Structural Modeling

Objectives

- Understand the rules and style guidelines for creating CRC cards, class diagrams, and object diagrams.
- Understand the processes used to create CRC cards, class diagrams, and object diagrams.
- Be able to create CRC cards, class diagrams, and object diagrams.
- Understand the relationship between the structural and use-case models.

Structural Model

- *Structural Model*
  - is a formal way of representing the objects that are used and created by a business system.
  - It illustrates people, places, or things about which information is captured, and how they are related to each other.
  - The structural model is drawn using an iterative process in which the model becomes more detailed and less "conceptual" over time.

Structural Model

- One of the primary purposes of the structural model is to create a vocabulary that can be used by both the analyst and users.
- Structural models represent the things, ideas, or concepts, that is, the objects, contained in the domain of the problem.
- They also allow the representation of the relationships between the things, ideas, or concepts.

Structural Model

- Structural model does not represent software components or classes in an object-oriented programming language, even though the structural model does contain analysis classes, attributes, operations, and relationships among the analysis classes.
- Structural model should represent the responsibilities of each class and the collaborations between the classes.
- Typically, structural models are depicted using CRC cards, class diagrams, and, in some cases, object diagrams.

Conceptual Model

- *Conceptual Model*
  - Shows the logical organization of the objects without indicating how the objects are stored, created, or manipulated.
  - Free from any implementation or technical details,
  - Analysts can focus more easily on matching the model to the real business requirements of the system.

Design Model

- Design Model
  - reflects how the objects will be organized in databases and files.
  - The model is checked for redundancy and the analysts investigate ways to make the objects easy to retrieve.

Class Relationships

- Relationships Three Classification
  - *Generalization relationships,*
  - *Aggregation relationships,*
  - *Association relationships.*

Classification of Relationship

- **Generalization Relationships**
  - Enables the analyst to create classes that inherit attributes and operation of other classes.
  - Represented with the a kind of relationship.

- **Aggregation Relationships**
  - Relates parts to wholes or parts to assemblies
- **Association Relationship**
  - Association is a relationship where all object have their own lifecycle and there is no owner

Class-Responsibility-Collaborator (CRC)

- CRC modeling provides a simple means for identifying and organizing the classes that are relevant to system or product requirements.
- A CRC model is a collection of standard index cards that represents classes.
- Used to document the responsibilities and collaboration of a class.
- The card are divided into three sections
  - Top- class name
  - Middle- class responsibilities on the left and collaboration on the right

Example

Class name	
Sub classes	
Superclasses	
Responsibilities	Collaborator

Types of Responsibilities

- Knowing Responsibility
  - Those things that an instance of a class must be capable of knowing
  - Instance of the class typically knows the values of attributes and its relationships
- Doing Responsibility
  - Those things that an instance of a class must be capable of doing

Collaboration

- Allow the analyst to think in terms of client, servers and contract
  - Client object- is an instance of a class that sends a request to an instance of another class for an operation to be executed.
  - A server object is the instance that receives the request from the client object
  - A contract formalizes the interactions between the client and server objects.

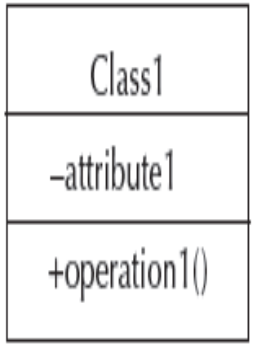
Example

Class name	Patient
Description	An individual that needs to receive or has received medical attention
Responsibilities	**Collaborator**
Make Appointment	Appointment
Calculate last visit	
Change status Provide medical history	Medical History

Class Diagram

- The *class diagram* is a *static model* that shows the classes and the relationships among classes that remain constant in the system over time.
- The class diagram depicts classes, which include both behaviors and states, with the relationships between the classes.

Elements of Class Diagram

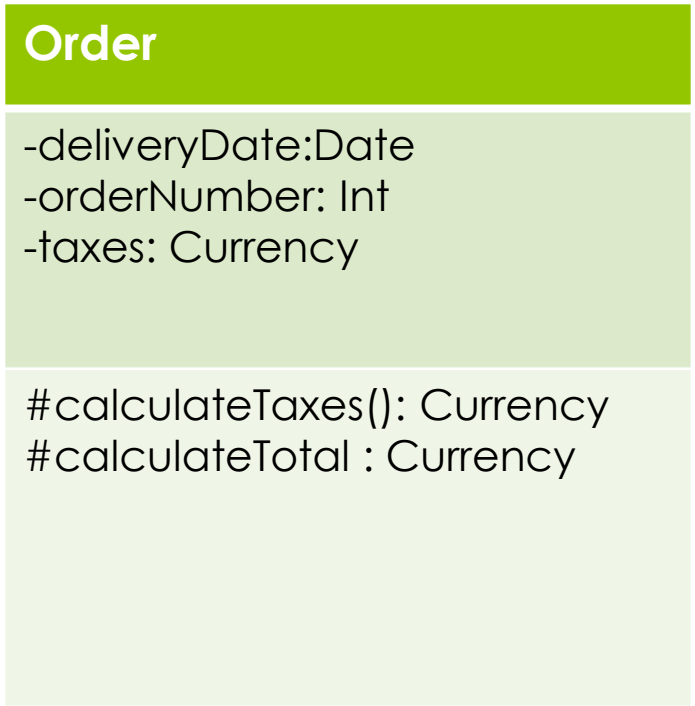
A CLASS • Represents a kind of person, place, or thing about which the system will need to capture and store information • Has a name typed in bold and centered in its top compartment • Has a list of attributes in its middle compartment • Has a list of operations in its bottom compartment • Does not explicitly show operations that are available to all classes	
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

Elements of Class Diagram

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 ()
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	1..* verb phrase 0..1

Elements of a class

- Attribute
  - A one word attribute name is written in lowercase letter. If the name consist of more than one word, the words are joined and each word other than the first word begins with ban uppercase letter
  - In the class icon, you can specify a type for each attribute's value.
  - Also you can specify the default value for attribute



Visibility

- *Visibility*
  - relates to the level of information hiding to be enforced for the attribute.
  - The visibility of an attribute can be public (+), protected (#), or private (–).
- A *public* attribute is one that is not hidden from any other.
- A *protected* attribute is one that is hidden from all other classes except its immediate subclasses.
- A *private* attribute is one that is hidden from all other classes. The default visibility for an attribute is normally private.

Operations

- *Operations* are actions or functions that a class can perform
- A class can contain three kinds of operations: constructor, query, and update.
  - A *constructor operation* creates a new instance of a class. For example, the patient class may have a method called "insert ()" that creates a new patient instance as patients are entered into the system.

Operations

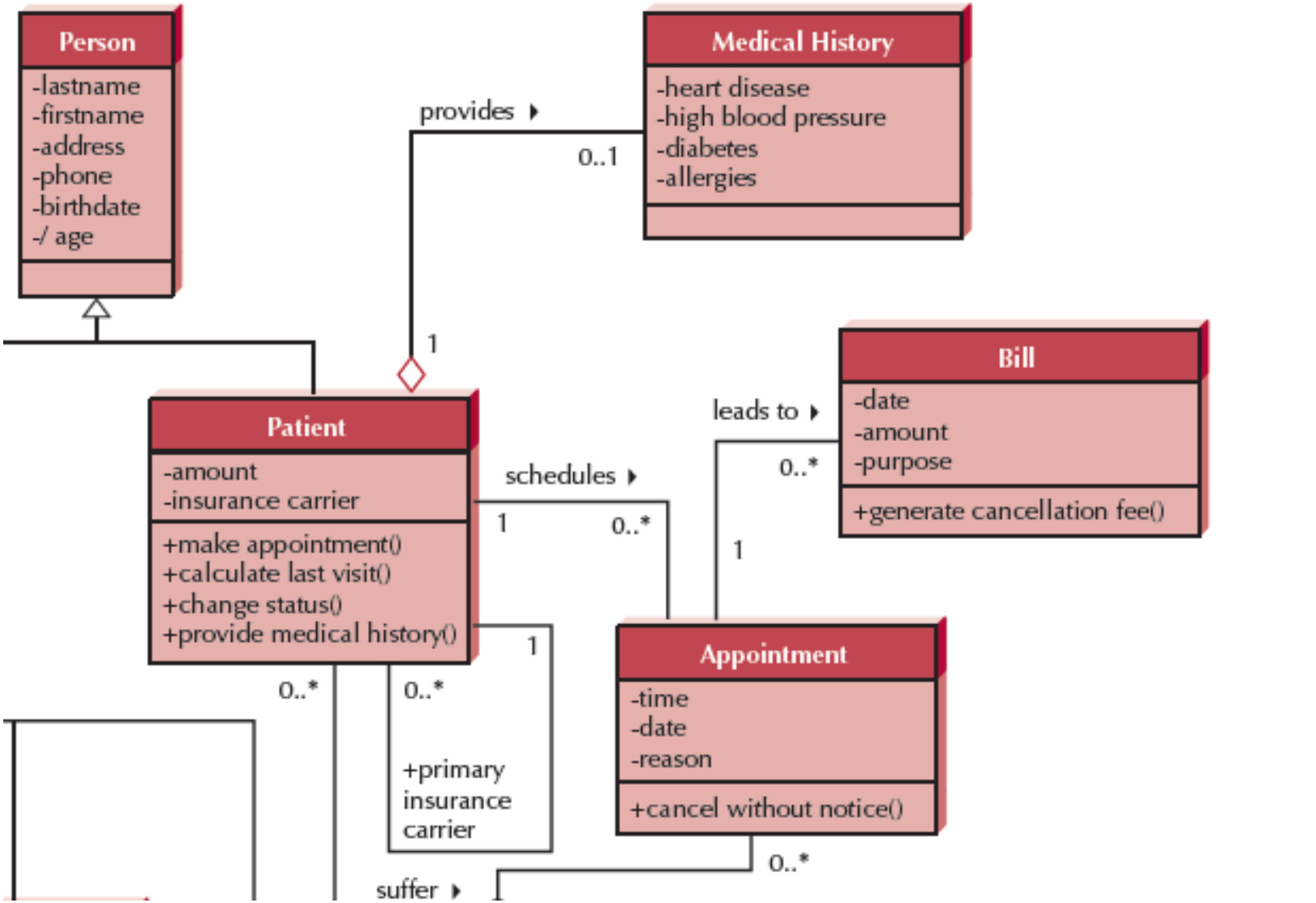
- A *query operation* makes information about the state of an object available to other objects, but it will not alter the object in any way. For instance, the "calculate last visit ()" operation that determines when a patient last visited the doctor's office will result in the object being accessed by the system, but it will not make any change to its information

Operations

- An *update operation* will change the value of some or all of the object's attributes, which may result in a change in the object's state. Consider changing the status of a patient from new to current with a method called "change status ()", or associating a patient with a particular appointment with "schedule appointment (appointment)."

Relationships

- A primary purpose of the class diagram is to show the relationships, or *associations,* that classes have with one another.
- These are depicted on the diagram by drawing lines between classes.
- When multiple classes share a relationship (or a class shares a relationship with itself), a line is drawn and labeled with either the name of the relationship or the roles that the classes play in the relationship.

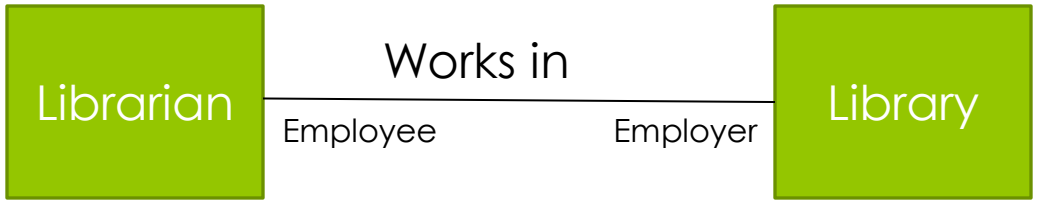

Person
-lastname
-firstname
-address
-phone
-birthdate
-/ age
provides
0..1
Medical History
-heart disease
-high blood pressure
-diabetes
-allergies
1
Bill
-date
-amount
-purpose
+generate cancellation fee()
leads to
0..*
1
Patient
-amount
-insurance carrier
+make appointment()
+calculate last visit()
+change status()
+provide medical history()
schedules
1
0..*
1
0..*
0..*
+primary
insurance
carrier
Appointment
-time
-date
-reason
+cancel without notice()
0..*
suffer

Association

- Represents a relationship between classes or a class and itself.
- Is labeled using a verb phrase or a role name whichever represents the relationship
- Contains multiplicity symbols which represents the minimum and maximum times a class instance can be associated with the related class instance

Elements of a class diagrams

- Relationships
  - An association



When a class associates with another, each one usually plays a role within that association. You can show those roles on the diagram by writing them near the line next to the class that plays the role.

Multiplicity

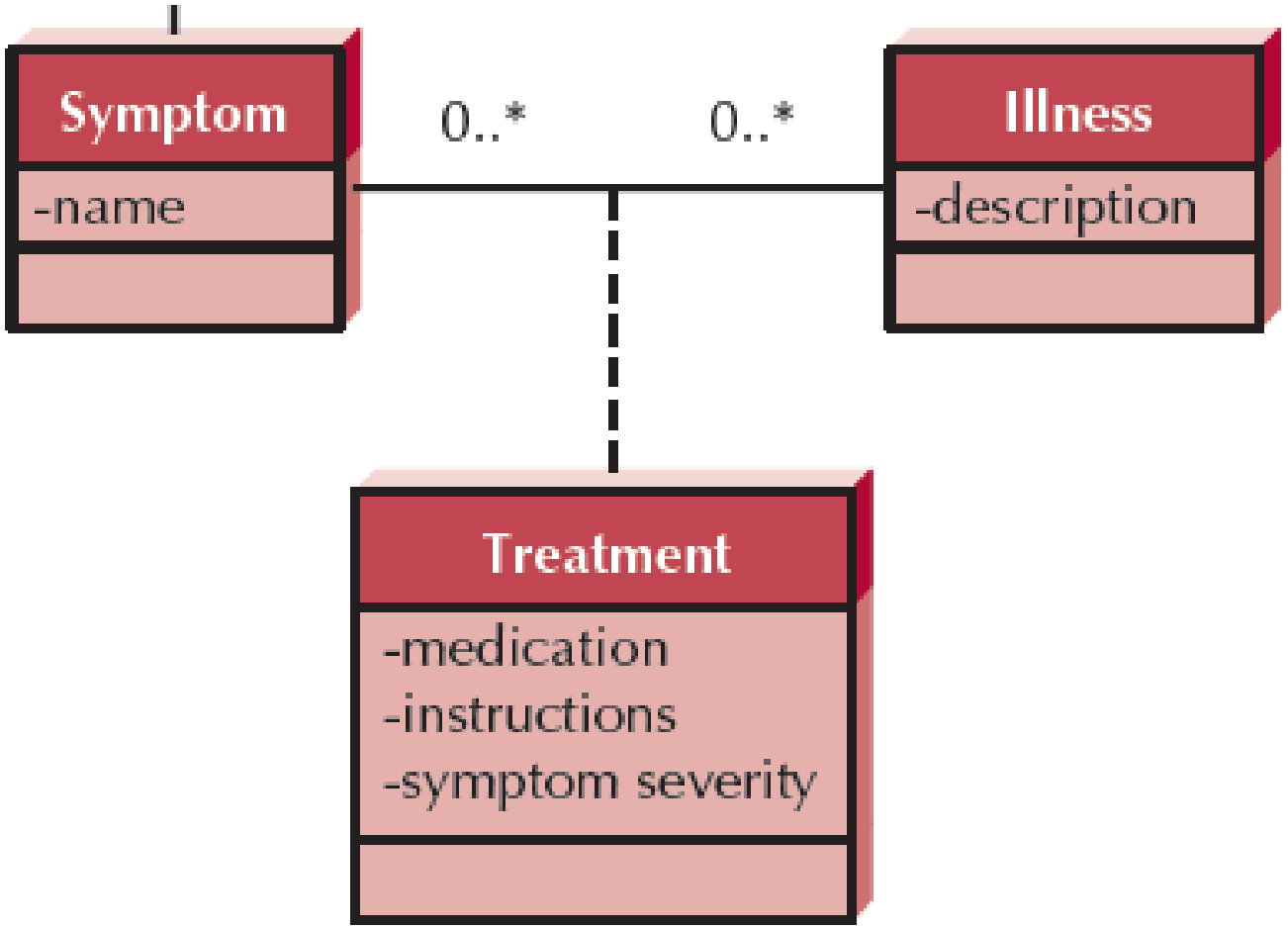
- Documents how an instance of an object can be associated with other instances

Multiplicity	Meaning
1	Exactly 1 or one only
0..*	Zero to more
1..*	One to more
0..1	Zero or one
0..n	Zero to n(n>1)
1..n	One to n(n>1)
2..4 1..3,5	Specified range Multiple, disjoint range

Department	1	Boss	A department has one and only one boss.
Employee	0..*	Child	An employee has zero to many children.
Boss	1..*	Employee	A boss is responsible for one or more employees.
Employee	0..1	Spouse	An employee can be married to zero or no spouse.
Employee	2..4	Vacation	An employee can take between two to four vacations each year.
Employee	1..3, 5	Committee	An employee is a member of one to three or five committees.

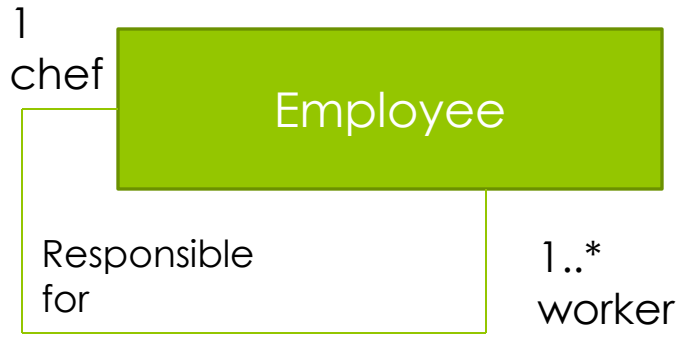
Elements of a Class Diagram

- Association class
  - There are times when a relationship itself has associated properties, especially when its classes share a many-to-many relationship.
  - It has its own attributes and operations.
  - It is shown as a rectangle attached by a dashed line to the association path, and the rectangle's name matches the label of the association.


Symptom
-name
0..*
0..*
Illness
-description
Treatment
-medication
-instructions
-symptom severity

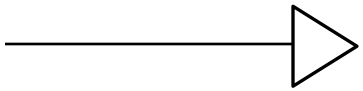
Elements of a Class Diagram

- Relationships
  - An Association
    - A class is an association with itself
    - This can happen when a class has objects that can play a variety of roles.
    - These association is called **Reflexive Associations.**



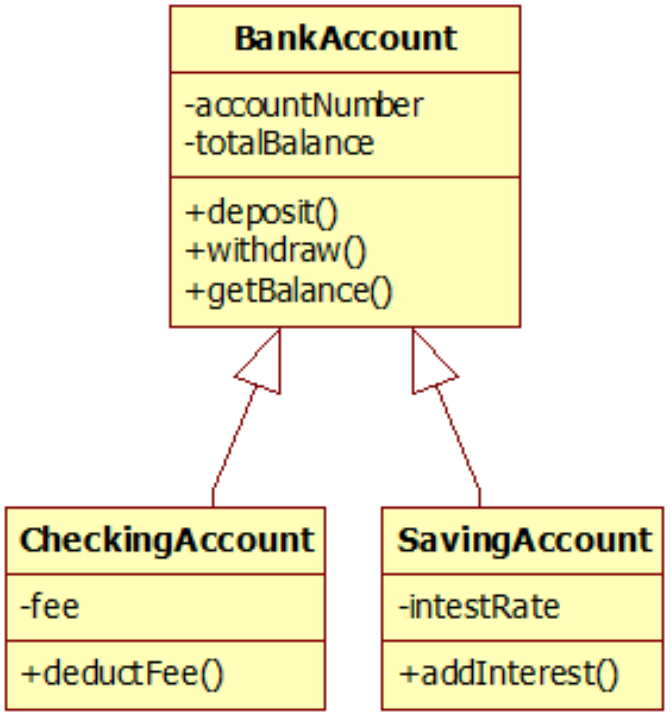
Elements of a Class

- Generalization shows that one class (subclass) inherits from another class (superclass), meaning that the properties and operations of the superclass are also valid for objects of the subclass.
- Represent "is a-kind-of" relationship between multiple classes.



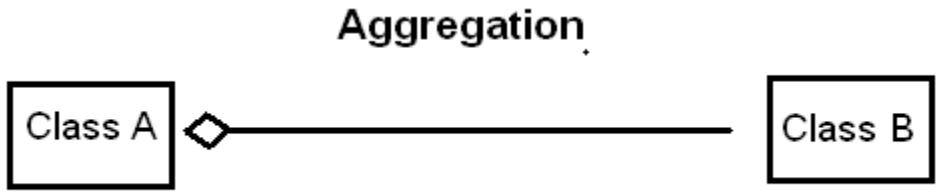
Elements of a Class

- In UML inheritance is represented with a line that connects the parent to a child class, and on the parents side you put an open triangle.



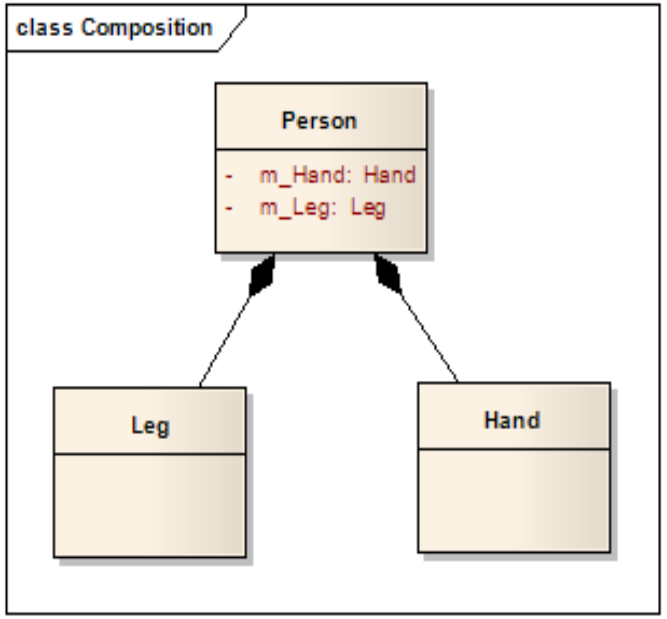
Aggregation

- Aggregation is used when classes actually comprise other classes.
- A diamond is placed nearest the class representing the aggregation
- Means “a part of, a member of, contained in, related to and associated with”



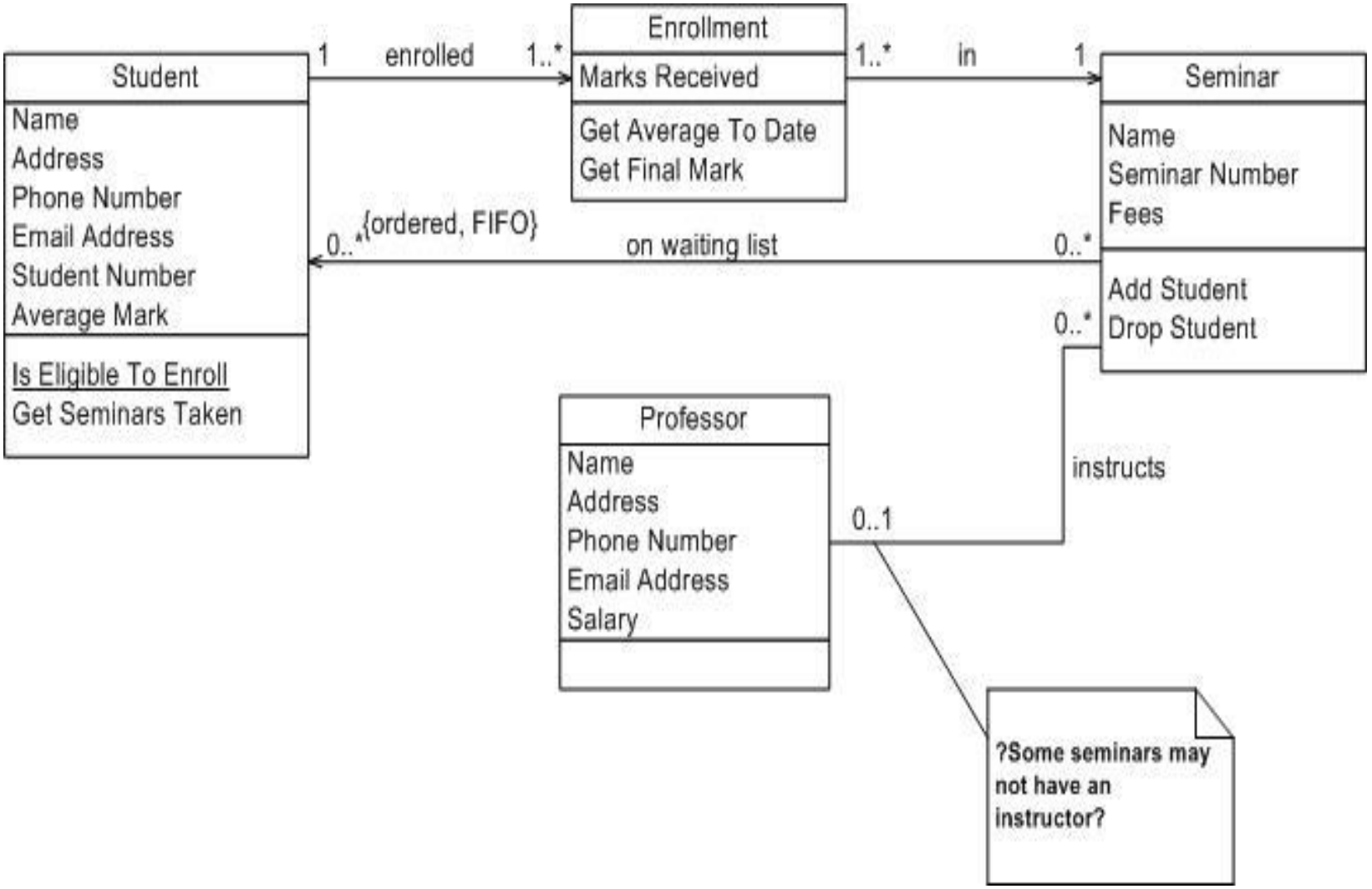
Composition

- A strong type of aggregation
- Each component in a composite can belong to just one whole. The symbol for composite is the same as the symbol of aggregation except the diamond is filled.
- Represents a *physical a part* of relationship between classes



Simplifying Class Diagram

- Show only concrete classes
- The view mechanism shows a subset of classes
- Package show aggregations of classes (or any elements in UML)


Student
Name
Address
Phone Number
Email Address
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
1
enrolled
1..*
Enrollment
Marks Received
Get Average To Date
Get Final Mark
1..*
in
1
Seminar
Name
Seminar Number
Fees
Add Student
Drop Student
0..*
{ordered, FIFO}
on waiting list
0..*
0..*
instructs
0..1
Professor
Name
Address
Phone Number
Email Address
Salary
?Some seminars may not have an instructor?

Clarifications?