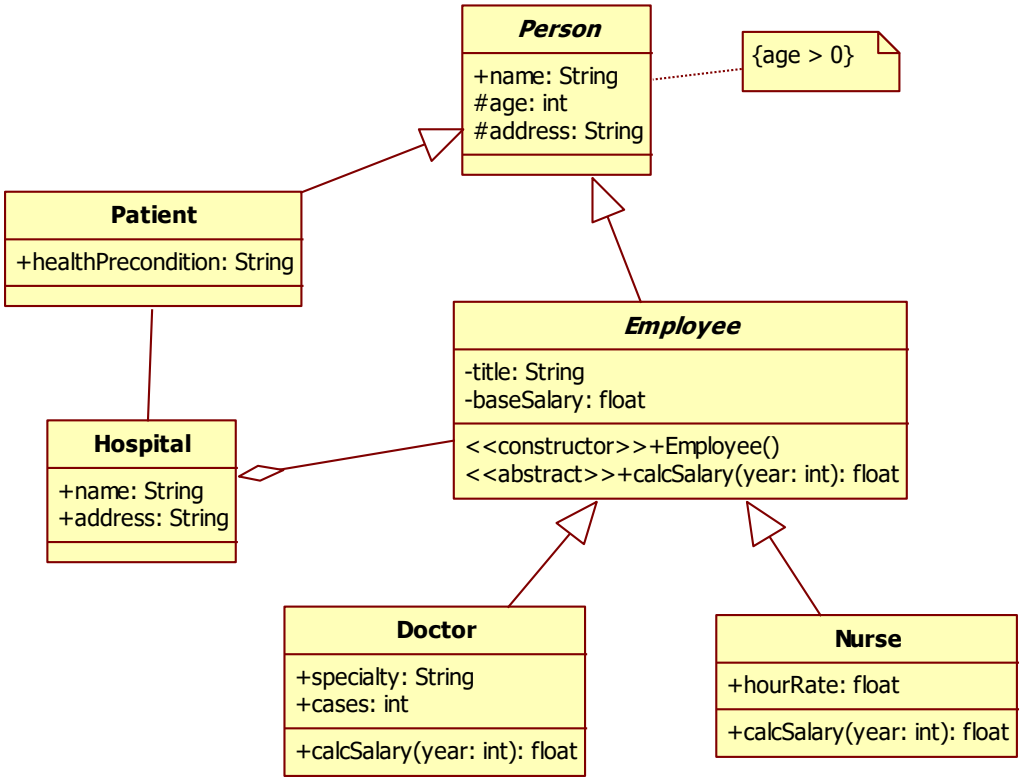
Topics

- Introduction to UML
- Use case diagrams
- Class diagrams

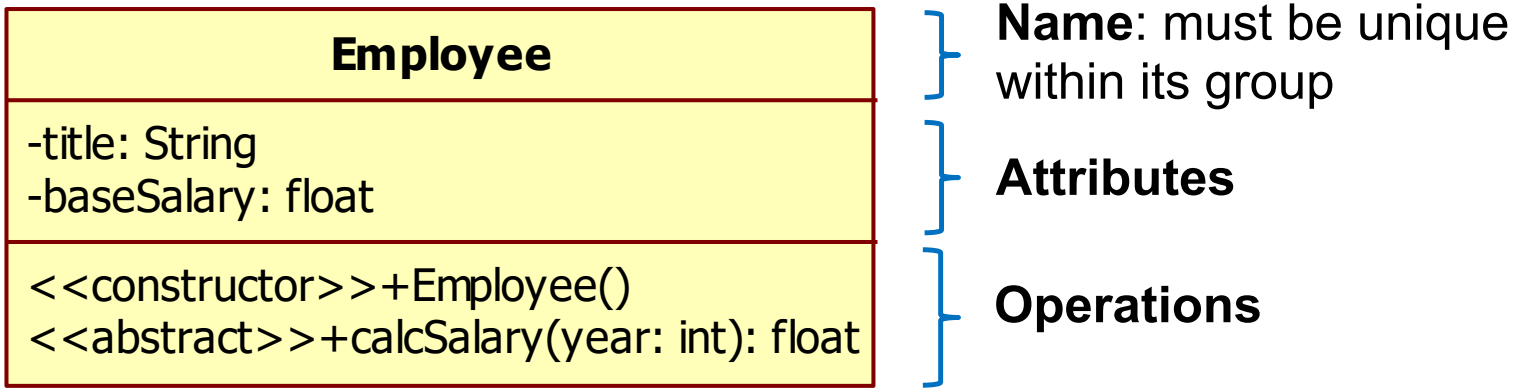
Class Diagram

- The most commonly used diagram in practice
- Main elements
  - Classes
  - Interfaces
  - Relationships
  - Common mechanisms



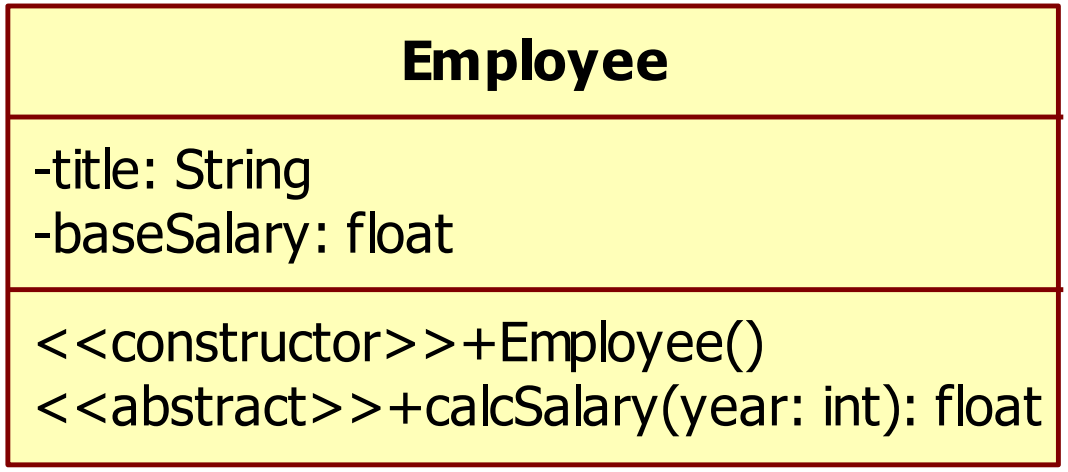
Class

- Defines the set of common objects that have same the same attributes, operations, relationships, and semantics
- Represents a thing
- Notation



Attribute

- Defines data that characterize a class
- An abstraction of the kind of data or object
  - *title* is an attribute of the kind of *String* object
- Data type is specified by a semicolon ":"

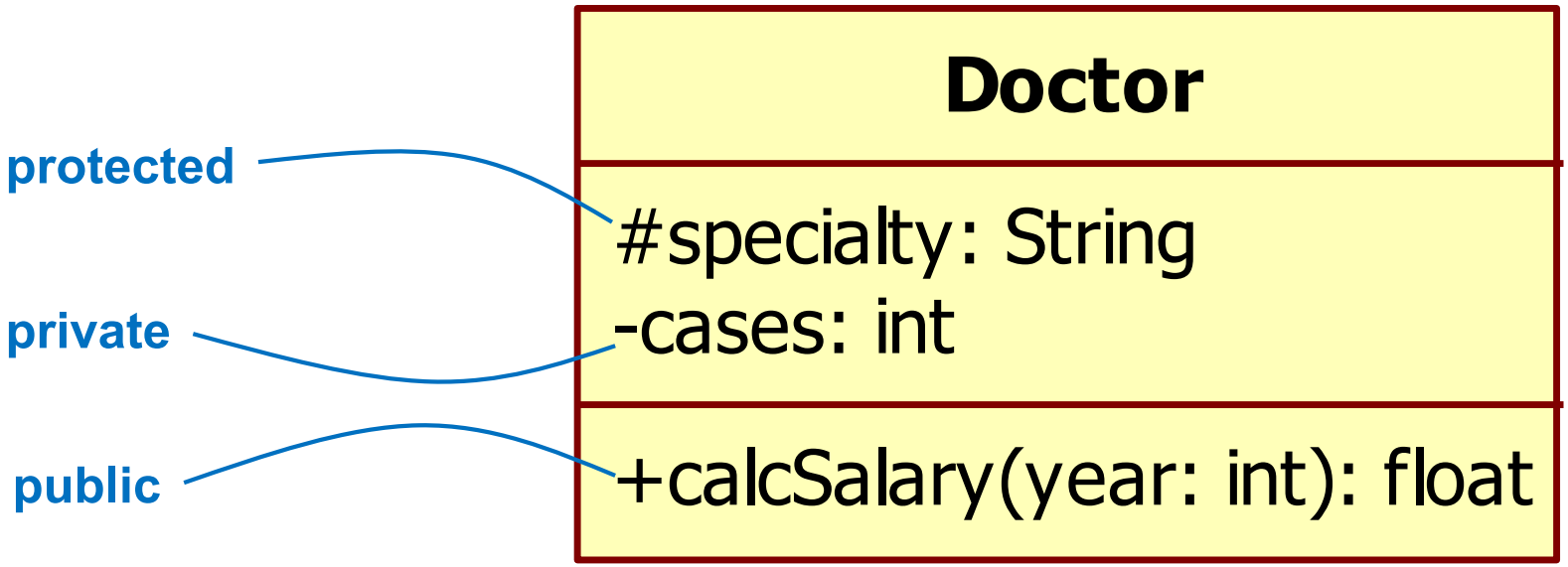


Title and *baseSalary* are two attributes of *String* and *float* data types, respectively

Operation

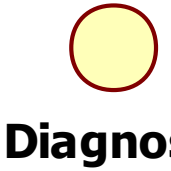
- An operation specifies a service that can be requested from objects of the class

- Attribute and operation visibility

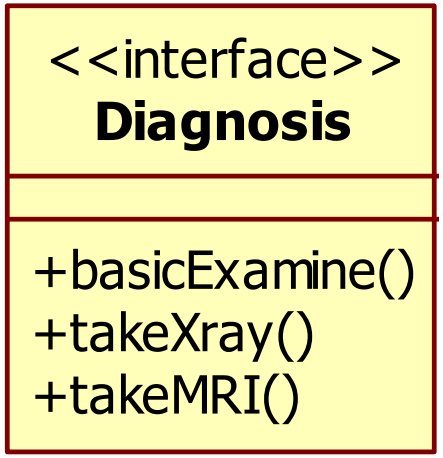


Interface

- A element that has a set of operations characterizing its behavior
- Notation
  - A circle with a name
  - OR a class with stereotype <<interface>>



Diagnosis



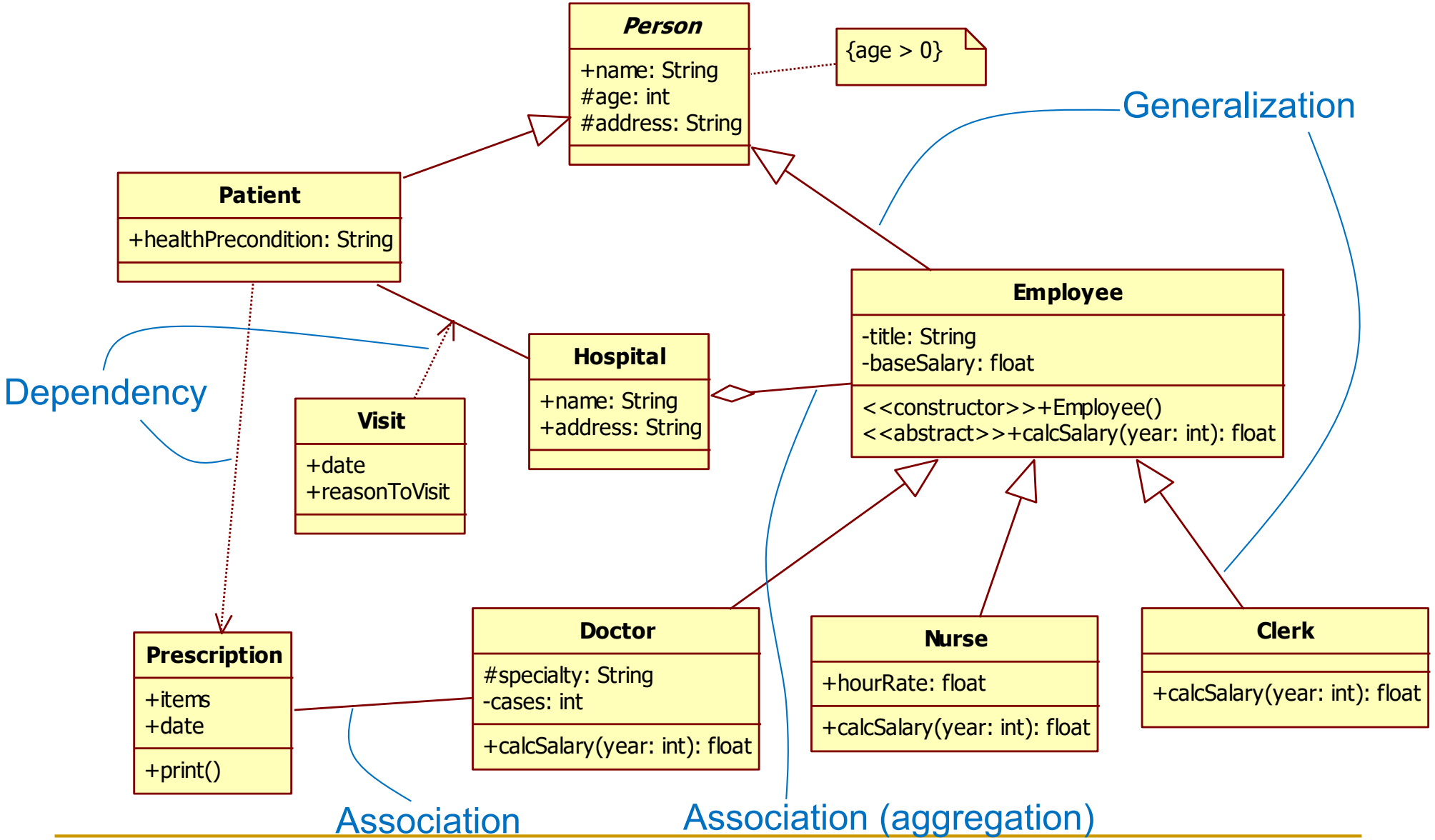
Class Diagram

- The most commonly used diagram in practice
- Main elements
  - Classes
  - Interfaces
  - Relationships
  - Common mechanisms

Class Relationship Types

- Most classes associate with others
- Classes have different types of relationship with each other
  - Dependency
  - Generalization
  - Association

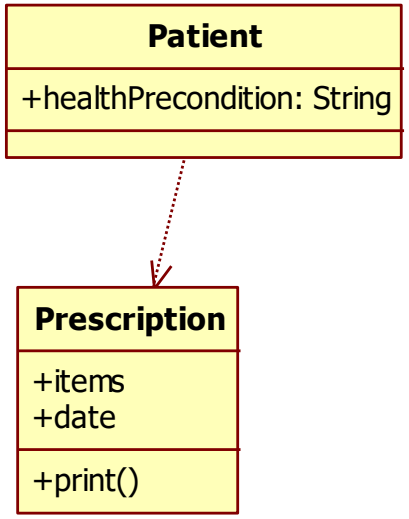
Class Relationship Types (cont'd)



Dependency Relationship

- Represents one class uses another
  - Change in one class may affect the one that uses it
  - e.g., two classes have dependency relationship if
    - one operation calls another operation on another class
    - one class is used as a parameter in another class

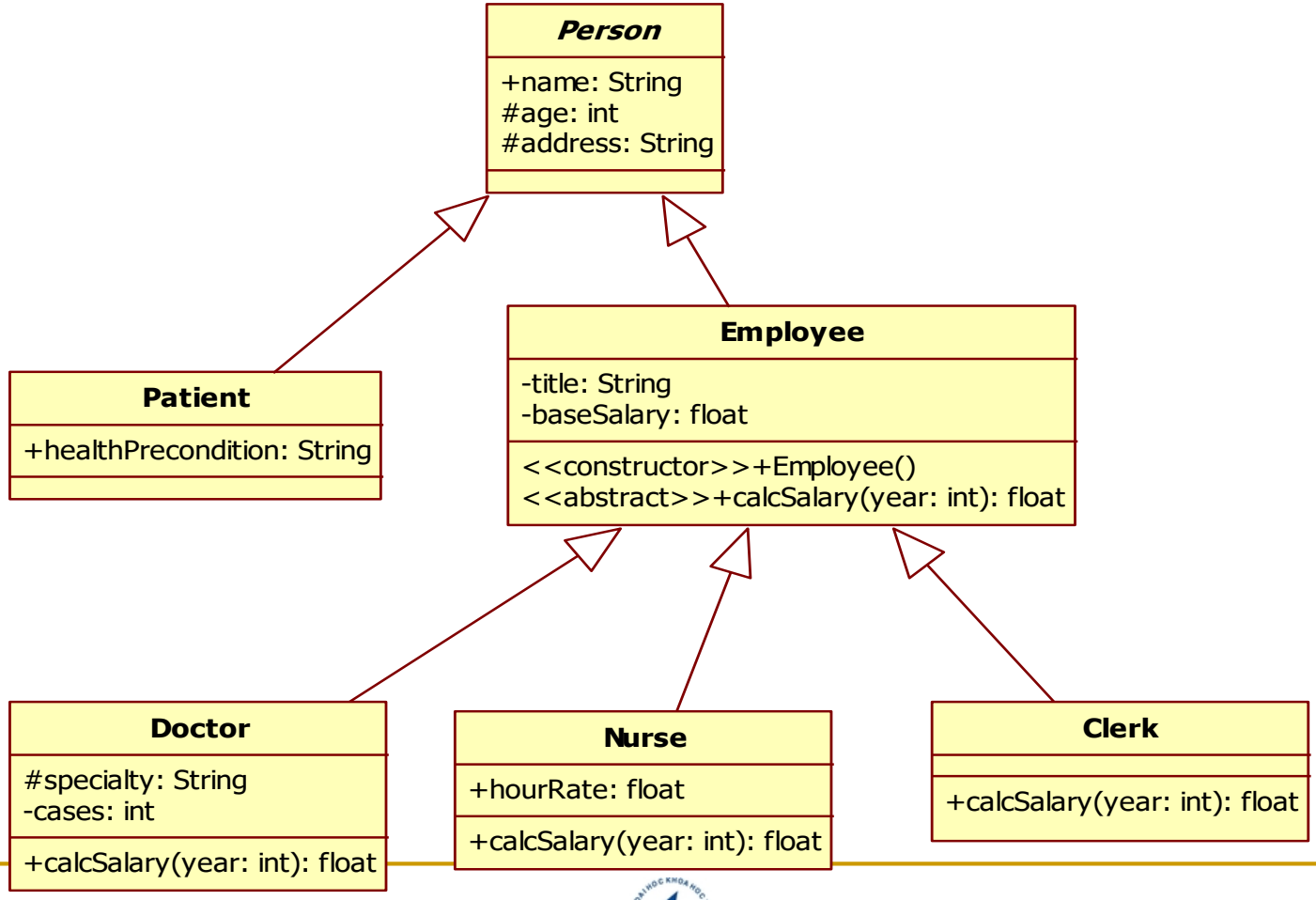
- Dependency is less restrictive than other relationships



Patient uses *Prescription*

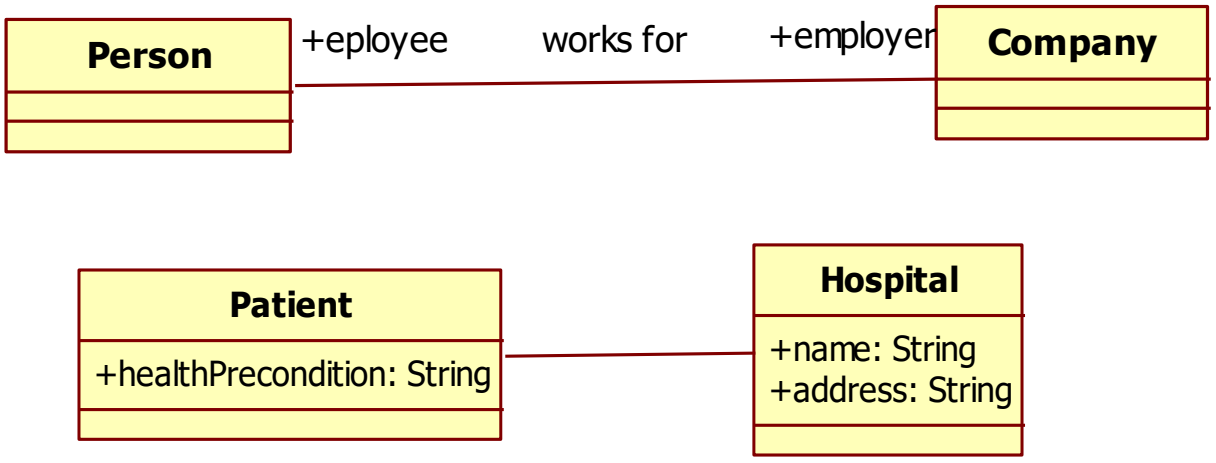
Generalization Relationship

- A relationship between a general class and more specific class (superclass and subclass)



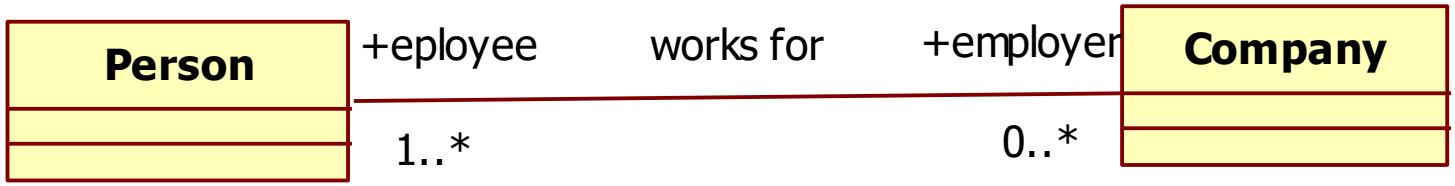
Association Relationship

- A relationship specifying objects of one class are connected to objects of another
- Typically, one object holds objects (instances) of the same class or another
- Association can have a name, roles at both ends



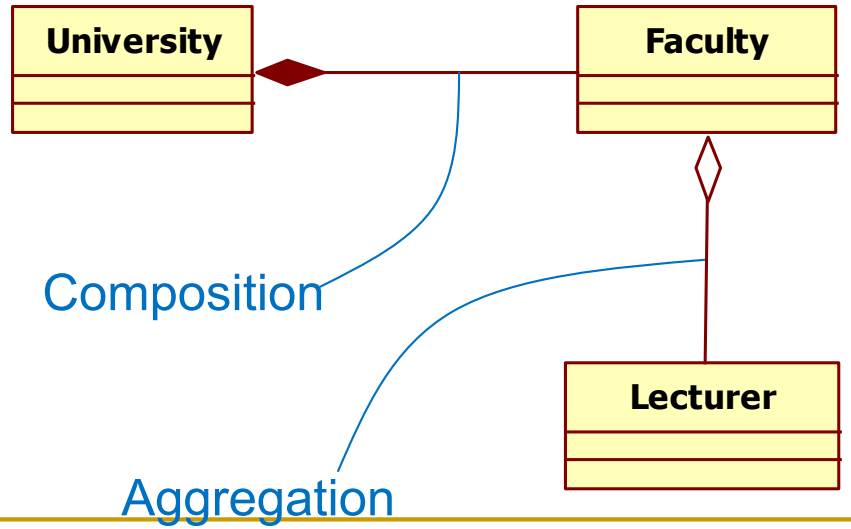
Multiplicity

- Refers to how many objects may be connected across an instance of an association



Aggregation and Composition

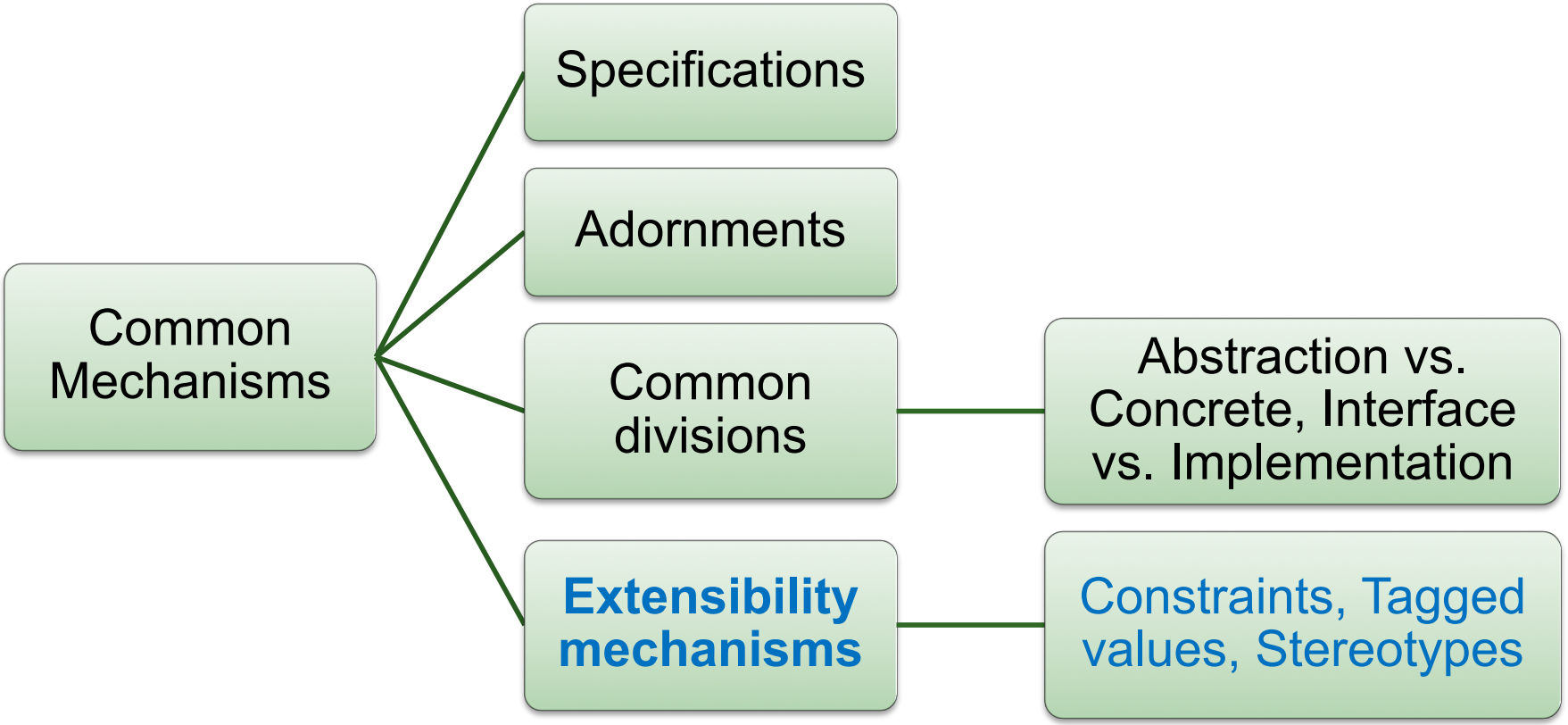
- Two special types of the association that one object of the whole has objects of the part
- Composition
  - The part may belong to only one whole class (composite)
- Difference between the relationships below?



Class Diagram

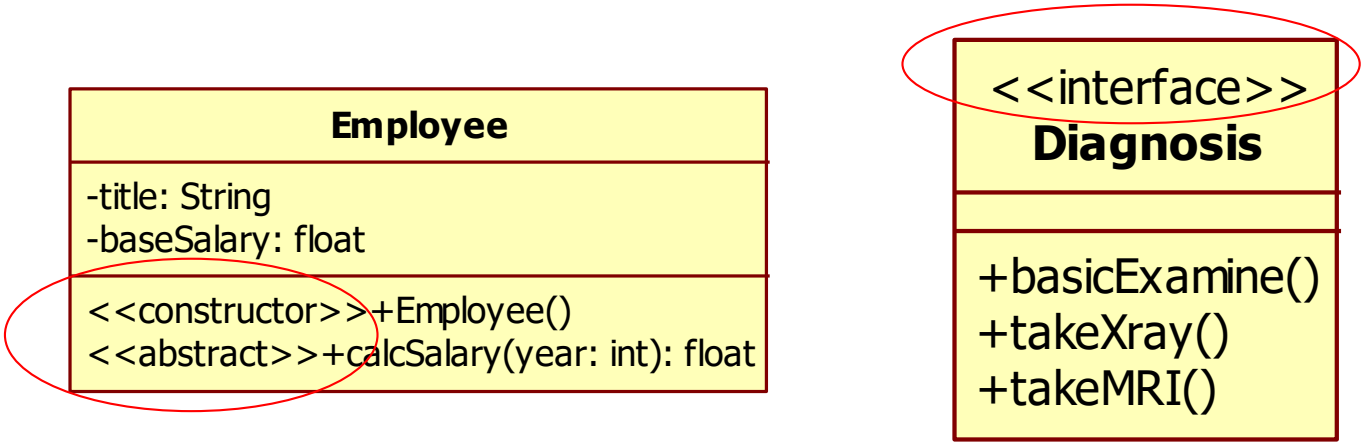
- The most commonly used diagram in practice
- Main elements
  - Classes
  - Interfaces
  - Relationships
  - Common mechanisms

Common Mechanisms



Stereotypes

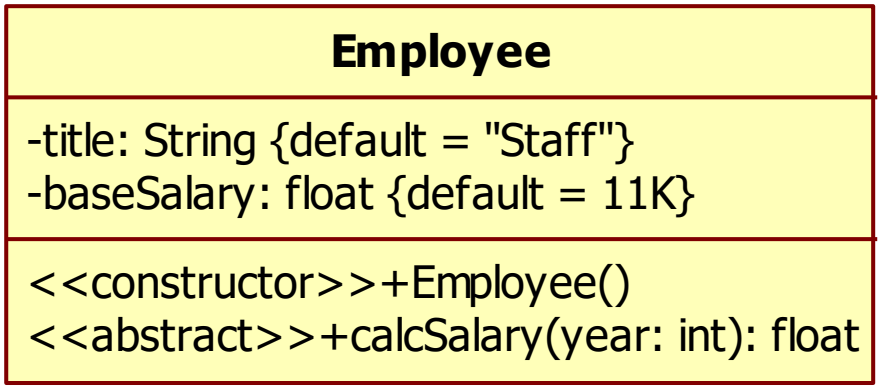
- An extension of the UML vocabulary provides specifics to an existing UML element
- It can be specified for class, attribute, operation, relationship, etc.
- A stereotype is specified within << >>



Tagged Values

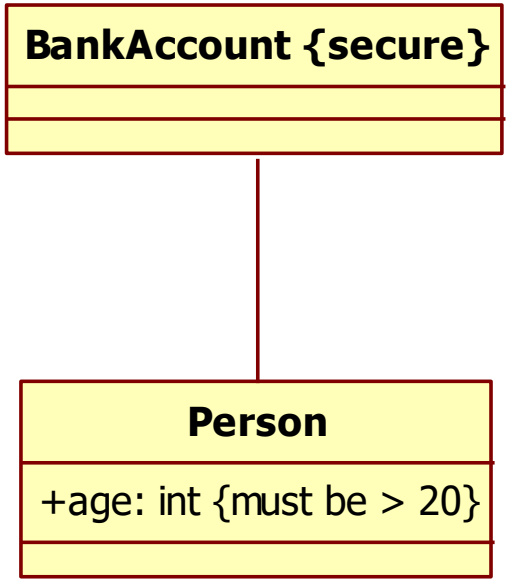
- Allows to provide extensions to a property of a UML element, e.g., specifying default values, ranges, etc.
- Notation

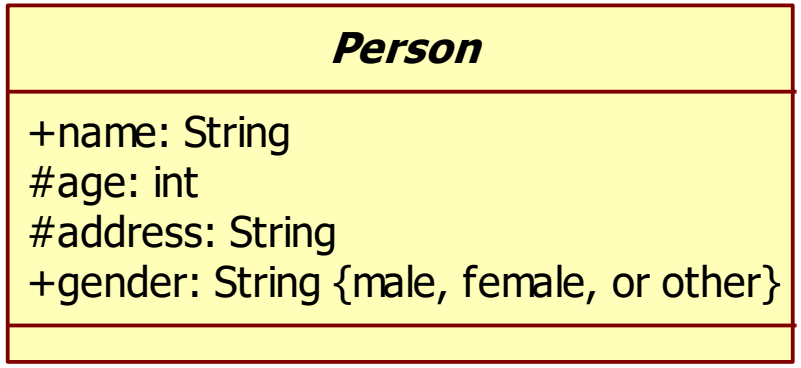
  {tagname = value, tagname = value}



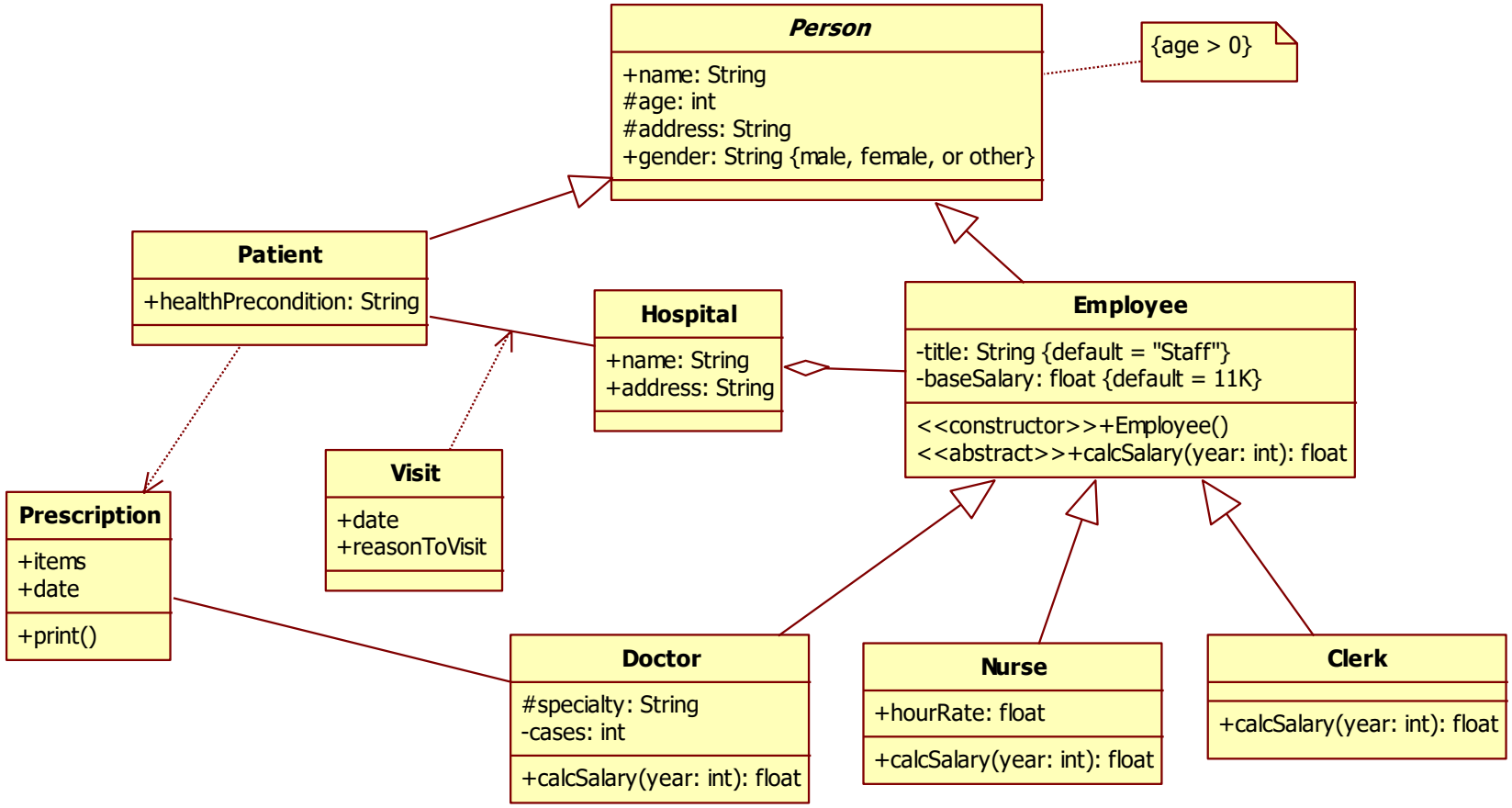
Constraints

- Allow to add new rules or modify existing ones for a UML element
- Notation: any text within curly braces {}





Putting Things Together



Some Tips

- Classes, relationships, etc. are abstractions of things (objects, links, etc.)

- To identify abstractions, specify things that users and implementers/developers use
- To identify attributes and operations
  - Identify **responsibilities** of each abstraction
  - Provide attributes and operations to perform these responsibilities

Some Tips (cont'd)

- Example, identifying attributes and operations of doctor
  - What are responsibilities of doctors?
    - Diagnose patients' problems
    - Consult patients
    - Write prescriptions
    - Get paid
    - Has title
    - Has salary
    - Has bonus
  - Specify attributes and operations based on the responsibilities identified above

Some Tips (cont'd)

- Group related classes into packages
- Show only related classes in the same diagrams
- Show only classes, operations, attributes that are important to understand
  - You can suppress operations and attributes from their classes
- Don't try to identify all possible classes, attributes, and operations at once
  - Important classes are identified first, and gradually identify other later