document that are relevant to this week's activities.
2. Watch the lecture recording.
3. Read this set of slides and other teaching resources each week before attending your tutorial.
4. You are expected to draw the first draft of your diagram before attending your tutorial and show it to your tutor during the session, so your tutor can give you more specific feedback.
5. Your tutor will answer your questions and provides feedback for improving your diagram.

Please note that the above schedule gives you an extended time for reading the teaching material and thinking on the tutorial activity and an opportunity for a personalised feedback from your tutor.

Goals for today

Object-Oriented Approach

Class Diagram

Multiplicity in Class Diagrams

Object-Oriented Approach

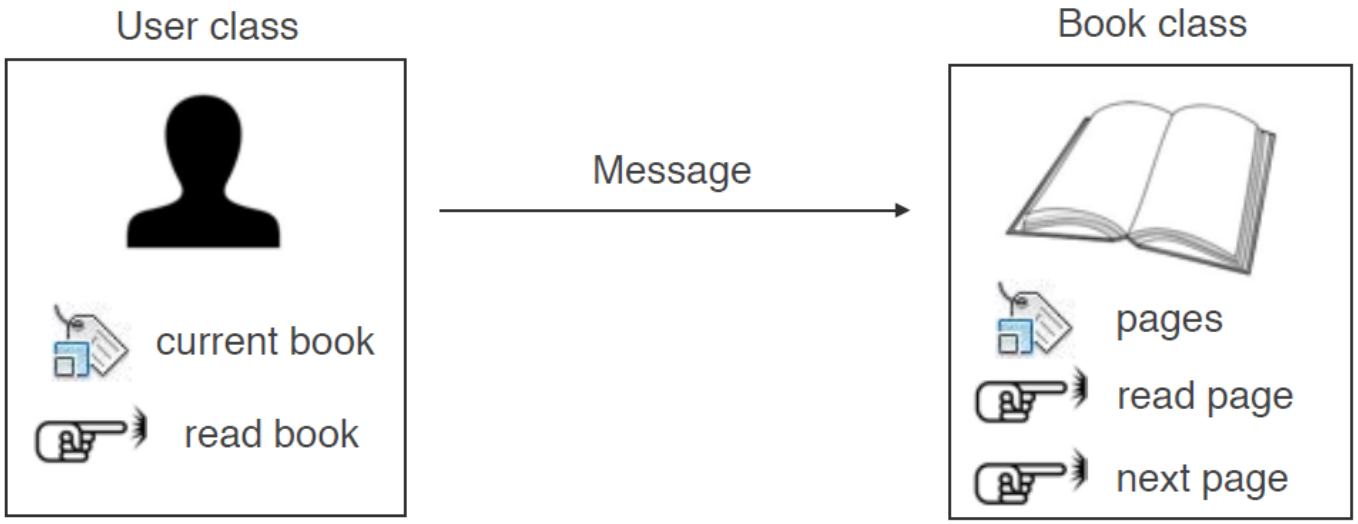
Class Diagram

Multiplicity in Class Diagrams

Wrap up

Reading a book

– You can think about reading a book as communication between two objects, the user and the book



"User reads a page from a book" is translated into "sends a message to the book asking for the current page contents"

Object-Oriented (OO) Approach

- Building models that are focused on "things", rather than "process" or "ordering of events", in the problem space
- Viewing the world as a meaningful collection of objects that collaborate to achieve some higher level behaviour
- More resilient to change and more maintainable
- Basic building blocks: objects and classes
- We 'usually' draw only one class diagram for a system. A class diagram will eventually be translated into programming code.

What is an object

- "An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain."

- Nearly anything can be an object
  - External entities (e.g. people, devices, other systems)
  - Things (e.g. sensors, computers, reports, displays, signals)
  - Roles (e.g. employer, employee, student)
  - Organisational unit (e.g. division, group, team)
  - Places (e.g. university, manufacturing workshop)
  - …

- But, some things cannot be objects
  - Attributes/properties (e.g. green, large)
  - Emotion (e.g. happy, angry)

What is a class

- A class represents a group of objects with
  - common structure (attributes/properties),
  - common behaviour (methods/operations),
  - common relationships to other objects, and
  - common meaning (“semantics”)

- An object is an *instance* of a class
  - An object has state, exhibits some well-defined behaviour, and has a unique identity.

- UML is often used to define a class (and any instance of a class, i.e. object)
  - UML Class Diagram

What is a class

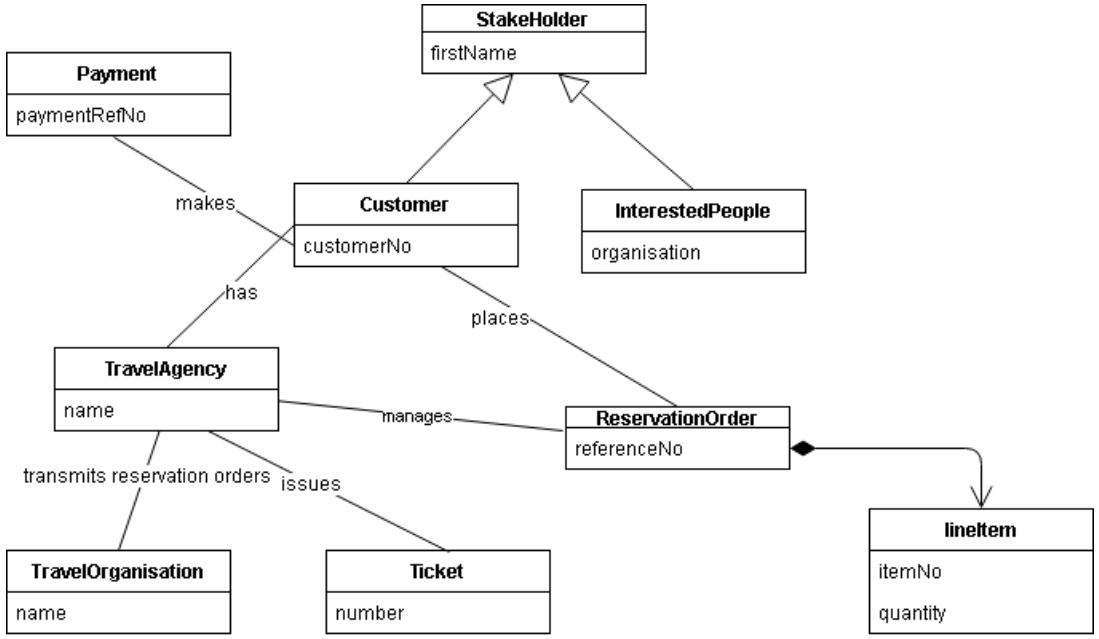
- Examples:
  - ***Student*** (with student identifier, student name, etc)
  - ***Unit*** (unit code, unit name)
  - ***Unit offering*** (unit offered in semester I/II of a year)
  - ***Lecture*** (lecture sequence identifier, duration, topic. location)
  - ***Lecturer*** (Staff identifier, staff name)
  - ***Workshop*** (workshop sequence number, duration, topic, location)
  - ***Tutor*** (Staff identifier, staff name)

Behaviour of an object

Behaviour is how an object acts and reacts, in terms of its state changes and message passing. State changes upon invoking certain ‘operations’ to an object

Modelling complex systems Using Object-oriented (OO) Approach

- View the world as a set of objects
- Each object has its own attributes and *behaviour*
- Operations are associated with objects



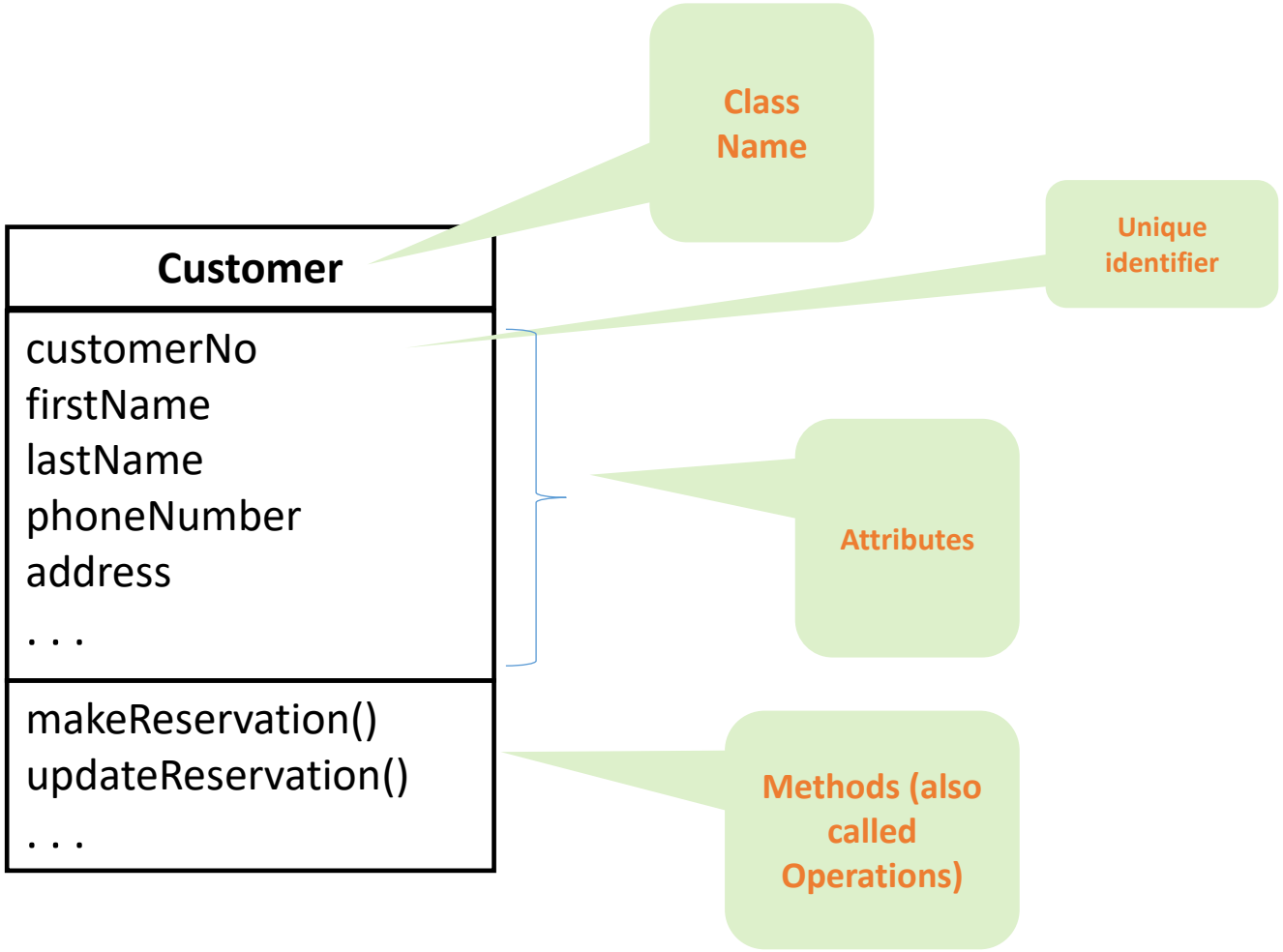
Object-Oriented Approach

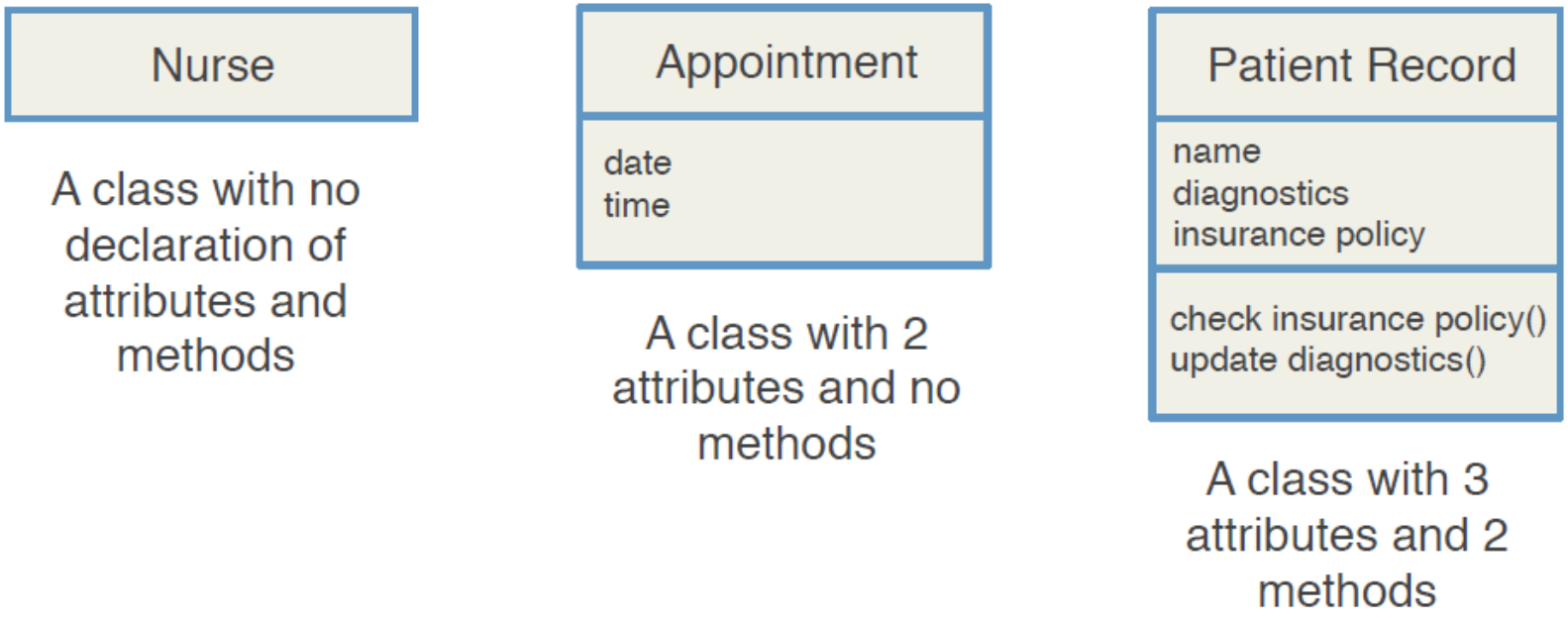
Class Diagram

Multiplicity in Class Diagrams

Wrap up

UML Class Diagram: Representing a class





A class that has no attributes or methods is suspect.

- May not be relevant

Example: “Customer” Objects

Customer

customerNo: B1001
firstName: James
lastName: Hogg
phone: 0739312005
address: 1 Oxford st, Redhill, QLD
. . .

Customer

customerNo: B1002
firstName: Lauren
lastName: Louise
phone: 0739313433
address: 1 George st, Brisbane, QLD
. . .

What is the unique identity?

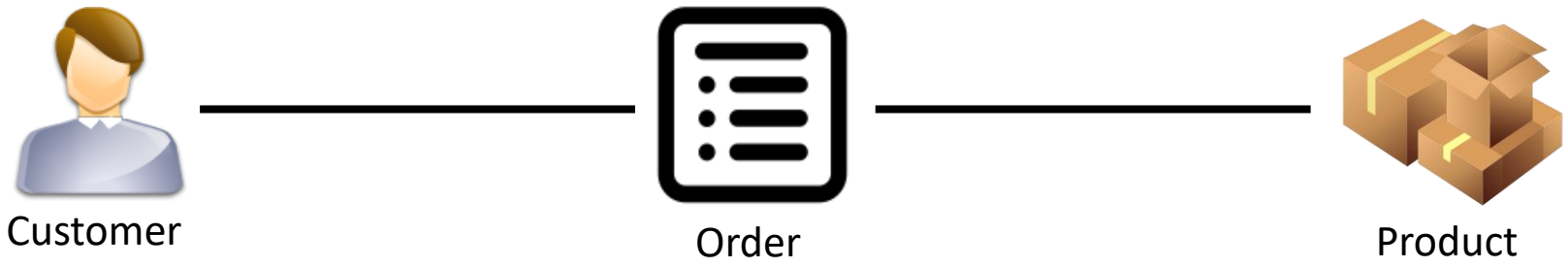
Relationships between classes

- Classes, like objects, do not exist in isolation.
- Three basic kinds of class relationships
  - **Association**: semantic dependency
  - **Aggregation**: whole/part ("part of" relationship)
    - *Composition*
  - **Inheritance**: generalisation/specialisation ("is a" relationship)

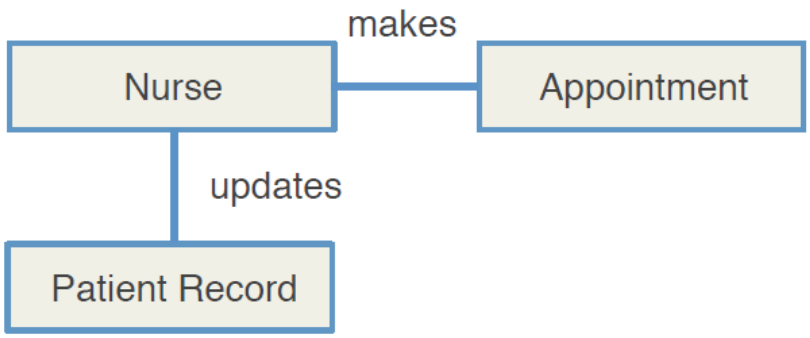
- Examples: consider the following observations
  - *A customer* is a kind of *stakeholder*
  - *An interestedPerson* is a (different) kind of *stakeholder*
  - A *lineItem* is a part of a *reservation order*
  - *Customers* place *reservations*, which may be transmitted to a *travel organisation by* a *travel agency*

Association between classes

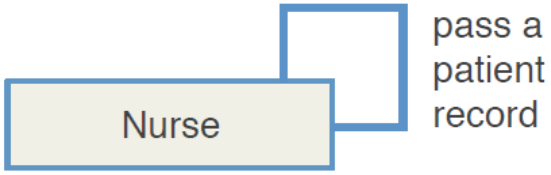
- An association only denotes a semantic dependency and does not (usually) state the direction of this dependency. It is a relationship between multiple classes or a class and itself



[Note: The most general but also the most semantically weak. Sometimes, weak association identified at early design stage may be turned into more concrete class relationships at later stage]



Association: a nurse makes appointments and updates patient records

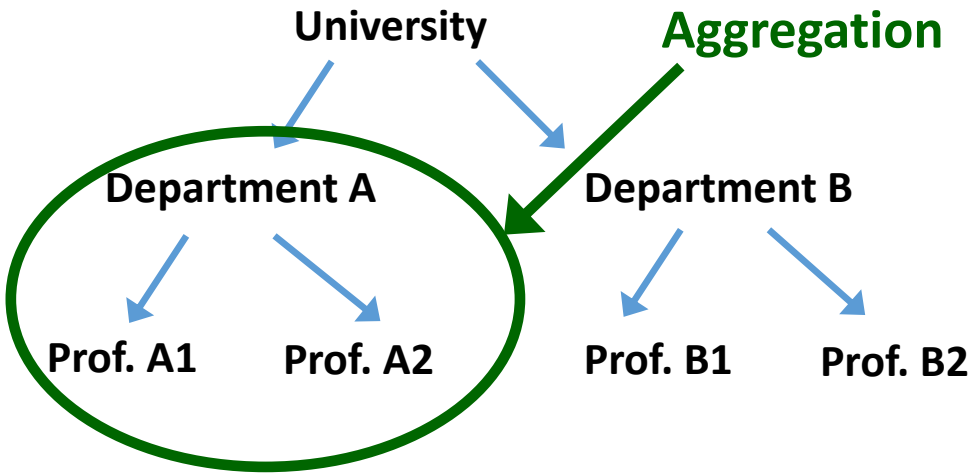


Association: a nurse can pass a patient record to other nurses

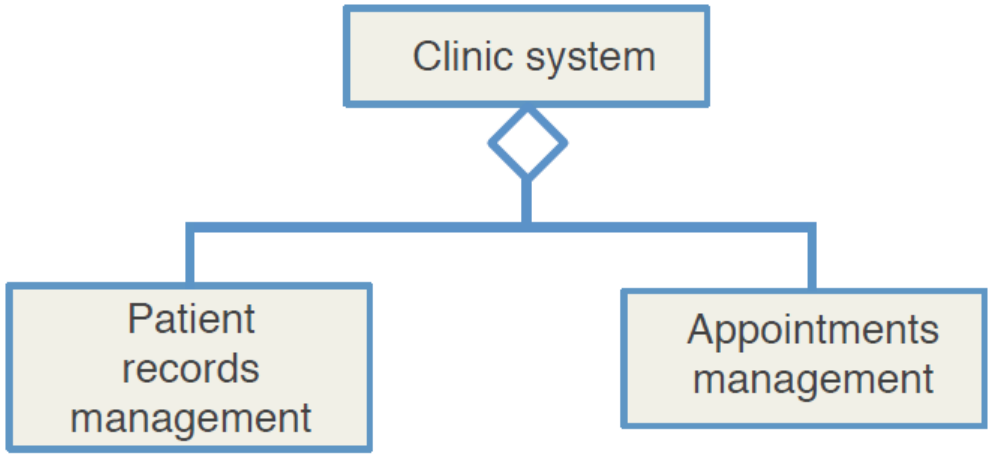
Aggregation

Aggregation *is used to combine simple concepts into more complex ones.*

- A special and directional case of association
- Direction specifies which concept contains the other concept
- Also known as “has-a” relationship between concepts
- **Does not imply ownership,** when the container object is destroyed, the contained objects continue to exist



An example of Aggregation relationship

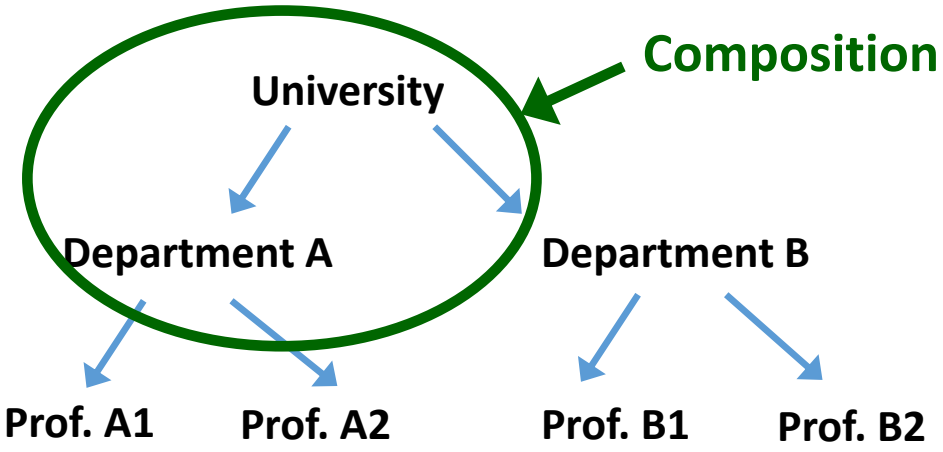


The clinic system has a patients record subsystem management and an appointments management subsystem

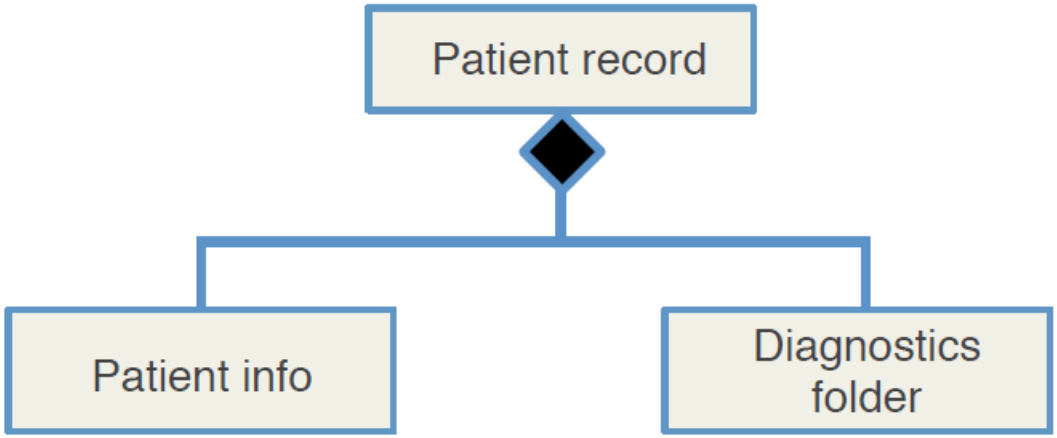
Composition

Composition *is used to combine simple concepts into more complex ones.*

- A special (strong) case of aggregation
- Also known as “part-of” relationship between concepts
- **Implies ownership -** when the containing object is destroyed, so are the contained objects



An example of Composition relationship



A patient record is **composed by** patient info and a diagnostics folder. The patient record **does not exist** without this relationship

Is it a good idea to use Aggregation and Composition relationships?

Can they be substituted by associations with multiplicity? (which you'll learn in a few minutes).

Generalisation/Specialisation

*Concept B is a **specialisation** of concept A if it holds that:*
every instance (i.e. object) of concept B is also an instance of concept A, and

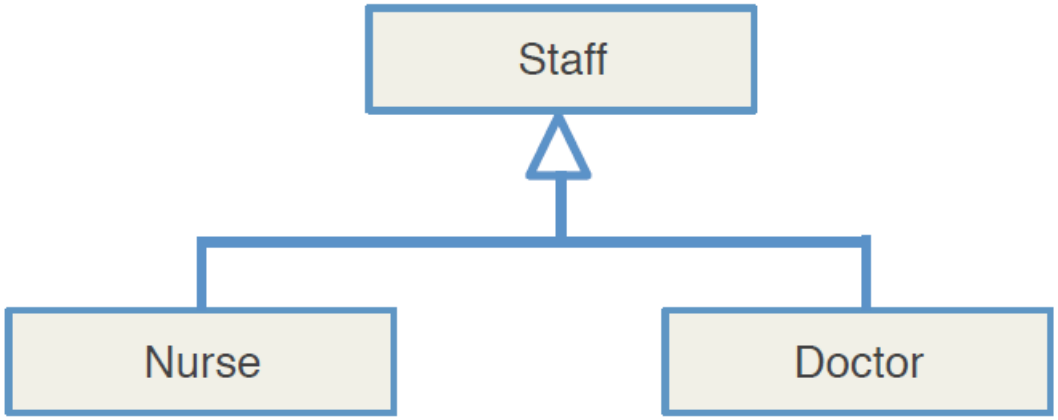
- Also known as “is-a” relationship between concepts
- **Generalisation** is an opposite of specialisation

Bear “is-a” kind of mammal

Hatchback “is-a” kind of automobile

Suggestion: try to focus on use Generalisation rather than specialization. You won’t lose anything if you don’t use specialization relationship.

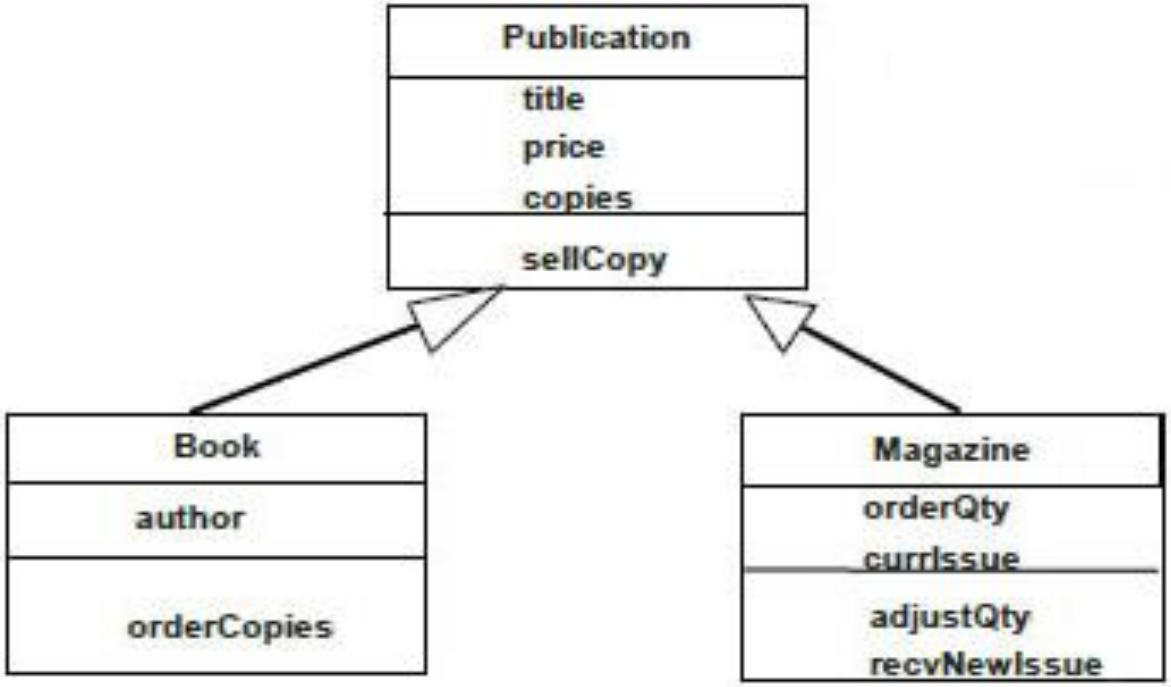
An example of Generalisation

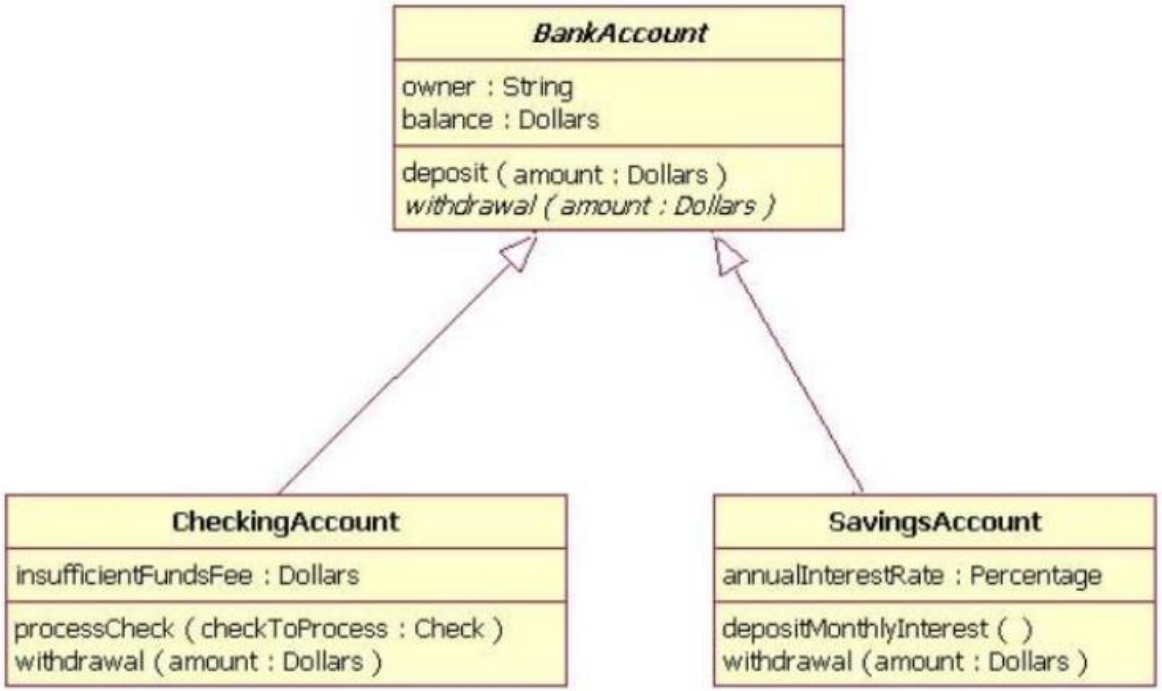


This relationship is also called **parent-child** relationship

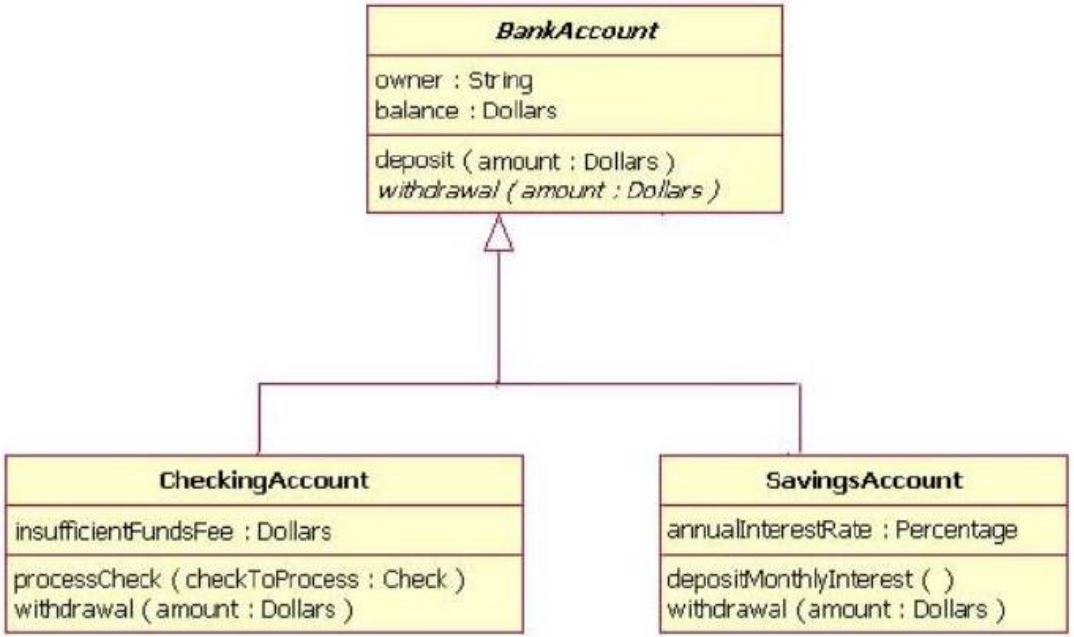
Inheritance expresses generalisation/specialisation relationships (i.e. “is a” relationship)

Do not forget that attributes inherited from a parent should not be represented in the children


Publication
title
price
copies
sellCopy
Book
author
orderCopies
Magazine
orderQty
currIssue
adjustQty
recvNewIssue



Is there any difference between these two diagrams? Are they providing different information?



(IBM): https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

String, percentage, char, integer,…

What are these and do we use them in this unit?

Object-Oriented Approach

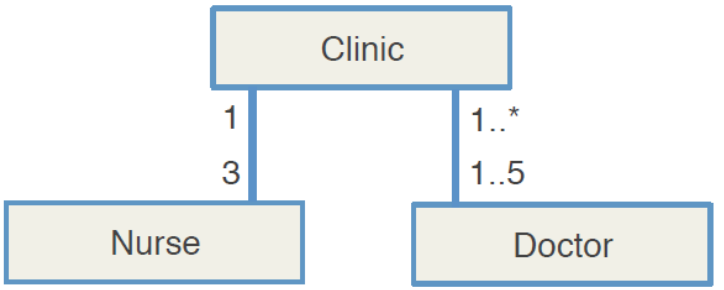
Class Diagram

Multiplicity in Class Diagrams

Wrap up

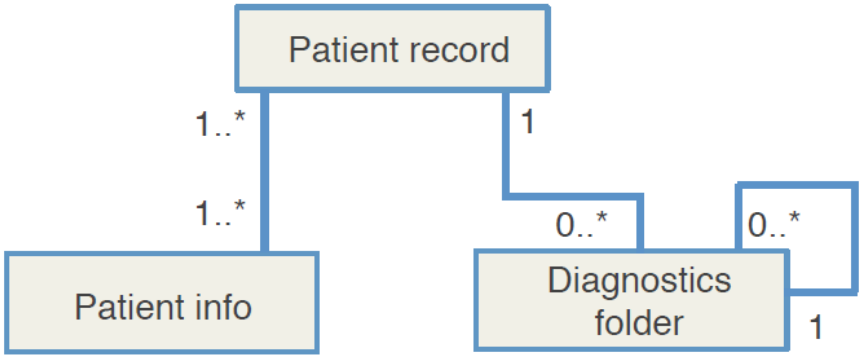
Multiplicity: describes the number of relationships

– One-to-one
– One-to-many
– Many-to-one
– Many-to-many



The clinic has 3 nurses. Each nurse works for 1 clinic

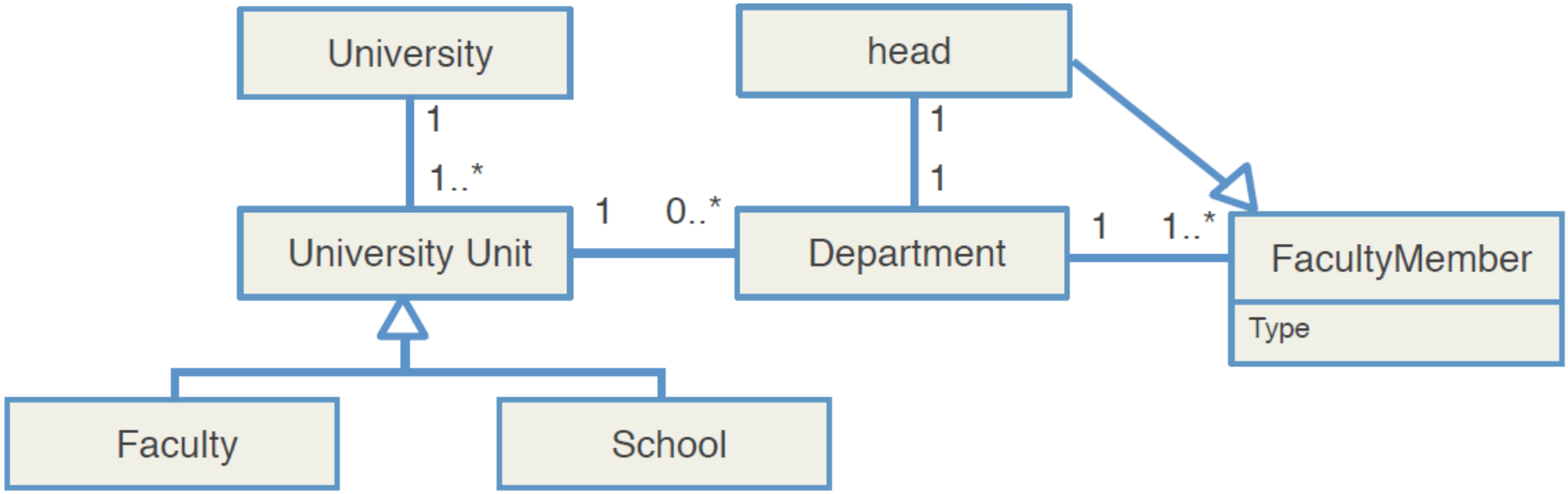
The clinic has 1 to 5 doctors. Each doctor works for at least one clinic



A patient record must have at least 1 patient info. The patient info must belong to at least 1 patient record, but may be related to more patient records

A patient record may not have diagnostics folders. A diagnostics folder belongs to 1 patient record. A diagnostics folder may have other diagnostics folders

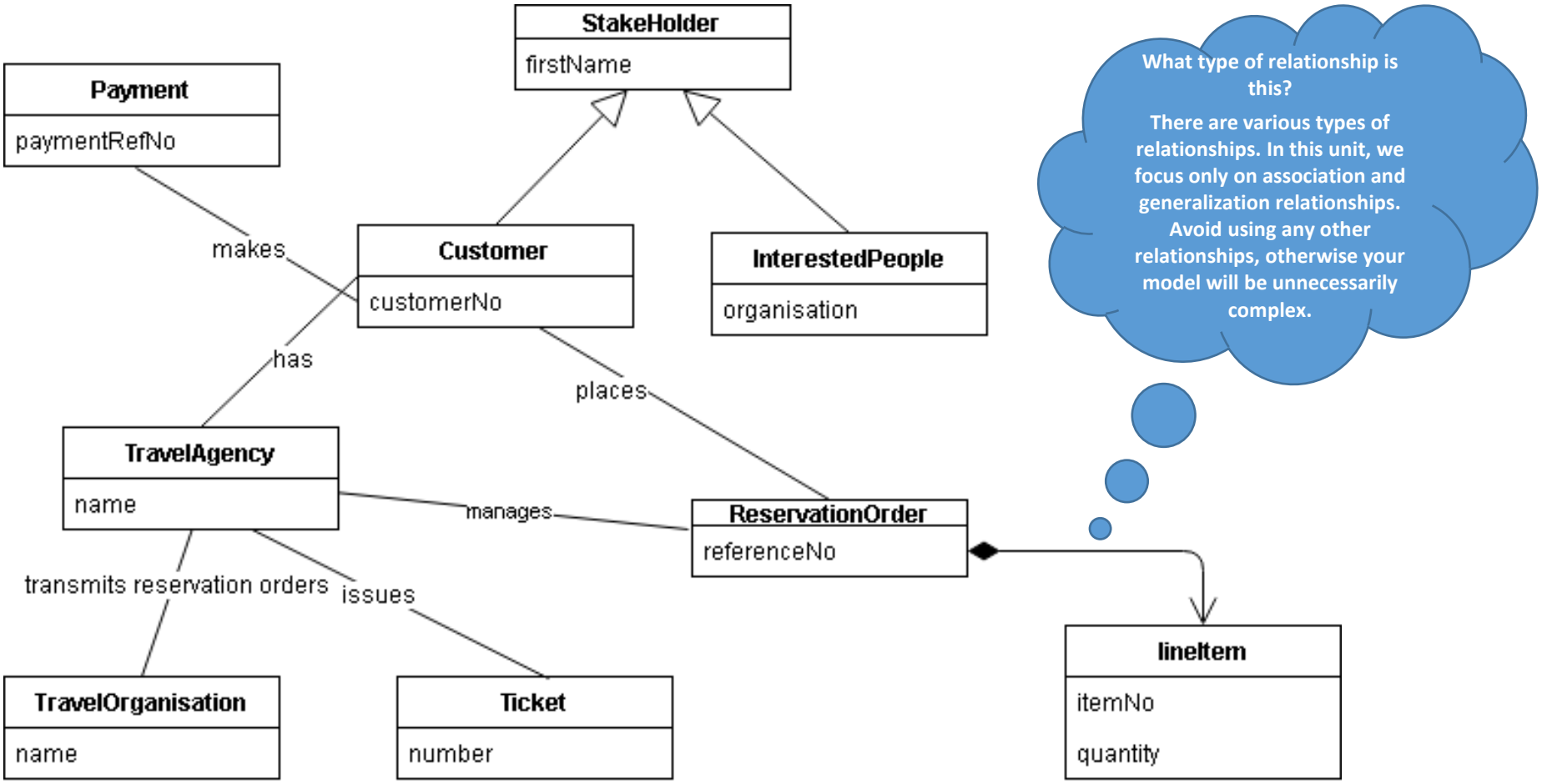
Multiplicity: another example



Some relationships may be named, others not (see the next slide)

- Use naming if the relationship is explicitly named in the case
- Not necessary to name trivial relationships like, has, uses, etc.

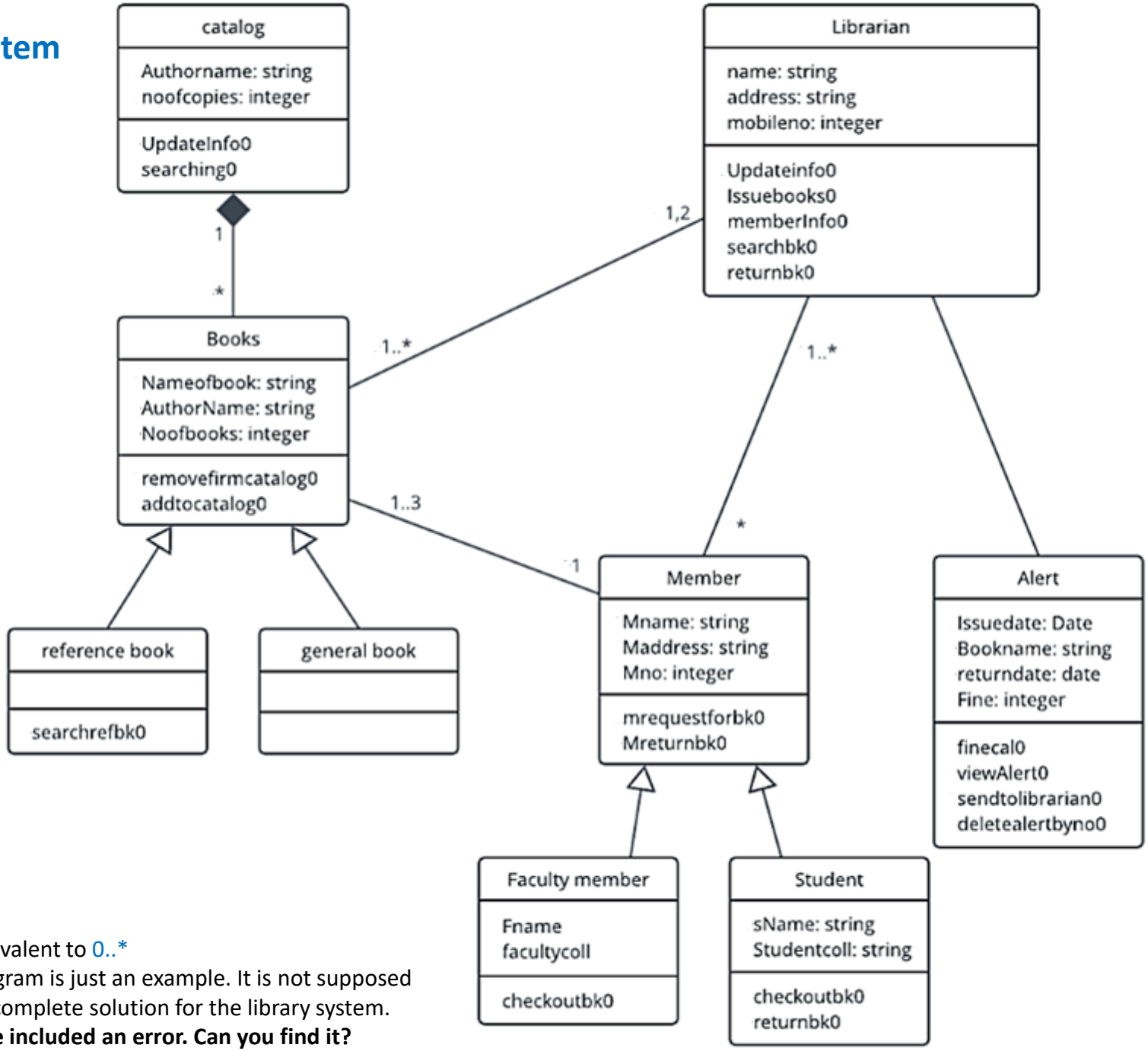
An example – The Travel Agency System



Question:

Interpret the UML class diagram (and multiplicity). See the next slide.

Library system



Note:

- * is equivalent to 0..*
- This diagram is just an example. It is not supposed to be a complete solution for the library system.
- **We have included an error. Can you find it?**

https://www.lucidchart.com/blog/uml-diagram-templates

Sometimes we see – and + signs in a class diagram.

Do we need to use them [in this unit]?

Template for your Class Diagram is available on Blackboard. Also, the DC2 assessment specification file provides guidance on how to complete the template.

Object-Oriented Approach

Class Diagram

Multiplicity in Class Diagrams

Wrap up

Drawing it all together

There are several types of UML diagrams. Each UML diagram has a specific purpose. They ‘together’ provide information on flow of data ‘and’ what happens within a process.

We have covered Use Case Diagrams and Class Diagrams. We will cover Activity and Sequence Diagrams next week.

Acknowledgment

The examples about e-book, health record management and university structure are based on the content taught at Victoria University of Wellington as a part of systems analysis and design course in 2013-14. Associate Professor Pedro Antunes (coordinator) and Alireza Nili were the teachers of the course. The rest of examples are from the relevant lecture slides used in 2017 onwards at QUT.

Questions?

a.nili@qut.edu.au