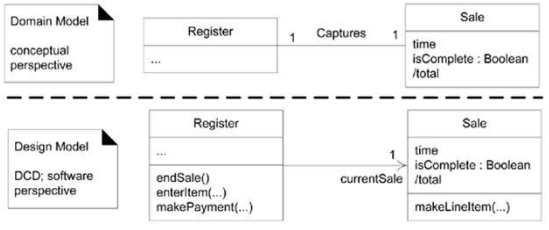
CLASS DIAGRAM

Class Diagrams

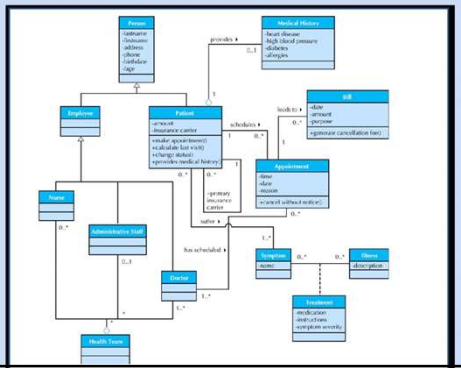
- Classes
- Relationships among classes
- Remains constant over time

From Domain Model to Design Model

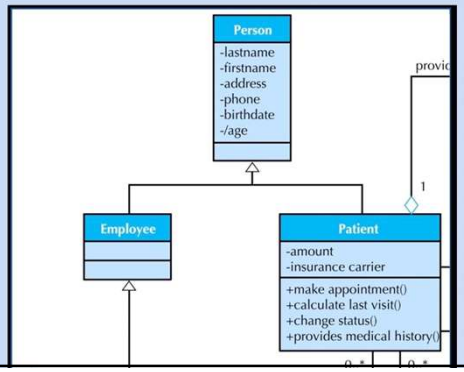
- In the same way as the domain model included conceptual class diagram, vision, use cases etc., the design model includes design class diagrams, interaction and package diagrams (etc.).
- From analysis to design we move from a conceptual view to a software view of the problem:



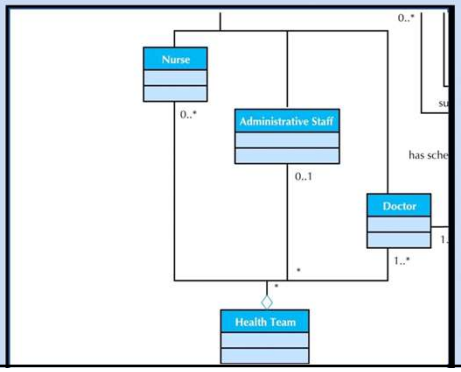
Example Class Diagram



Example Class Diagram



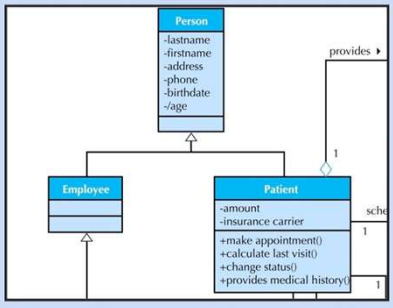
Example Class Diagram



Elements of a Class Diagram

A class: • Represents a kind of person, place, or thing about which the system will need to capture and store information. • Has a name typed in bold and centered in its top compartment. • Has a list of attributes in its middle compartment. • Has a list of operations in its bottom compartment. • Does not explicitly show operations that are available to all classes.	Class 1 -attribute1 +operation1()
An attribute: • Represents properties that describe the state of an object. • Can be derived from other attributes, shown by placing a slash before the attribute's name.	attribute name /derived attribute name
An operation: • Represents the actions or functions that a class can perform. • Can be classified as a constructor, query, or update operation. • Includes parentheses that may contain parameters or information needed to perform the operation.	operation name ()

Elements of a Class Diagram

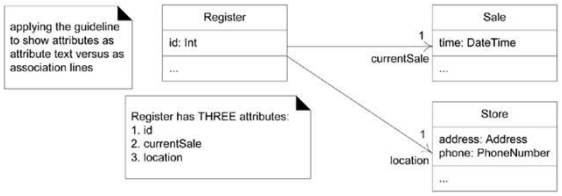


Attribute Visibility

- Attribute visibility can be specified in the class diagram
  - Public attributes (+) are visible to all classes
  - Private attributes (-) are visible only to an instance of the class in which they are defined
  - Protected attributes (#) are like private attributes, but are also visible to descendant classes

Class Attributes Visualisation

- Attributes can be shown visually : for class associations; or in text : for basic data types only;

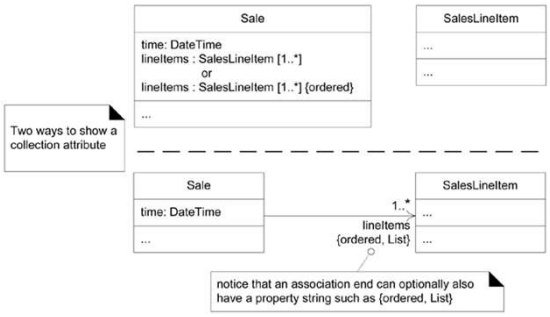


- Semantically, showing attributes as text or as visual association is equivalent: the code will be the same e.g. for the above:

```
public class Register {
    private int id;
    private Sale currentSale;
    private Store location;
// … }
```

- In a design class diagram the associations have the following properties (using the previous diagram as example):
  - a navigability arrow pointing from the source (Register) to target (Sale) object, indicating a Register object has an attribute of one Sale;
  - a multiplicity at the target end, but not the source end;
  - a rolename (currentSale) only at the target end to show the attribute name;
  - no association name;
- As mentioned, association ends can have a role name, show a multiplicity value but also have a keyword:
  - Examples of keywords include *{ordered}* to indicate that the elements of a collection are to be kept in some kind of order by the data structure that will hold the many target objects associated with the source object.

- We can even specify what data structure should be used to hold the many objects (e.g. a List): use the user-defined keyword *{List}*:



- We should prefer the visual version rather than the text version …

Operations

- **Constructor**
  - Creates object
- **Query**
  - Makes information about state available
- **Update**
  - Changes values of some or all attributes

Operations

- **The operations that are available to all classes** (e.g., create a new instance, return attribute value, set attribute value, delete an instance) **are not explicitly shown**.

- Only those operations that are unique to the class are included.

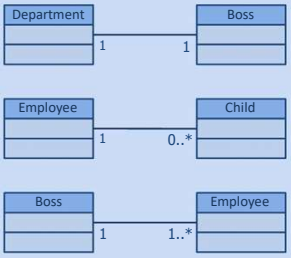
Relationships

- A primary purpose of class diagrams is to show relationships, or ***associations***, between classes

More Elements of Class Diagrams

An association: • Represents a relationship between multiple classes or a class and itself. • Is labeled using a verb phrase or a role name, whichever better represents the relationship. • Can exist between one or more classes. • Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance.	AssociatedWith 0..* 1
A generalization: • Represents a-kind-of relationship between multiple classes.	
An aggregation: • Represents a logical a-part-of relationship between multiple classes or a class and itself. • Is a special form of an association.	0..* IsPartOf 1
A composition: • Represents a physical a-part-of relationship between multiple classes or a class and itself • Is a special form of an association.	1..* IsPartOf 1

Multiplicities

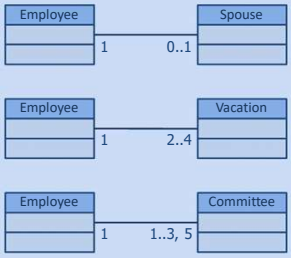


Exactly one:
A department has one and only one boss

Zero or more:
An employee has zero to many children

One or more:
A boss is responsible for one or more employees

More Multiplicities



Zero or one:
An employee can be married to 0 or 1 spouse

Specified range:
An employee can take 2 to 4 vacations each year

Multiple disjoint ranges:
An employee can be in 1 to 3 or 5 committees

Simplifying Class Diagrams

- **Use View Mechanism to shows a subset of classes**
  - Only classes & relationship relevant to the use-case
  - Shows particular type of relationships
  - Restrict the information shown in each class
- **Packages show aggregations of classes** (or any elements in UML)