

# CIS\*3750 - System Analysis and Design in Applications

Luiza Antonie, Fall 2025, University of Guelph

# Week 10 Demo (Nov 3 & 5)

- In person, during your regular lab session
- 10-minute live demo of your high-fidelity prototype
- It must showcase **2-3 key user walkthroughs** focused on the core "intelligent" learning functionality of your ITS
- Simple walkthroughs like account creation or logging in will not be considered a valid main walkthrough
- While a code-based UI is recommended, navigating a Figma prototype is also acceptable
- Provide a shareable link to your prototype (e.g., GitHub or Figma link) for review. The entire group must be present for the demo.

# Unified Modeling Language (UML)

- UML is a modeling language in software engineering
- It provides a standard way of visualizing the design of a system



# Class Diagram

- A type of static structure diagram
- Describes the static structure of a software system
- It shows the system's classes, their attributes, operations (or methods), and the relationships among the classes.

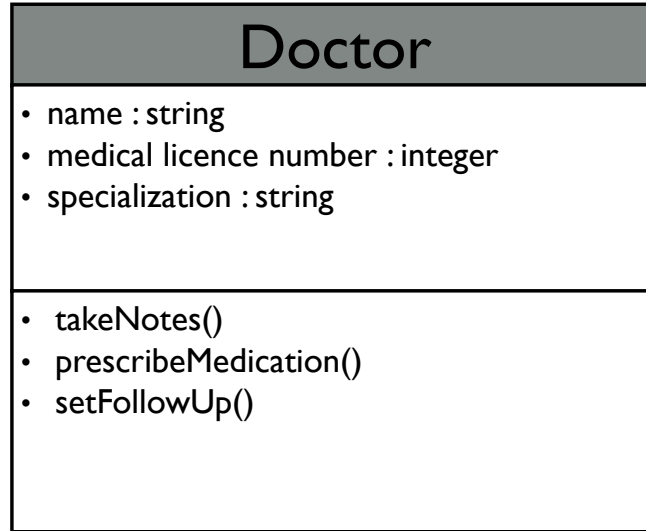


# Class Diagram

- A class is a description for a group of objects:
  - similar properties (attributes)
  - common behaviour (methods)



# Class Diagram



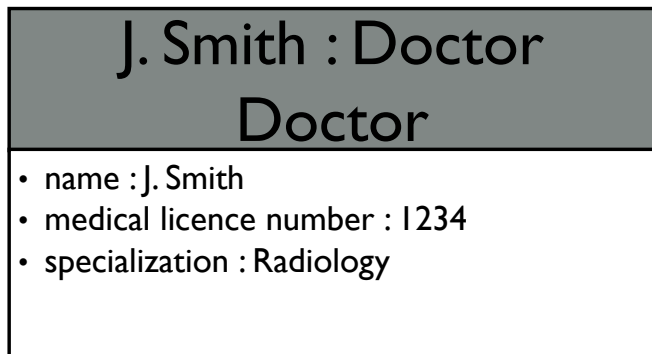
*Class Name*

*Attributes*

*Methods*

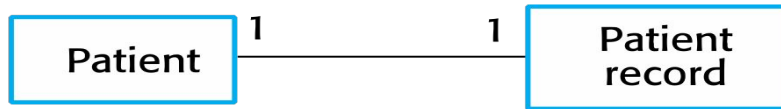
# Class Diagram

- The instances of a class are called objects
- What is an instance?
  - An instance is a realization of a class
  - It's the thing that is built based on the blueprint



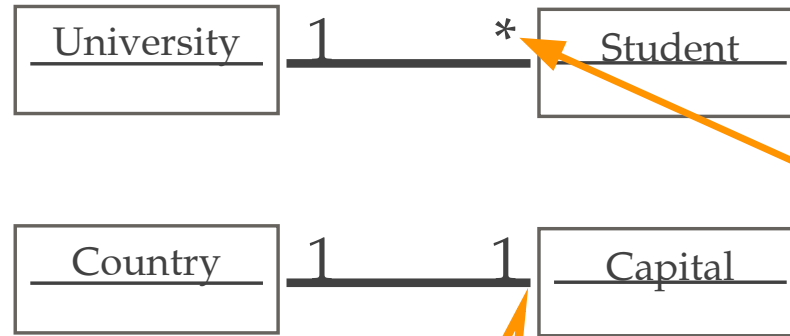
# Class Diagram - Relationships

- Association (line)
- Inheritance (open triangle)
- Aggregation (open diamond)
- Composition (solid diamond)





# Class Diagram - Association



**Multiplicity**  
A university has one or more students ( $0..* = *$ )

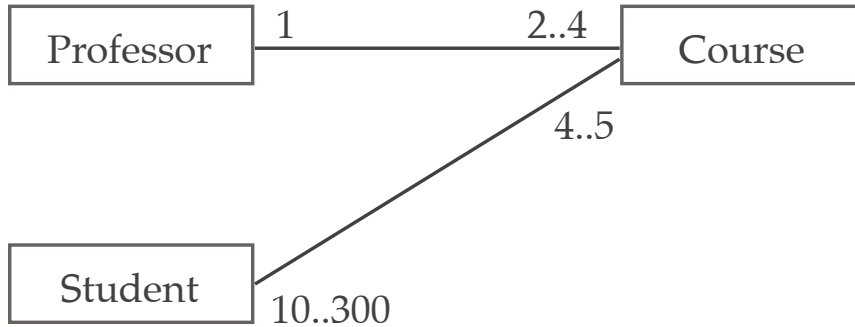
**Multiplicity**  
A country has one capital

# Class Diagram - Multiplicity

- Exactly 1  $\longrightarrow$  1
- Zero or more  $\longrightarrow$  0..\* or \*
- One or more  $\longrightarrow$  1..\*
- Zero or one  $\longrightarrow$  0..1
- A range of values  $\longrightarrow$  1..3



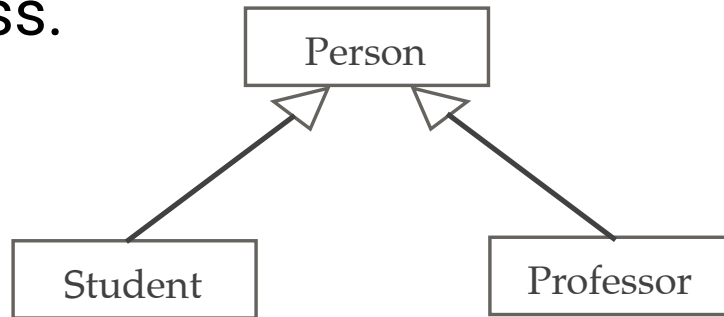
# Class Diagram - Association



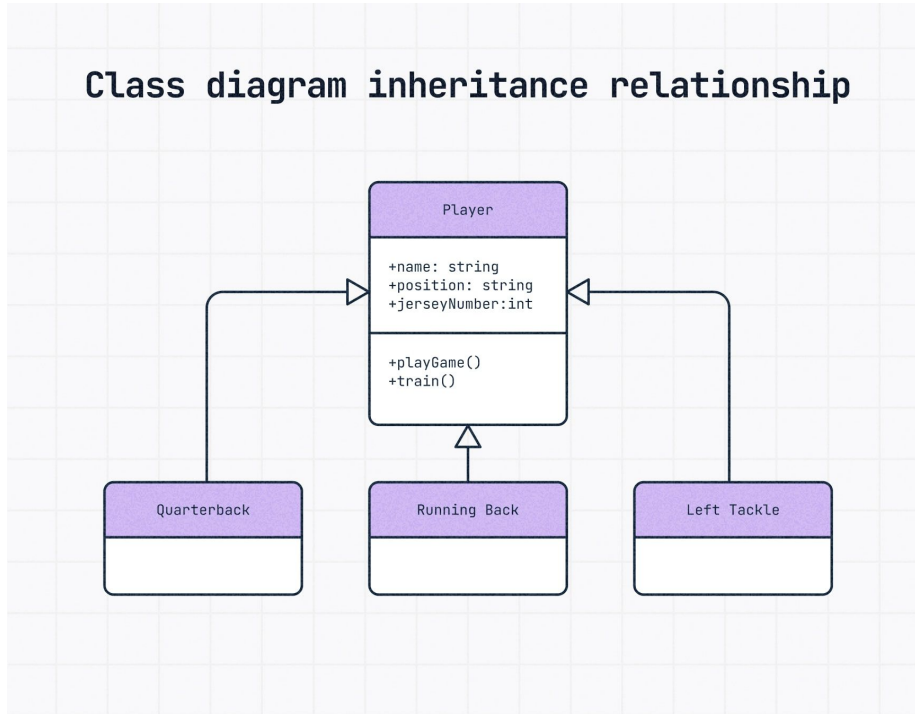
# Class Diagram - Generalization/Inheritance

**Generalization:** the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass.

**Inheritance:** classes “inherit” the attributes and methods of the superclass.



# Class Diagram - Generalization/Inheritance



# Class Diagram - Generalization/Inheritance

- Generalization is an everyday technique that we use to manage complexity.
- Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, etc.) and learn the characteristics of these classes.
- This allows us to infer that different members of these classes have some common characteristics (e.g., squirrels and rats are rodents).

# Class Diagram - Generalization/Inheritance

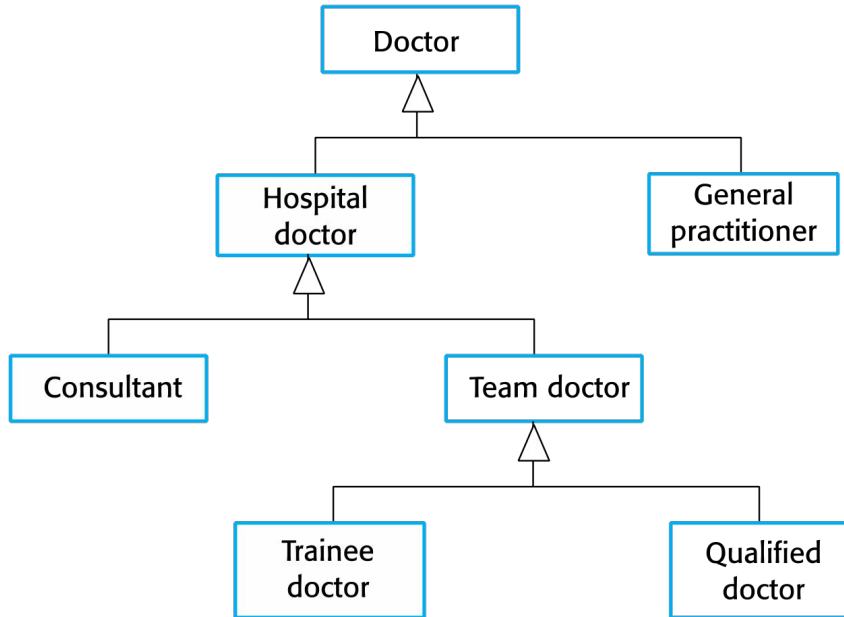
- In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.

# Class Diagram - Generalization/Inheritance

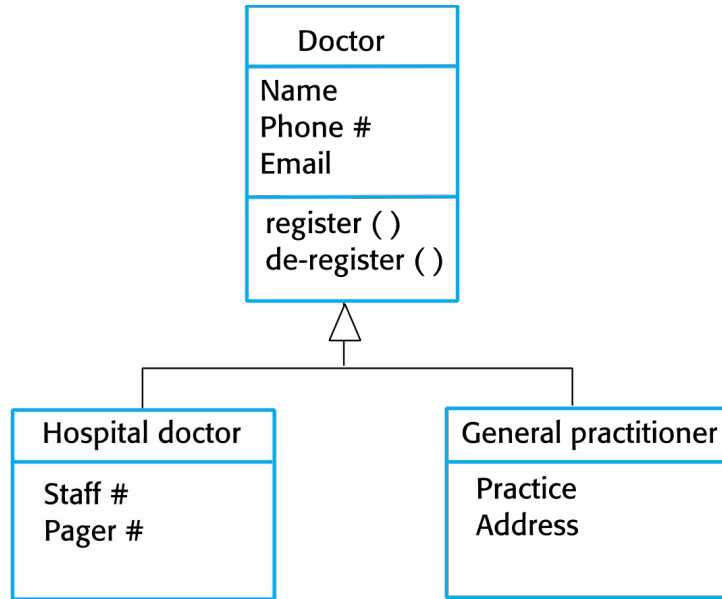
- In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level classes.
- The lower-level classes are subclasses inherit the attributes and operations from their superclasses. These lower-level classes then add more specific attributes and operations.



# Class Diagram - Generalization/Inheritance



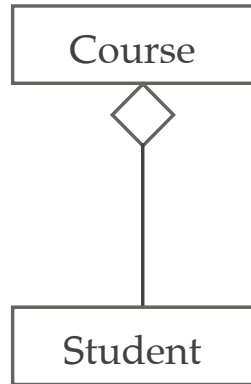
# Class Diagram - Generalization/Inheritance



# Class Diagram - Aggregation

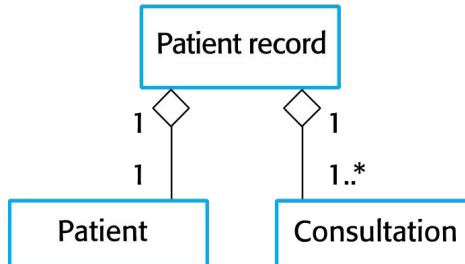
**Aggregation:** a relationship where a “child” class can exist without the “parent” class

**Aggregation:** is a “has-a” or “whole/part” relationship



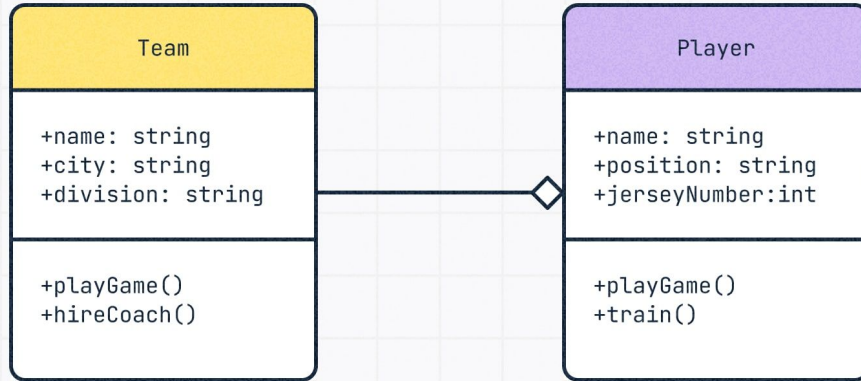
# Class Diagram - Aggregation

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.



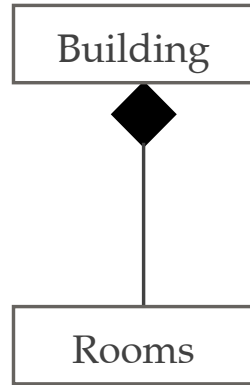
# Class Diagram - Aggregation

Class diagram aggregation relationship



# Class Diagram - Composition

**Composition:** a relationship where a “child” class cannot exist without the “parent” class



# Class Diagram - Composition

Class diagram composition relationship



# Class Diagrams


- When designing our classes, we need to consider if they are sufficient and necessary
- We also need to consider which class takes responsibility for certain things.





# Class Diagrams - Activity

Create a class diagram to capture the following:

- A university has different colleges. Each college has several departments.
  - Colleges and departments have names; Departments have classrooms and offices.
  - A person studying or working at the university has a unique ID; they can be students; faculty or staff.
  - Students can be undergraduate or graduate
  - Faculty can be sessional, assistant, associate or full.
  - Faculty is employed by one department only.
  - Offices and classrooms have numbers; Classrooms have numbers of seats.
  - Every faculty works in an office.
- 

# Class Diagrams - Activity

A small tennis club is asking you to build a web application that would allow customers to book tennis lessons and courts online. In order for the customer to be able to book lessons and courts online, they must have an account. The customer creates an account by filling their personal information and choosing a username/password combination. In addition, their address and credit card number can be linked to the account, but not required as they can fill in that information at the time of booking. The tennis club manager should be able to update the tennis lessons and court availability. The tennis club manager should be able to contact the customers by email or phone, in case there are changes to the lessons' schedule. The customers can search for tennis lessons by level and time, read descriptions about the lessons and add lessons to the shopping cart. The customers should be able to pay for lessons with credit card or through PayPal.

# Class Diagrams - Activity

- Candidate classes - identify nouns
  - customer, tennis club manager, lesson, booking, account
- Candidate methods/behaviours - identify verbs
  - Create an account, book lesson, update lesson, contact customer, search lessons
- Think about relationships
  - customer - booking
  - customer - account
  - booking - payment
  - manager - lesson

