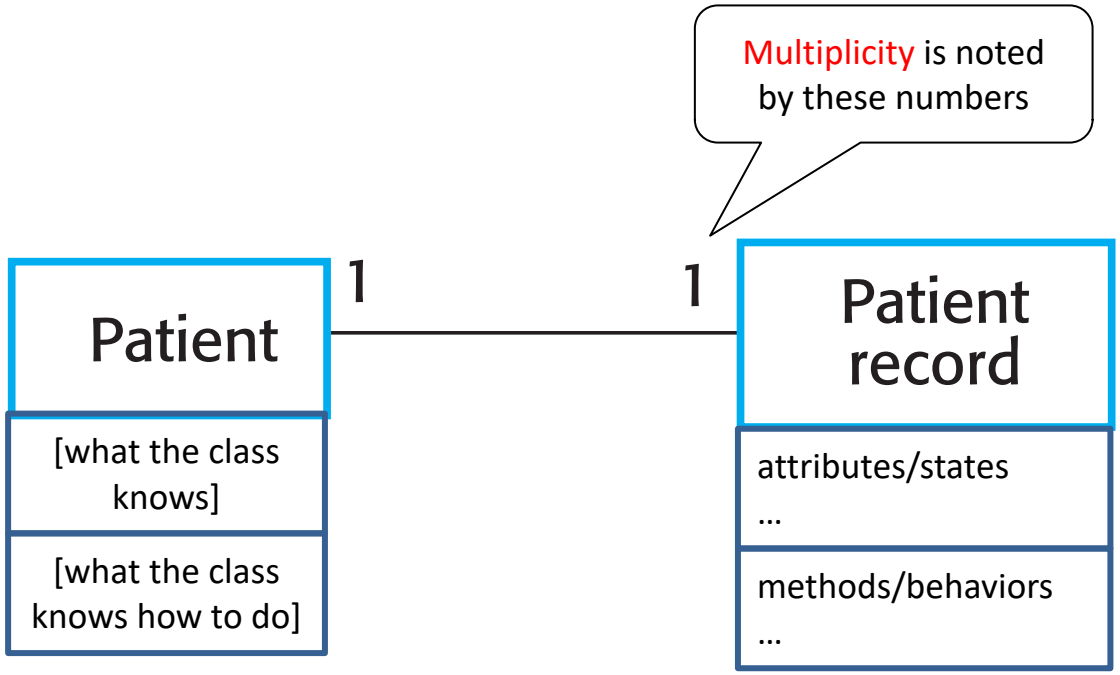
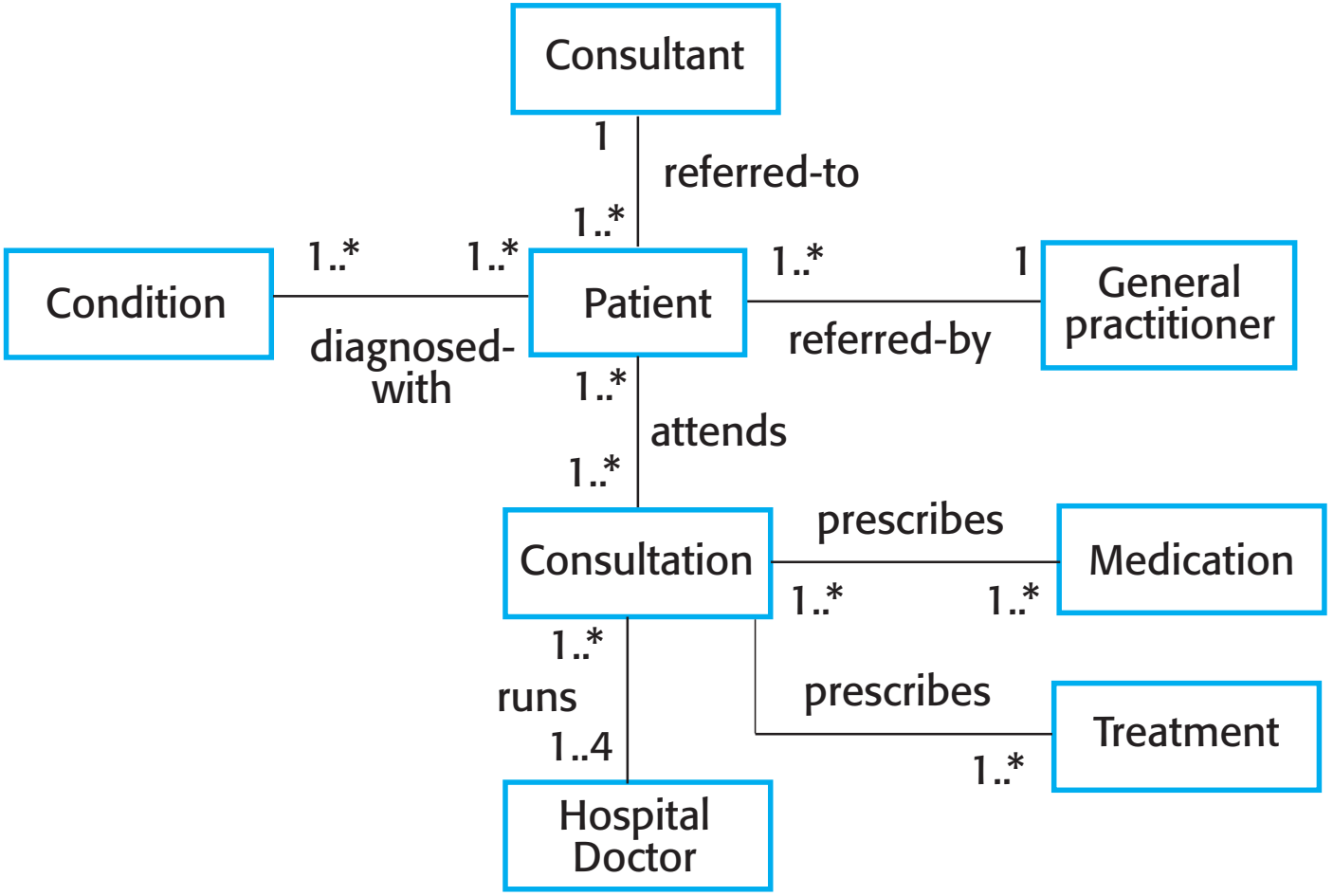
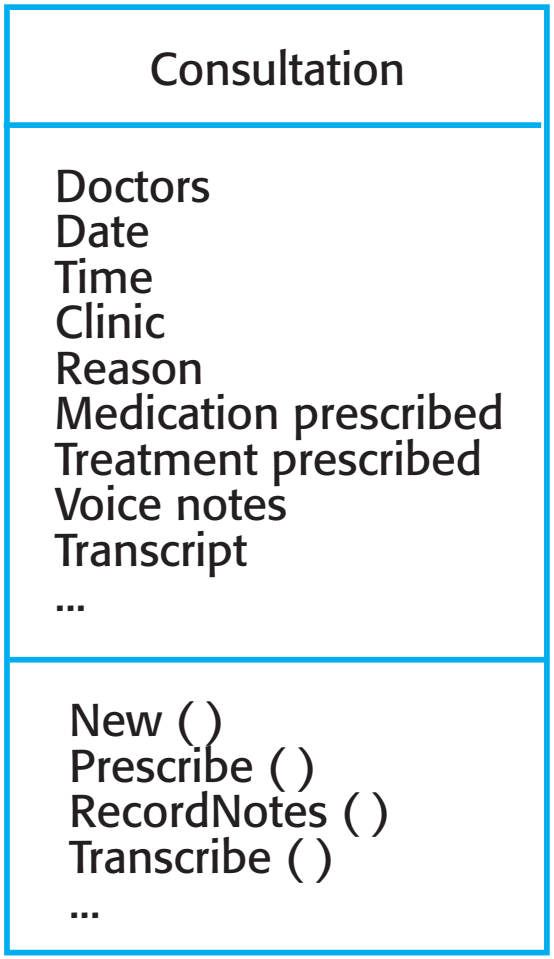
UML classes and association



Classes and associations in the MHC-PMS



The Consultation class



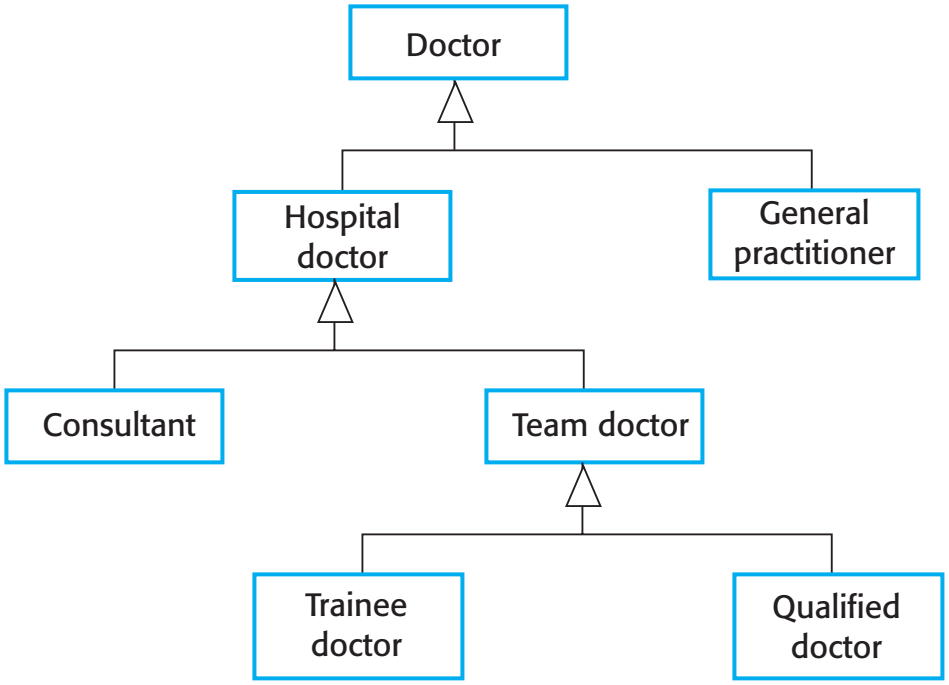
Generalization

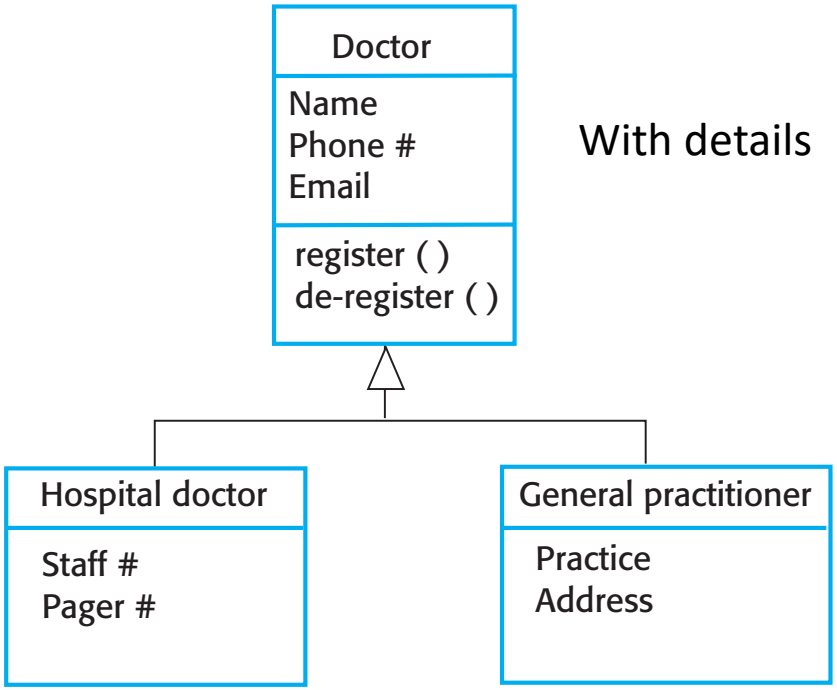
- Generalization is an everyday technique that we use to manage complexity.
- Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, transactions, etc.) and learn the characteristics of these classes.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

Generalization

- 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.
- In object-oriented languages, such as Java and Python, generalization is implemented using the class inheritance mechanisms built into the language.
- In a generalization, the attributes and operations associated with higher-level classes are *inherited* by the lower-level classes.
- The lower-level classes then add more specific attributes and operations.

2 generalization hierarchies

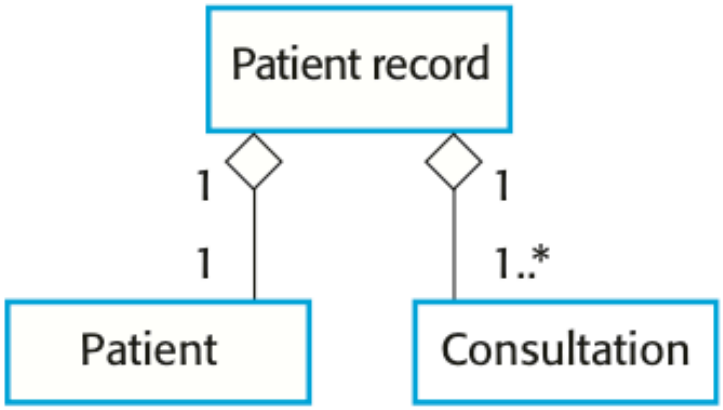


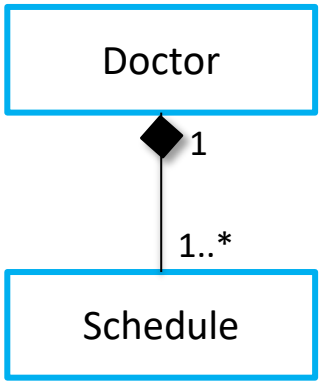


With details

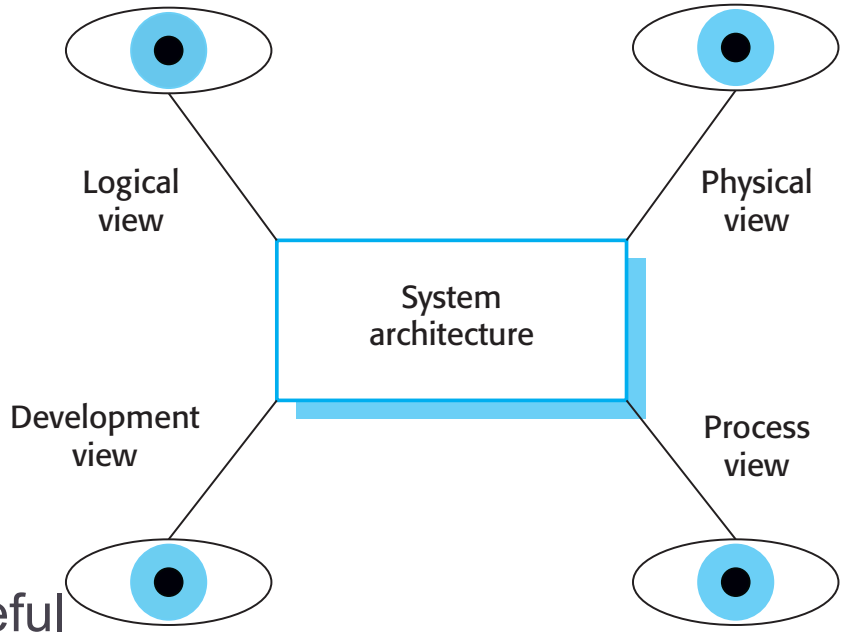
Object class aggregation models

- An aggregation model shows how classes can be composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.
- Composition is a form of aggregation.
- Compositions are whole-part relationships where the lifetime of the part is dependent on the lifetime of the whole.





Architectural/modeling views



- What views or perspectives are useful when designing and documenting a system's architecture?
- A view (a diagram) is only a single rendering of some aspect of the software
- Each model captures a perspective useful when describing and understanding the system.
  - It might show how a system is decomposed into modules, how the run-time processes interact or the different ways in which system components are distributed across a network.
  - For both design and documentation, you usually need to present multiple views of the software architecture.

4 + 1 view model of software architecture (Kruchten)

Logical View
Objects and object classes - the system's data and methods

Process View
An "alive view" of interacting processes

Use case views/scenarios
Requirements used as test cases for the architecture

Physical View
Software distributed across hardware; communications

Development View
Software decomposed for development

Summary

- Modeling offers abstract views of what the software is and what it will do based on the requirements
- Unified Modeling Language (UML) is widely used for software system modeling
- Software system structures are often captured using object class diagrams.
- Use case and sequence diagrams often used to capture interactions
- State transition diagrams often used to capture event-driven sequences
- Multiple diagrams (views) are needed to describe and understand the software design.

Questions?