Chapter 14: Object-Oriented Data Modeling

Modern Database Management

6th Edition

Jeffrey A. Hoffer, Mary B. Prescott, Fred R. McFadden

What is Object Oriented Data Modeling?

Centers around objects and classes

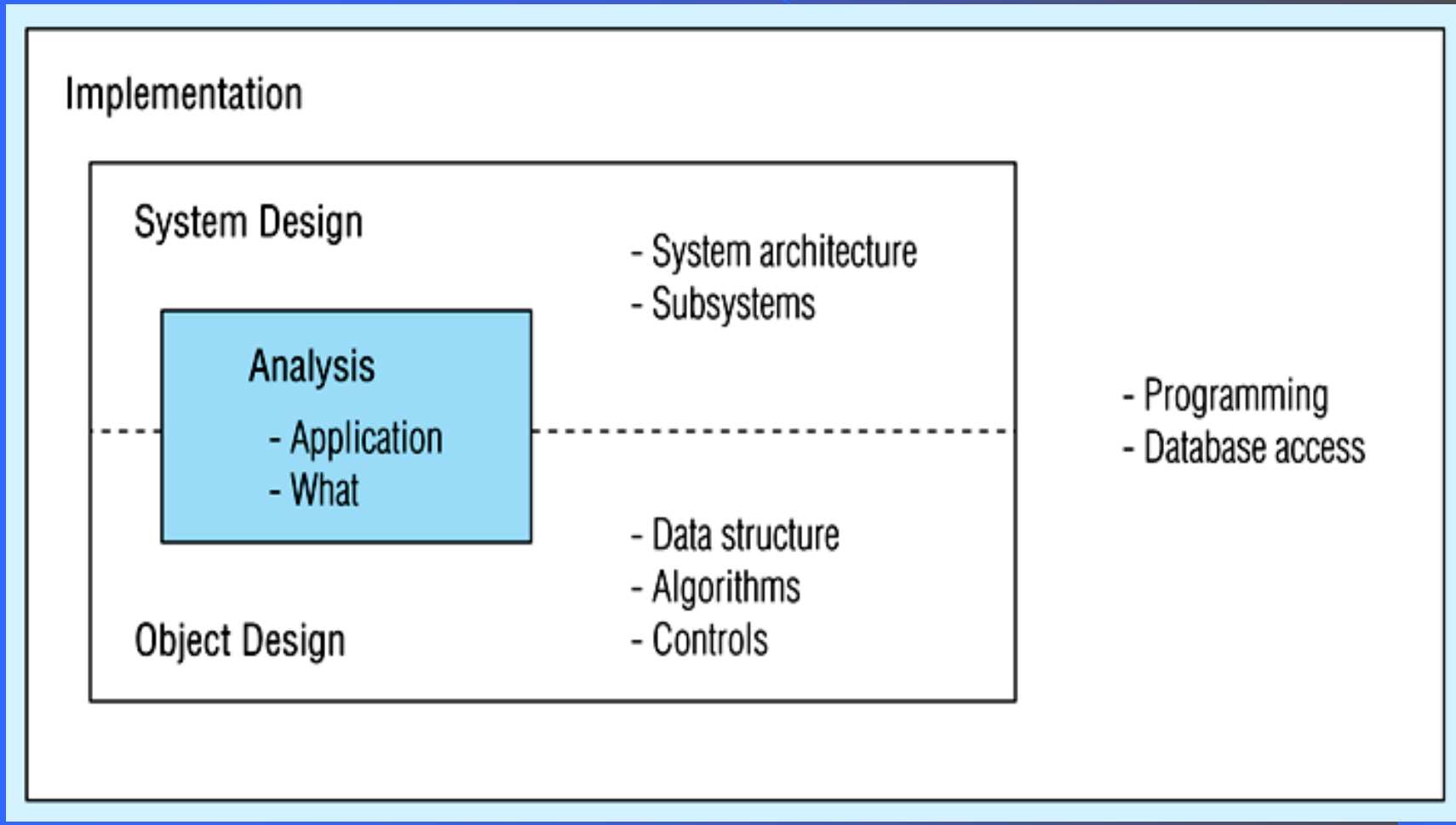
Involves inheritance

Encapsulates both data and behavior

Benefits of Object-Oriented Modeling

- Ability to tackle challenging problems
- Improved communication between users, analysts, designer, and programmers
- Increased consistency in analysis and design
- Explicit representation of commonality among system components
- System robustness
- Reusability of analysis, design, and programming results

Figure 14-1 – Phases of object-oriented systems development cycle



OO vs. EER Data Modeling

Object Oriented

Class
Object
Association
Inheritance of attributes
Inheritance of behavior

EER

Entity type
Entity instance
Relationship
Inheritance of attributes
No representation of behavior

Object-oriented modeling is frequently accomplished using the **Unified Modeling Language (UML)**

Object

An entity that has a well-defined role in the application domain, as well as state, behavior, and identity

- Tangible: person, place or thing
- Concept or Event: department, performance, marriage, registration
- Artifact of the Design Process: user interface, controller, scheduler

Objects exhibit BEHAVIOR as well as attributes

➔ Different from **entities**

State, Behavior, Identity

State: attribute types and values

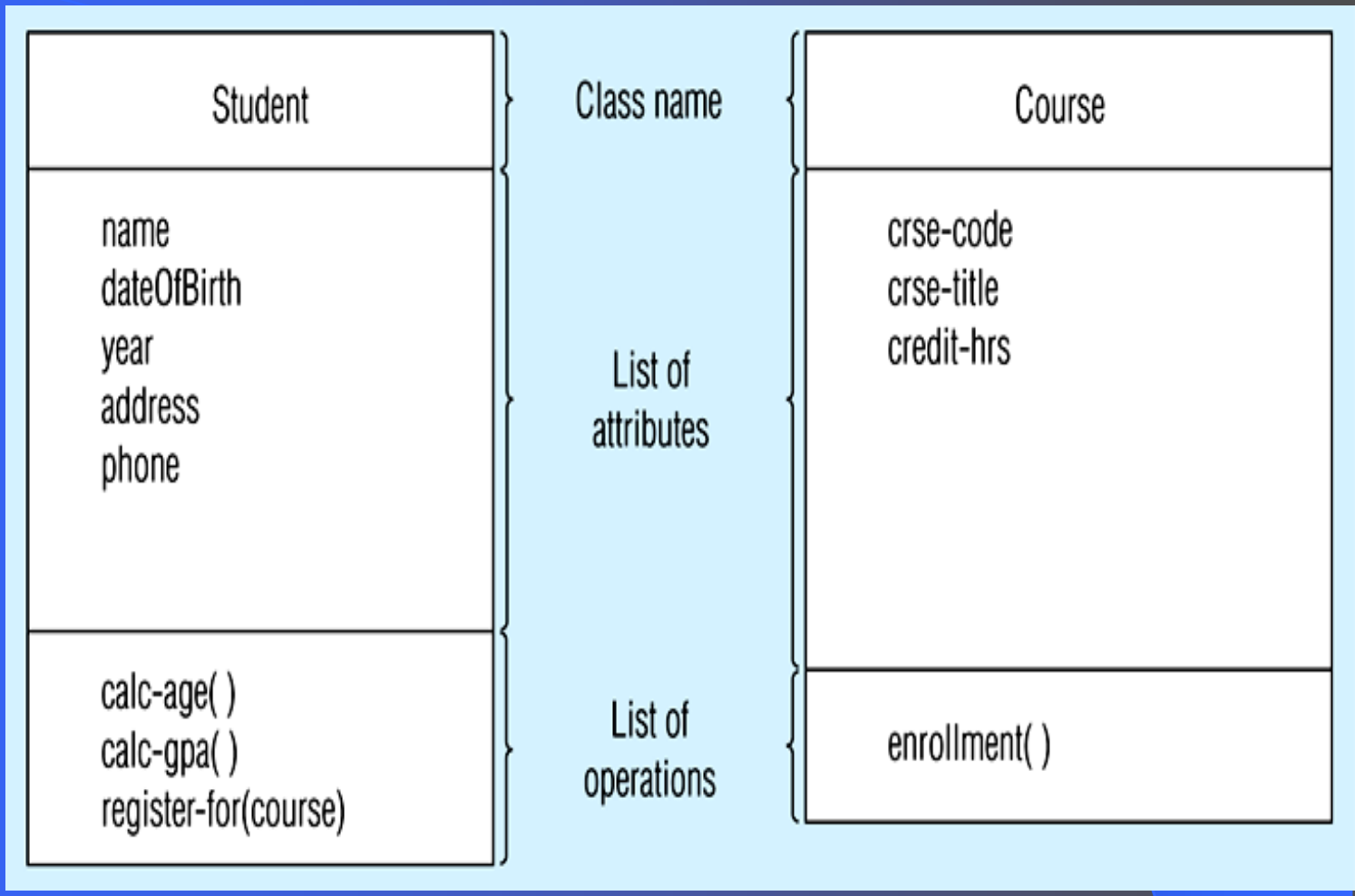
Behavior: how an object acts and reacts

- Behavior is expressed through operations that can be performed on it

Identity: every object has a unique identity, even if all of its attribute values are the same

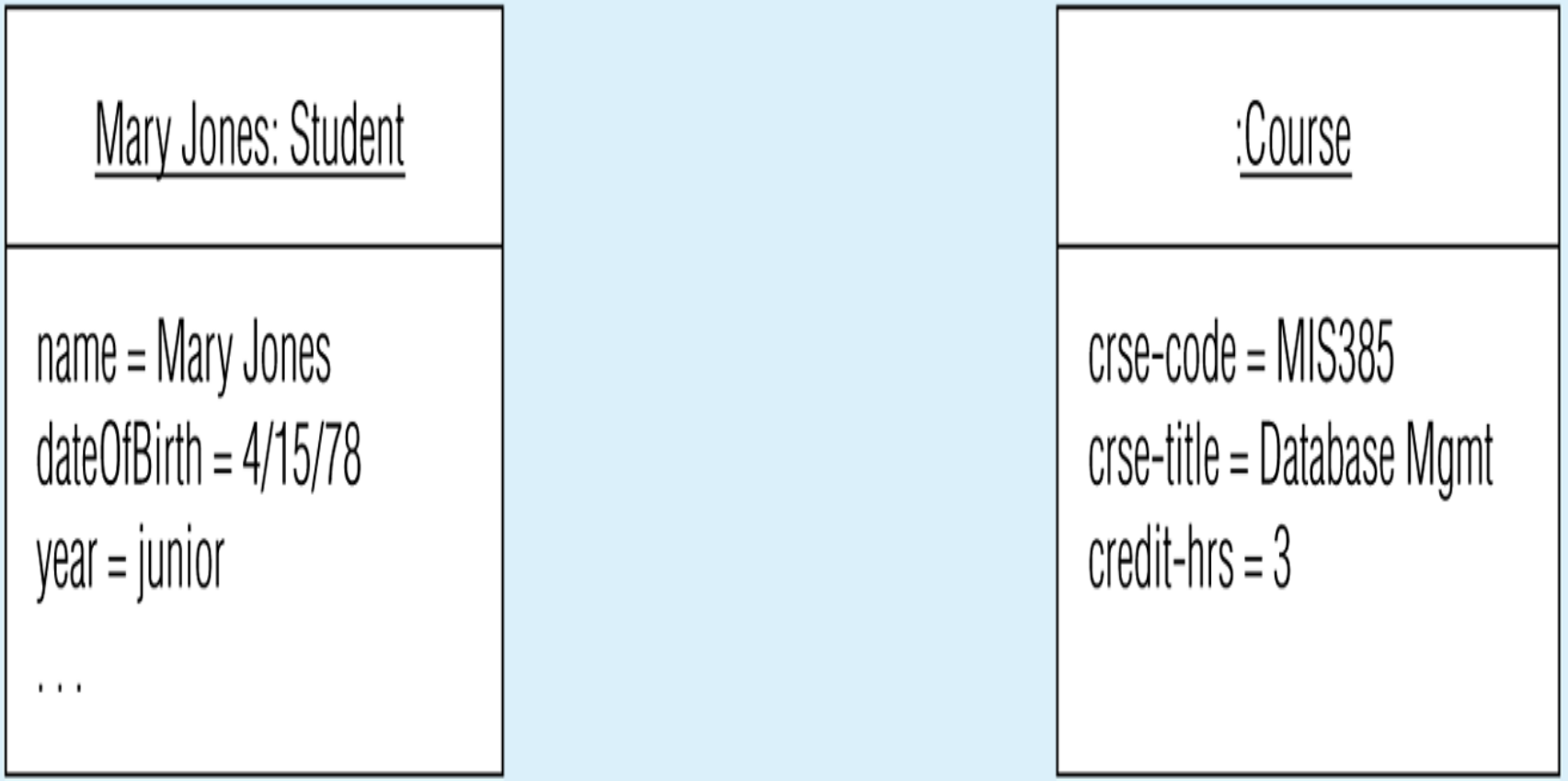
Figure 14-2 – UML class and object diagrams

(a) Class diagram showing two classes



Class diagram shows the static structure of an object-oriented model: object classes, internal structure, relationships.

(b) Object diagram with two instances



Object diagram shows instances that are compatible with a given class diagram.

Operations

A function or service that is provided by all instances of a class

Types of operators:

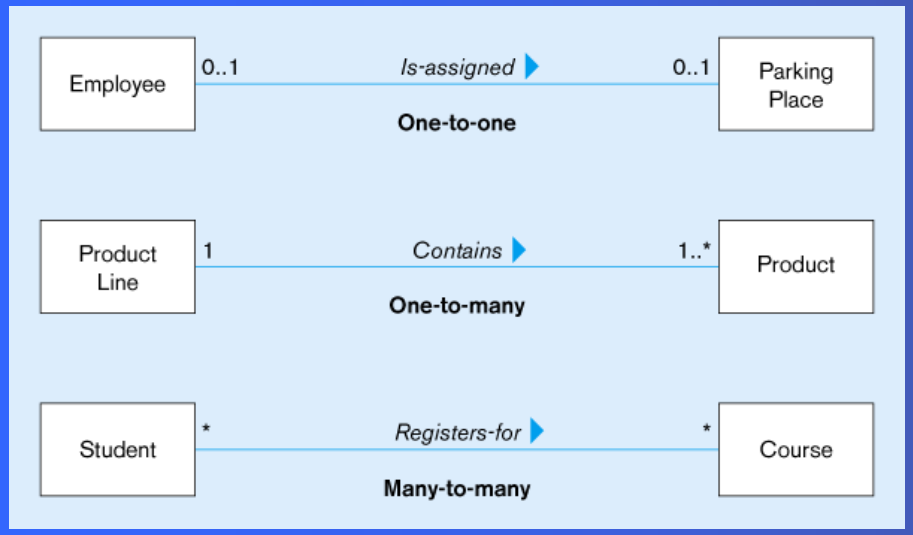
- **Constructor**: creates a new instance of a class
- **Query**: accesses the state of an object but does not alter its state
- **Update**: alters the state of an object
- **Scope**: operation applying to the class instead of an instance

Operations implement the object's **behavior**

Associations

- **Association**:
  - Relationship among object classes
- **Association Role**:
  - Role of an object in an association
  - The end of an association where it connects to a class
- **Multiplicity**:
  - How many objects participate in an association. Lower-bound..Upper bound (cardinality).

Figure 14-3 – Association relationships of different degrees



Lower-bound – upper-bound

Represented as:

0..1, 0..*, 1..1, 1..*

Similar to minimum/maximum cardinality rules in EER

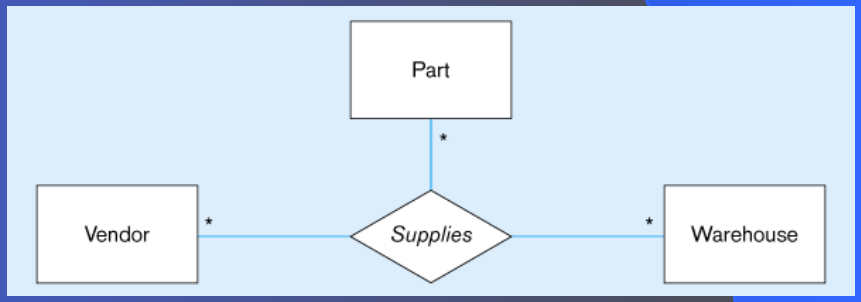
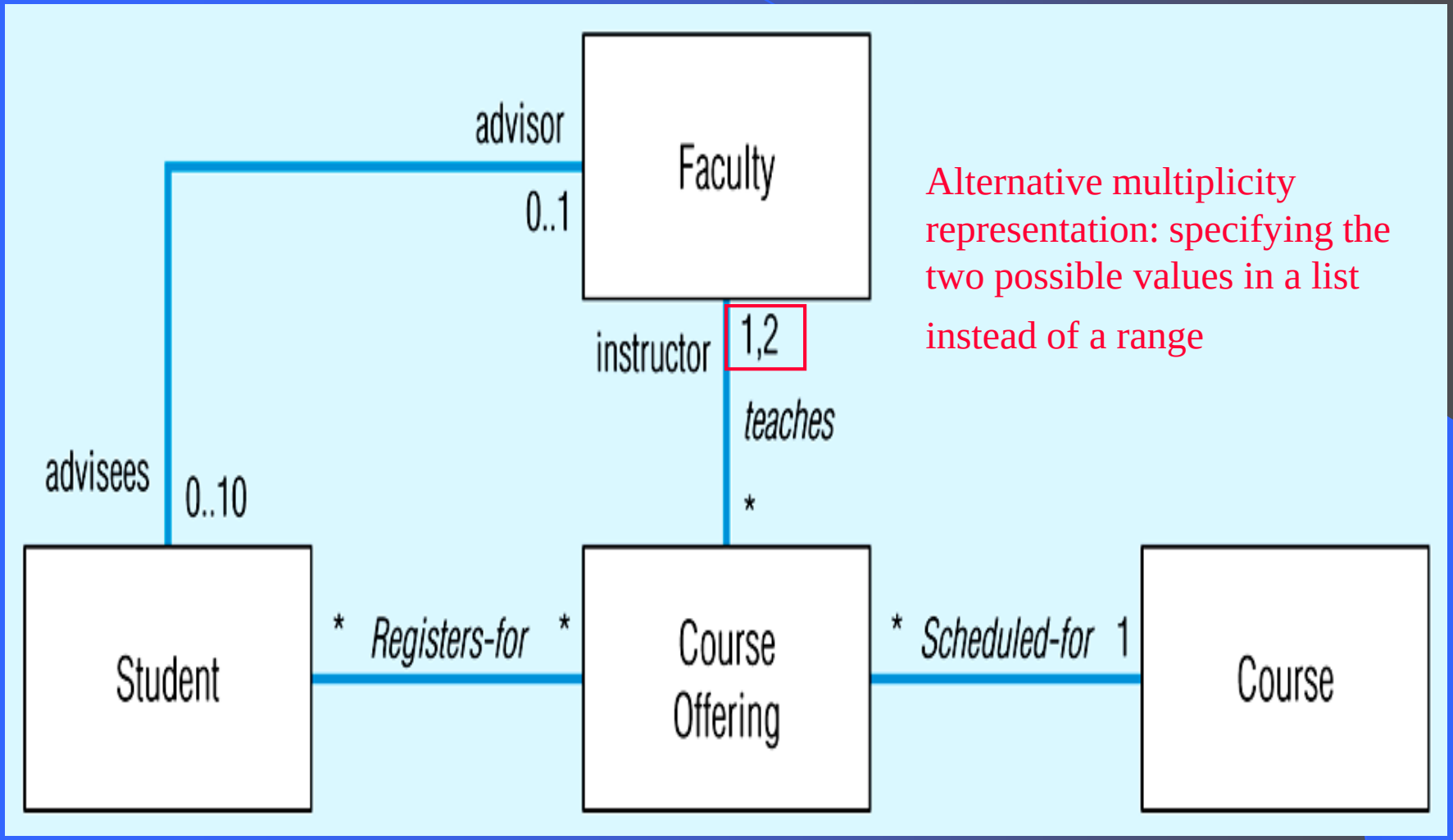


Figure 14-4 – Examples of binary association relationships (a) University example



(b) Customer order example

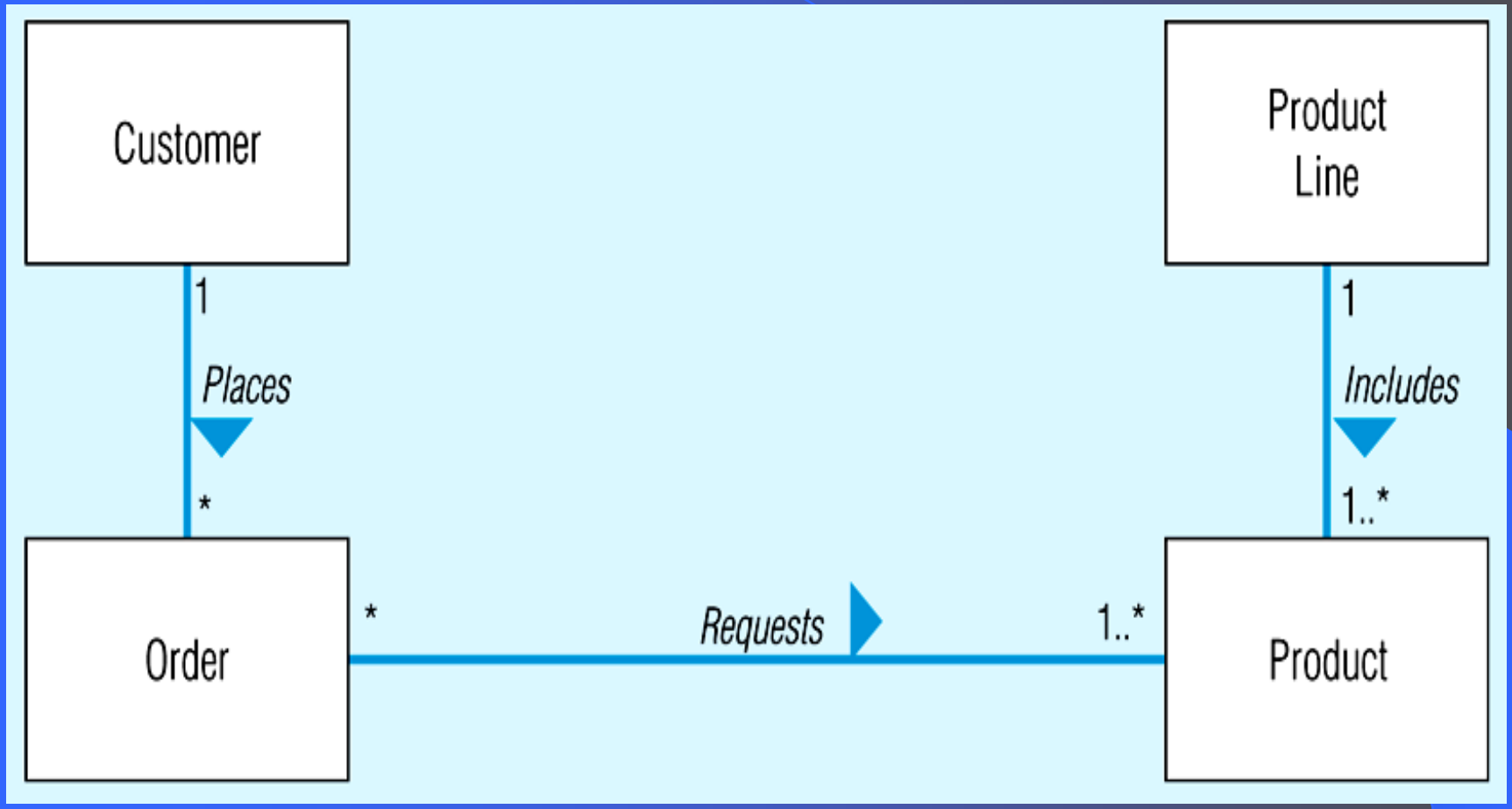
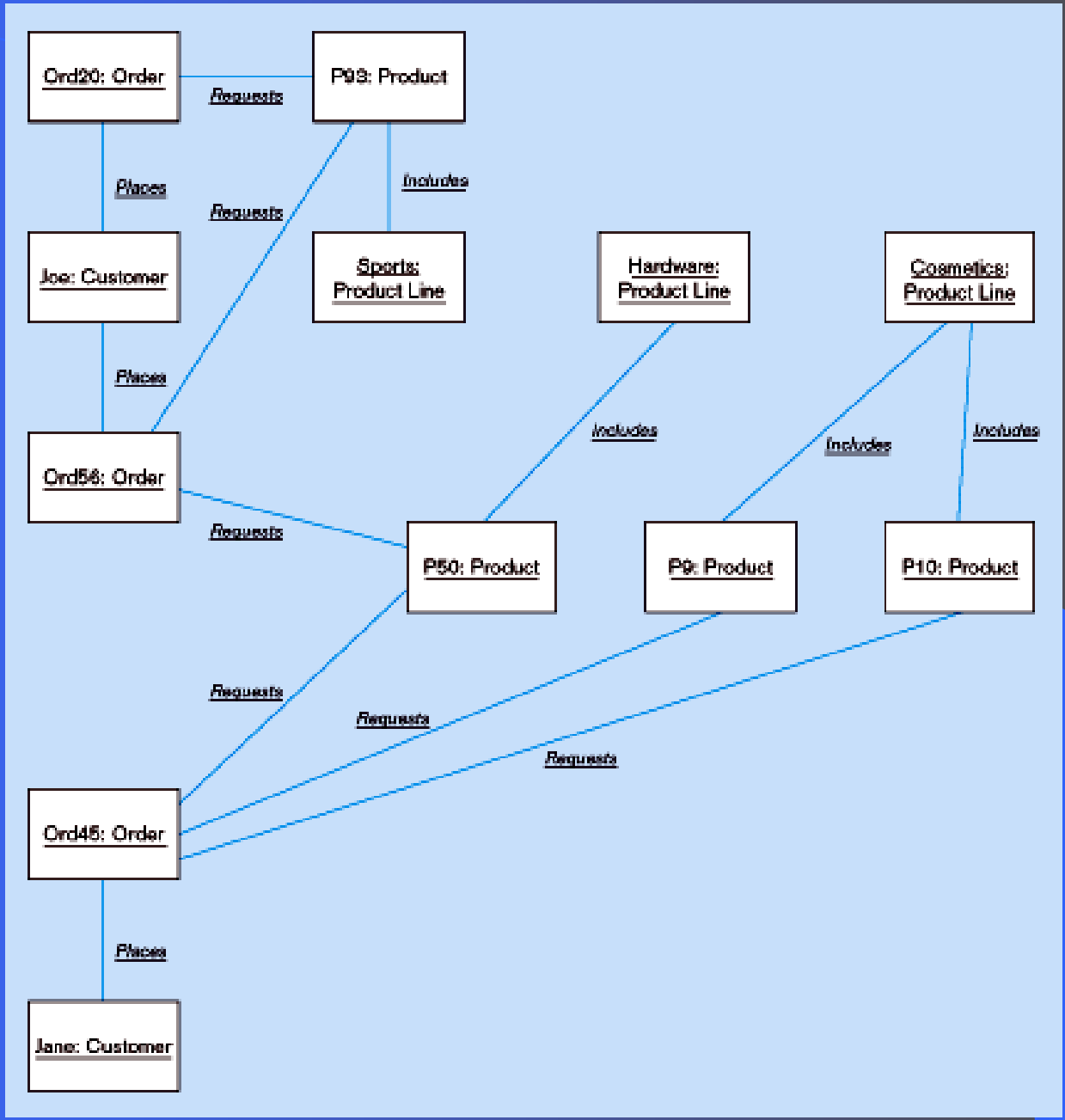


Figure 14-5 – Object diagram for customer order example



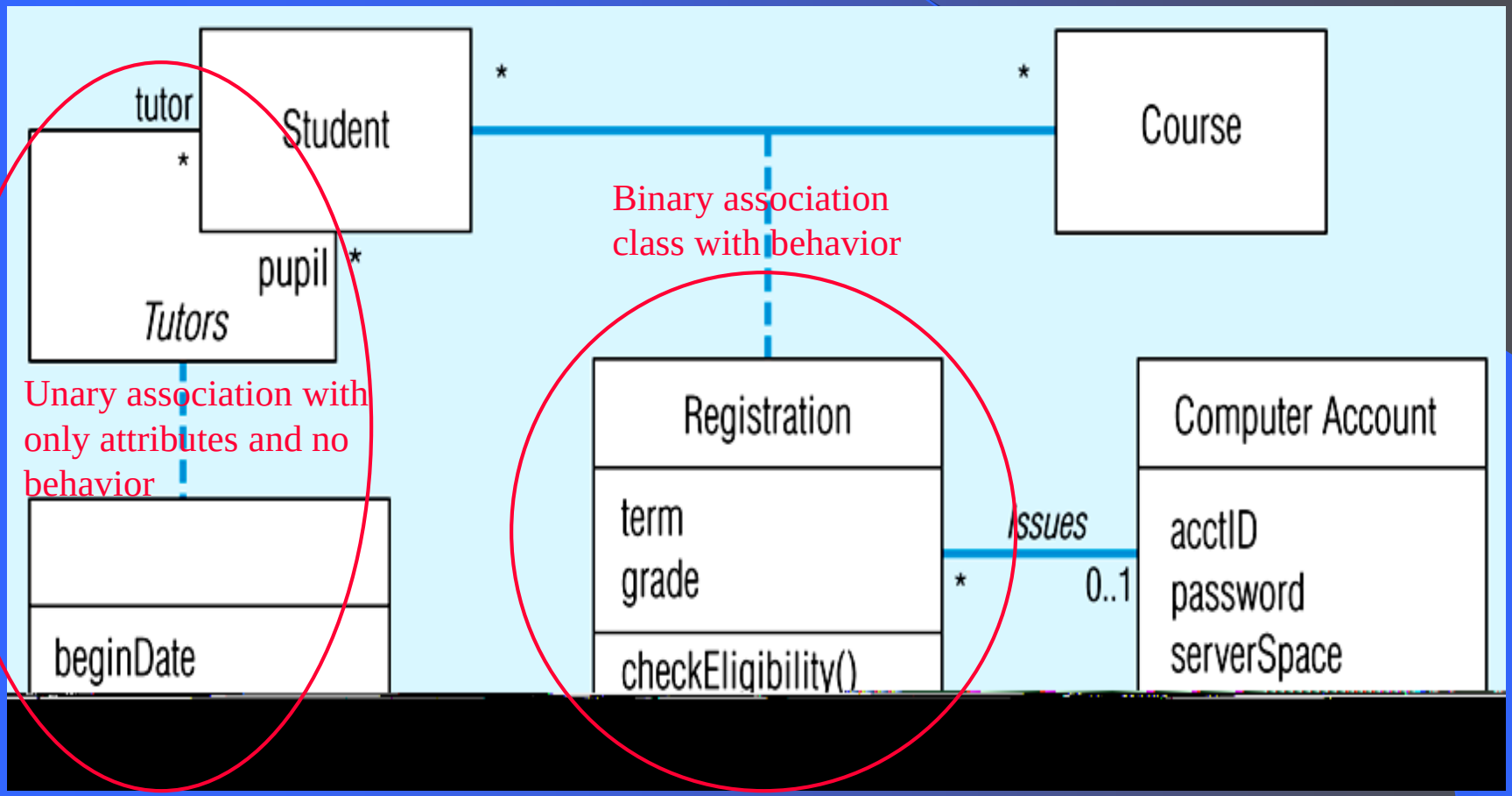
Association Class

An association that has attributes or operations of its own or that participates in relationships with other classes

Like an associative entity in ER model

Figure 14-6 – Association class and link object

(a) Class diagram showing association classes



(b) Object diagram showing link objects

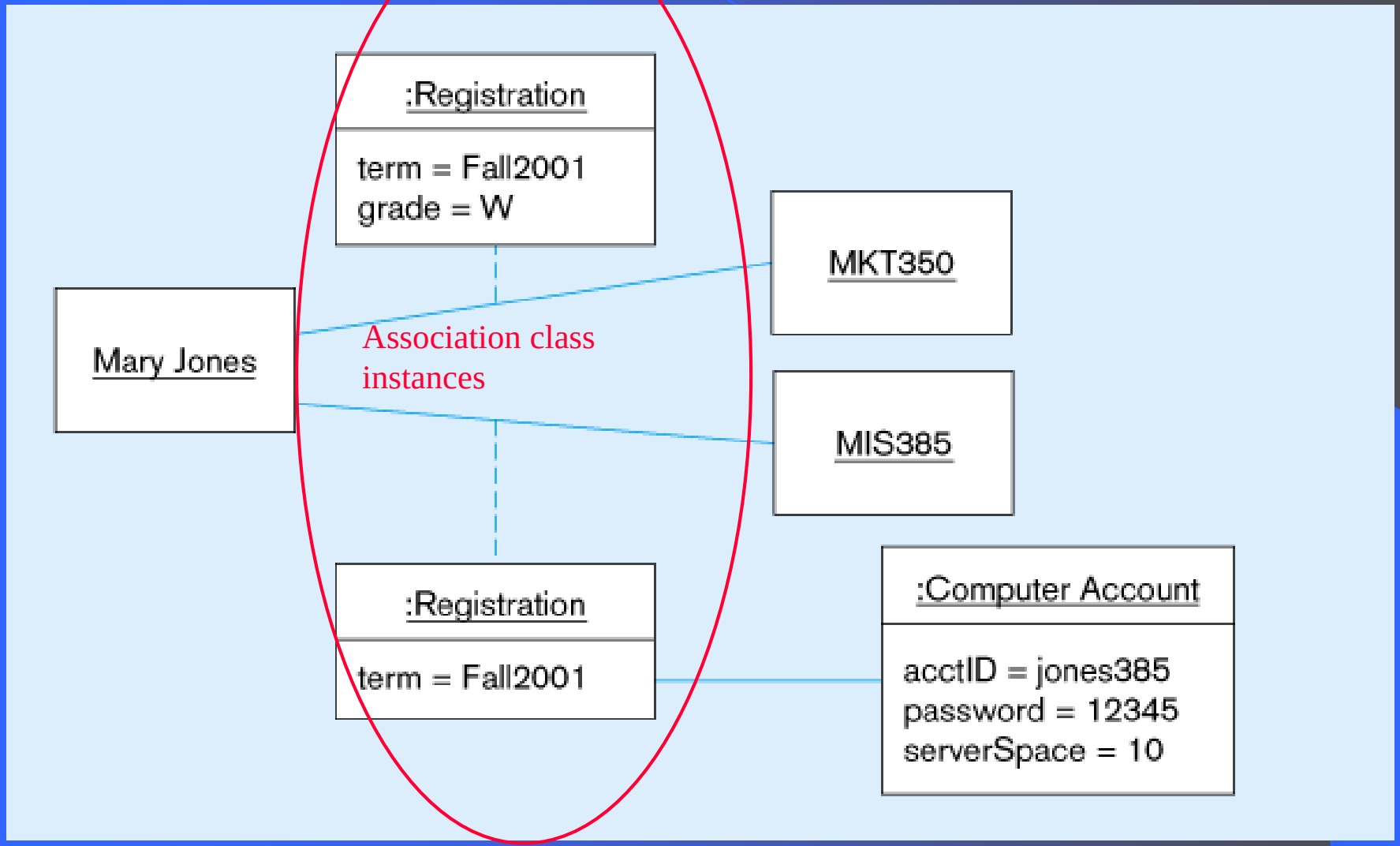


Figure 14-7 –Ternary relationship with association class

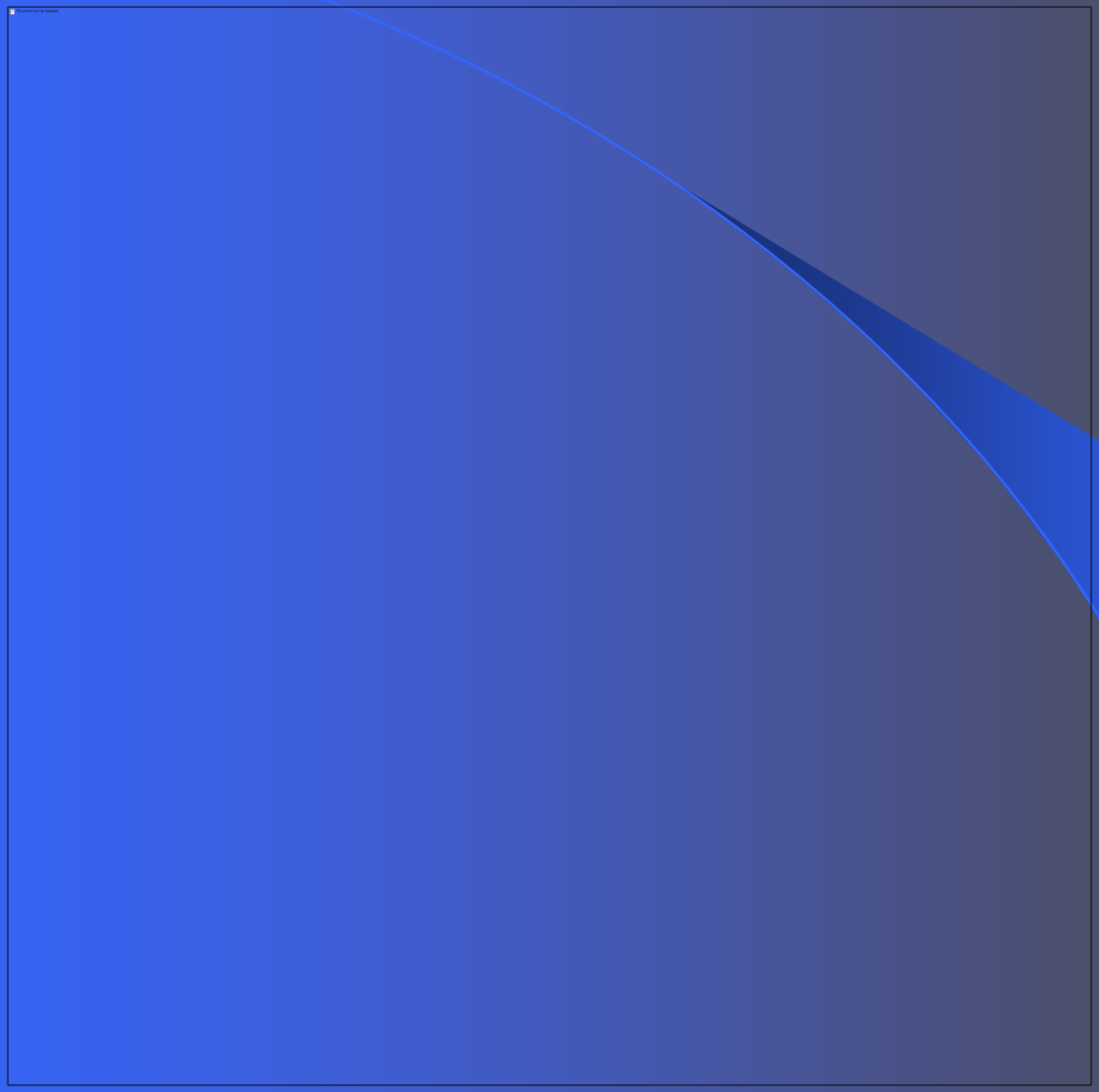


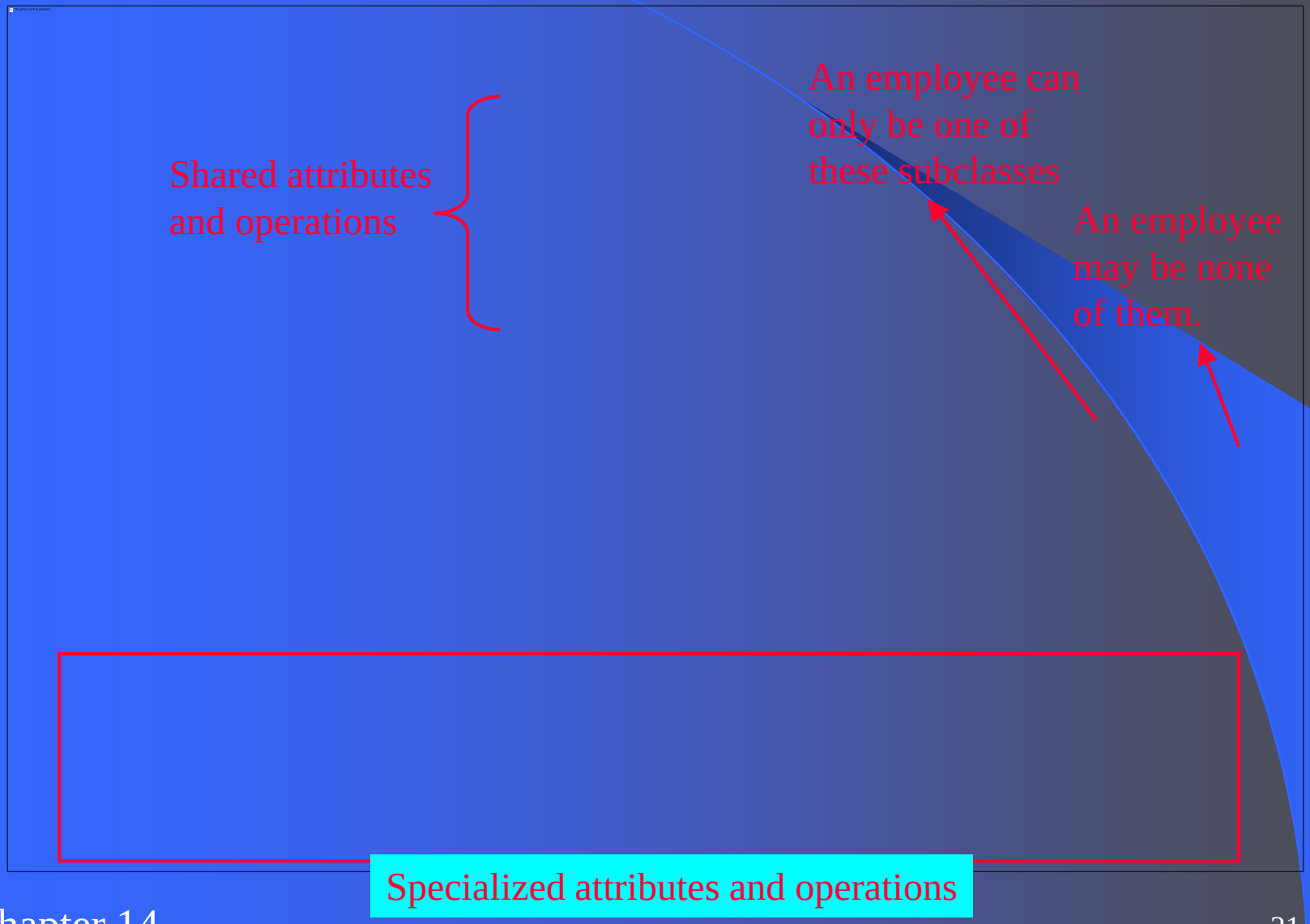
Figure 14-8 – Derived attribute, association, and role

Derived attributes an relationships shown with / in front of the name

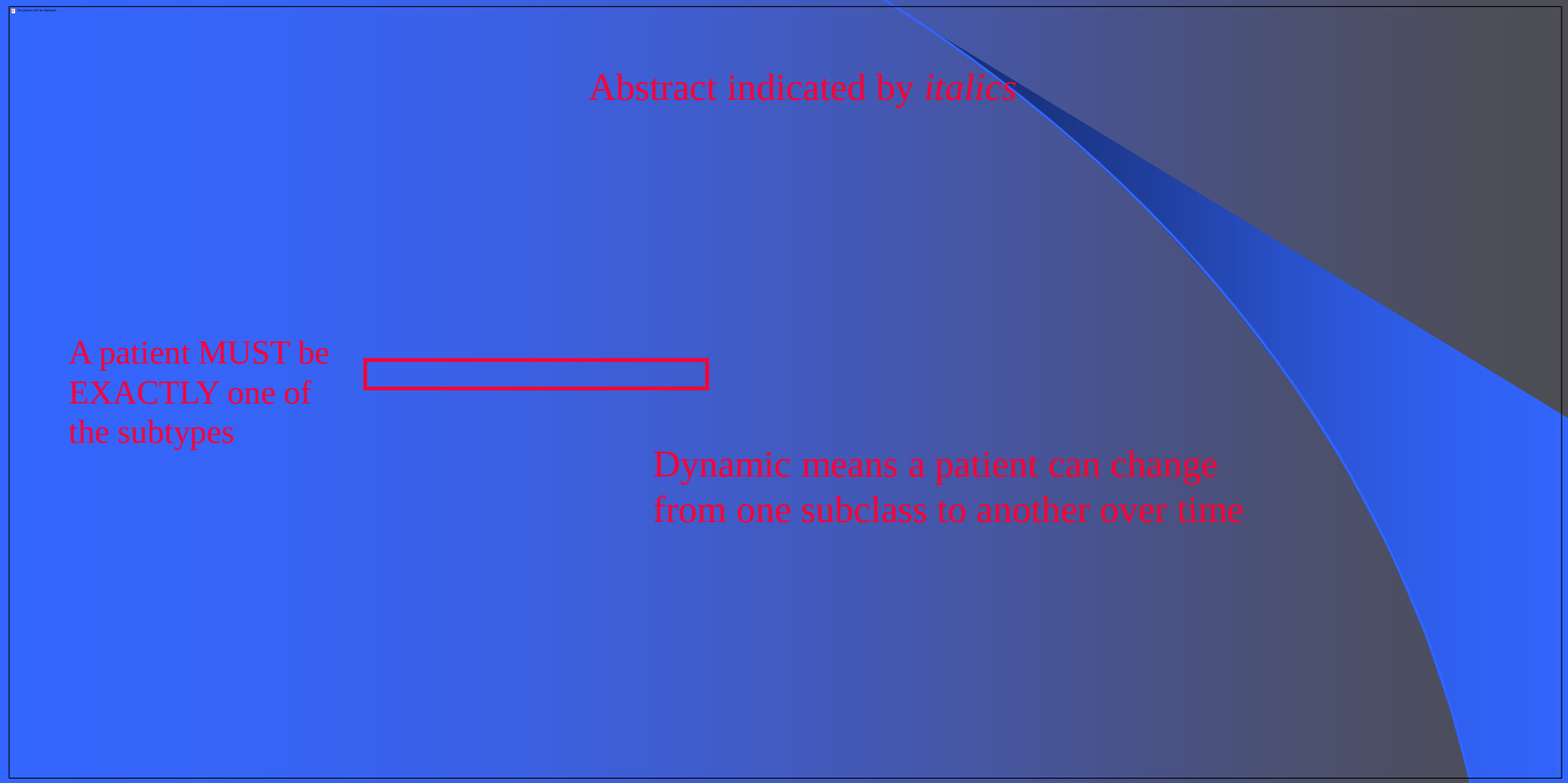
Generalization/Specialization

- Subclass, superclass
  - similar to subtype/supertype in EER
- Common attributes, relationships, ***AND operations***
- Disjoint vs. Overlapping
- Complete (total specialization) vs. incomplete (partial specialization)
- Abstract Class: no direct instances
- Concrete Class: direct instances

Figure 14-9 – Examples of generalization, inheritance, and constraints (a) Employee superclass with three subclasses



(a) Abstract patient class with two concrete subclasses



Class-Level Attribute

Specifies a value common to an entire class, rather than a specific value for an instance.

Represented by underlining

“=“ is initial, default value.

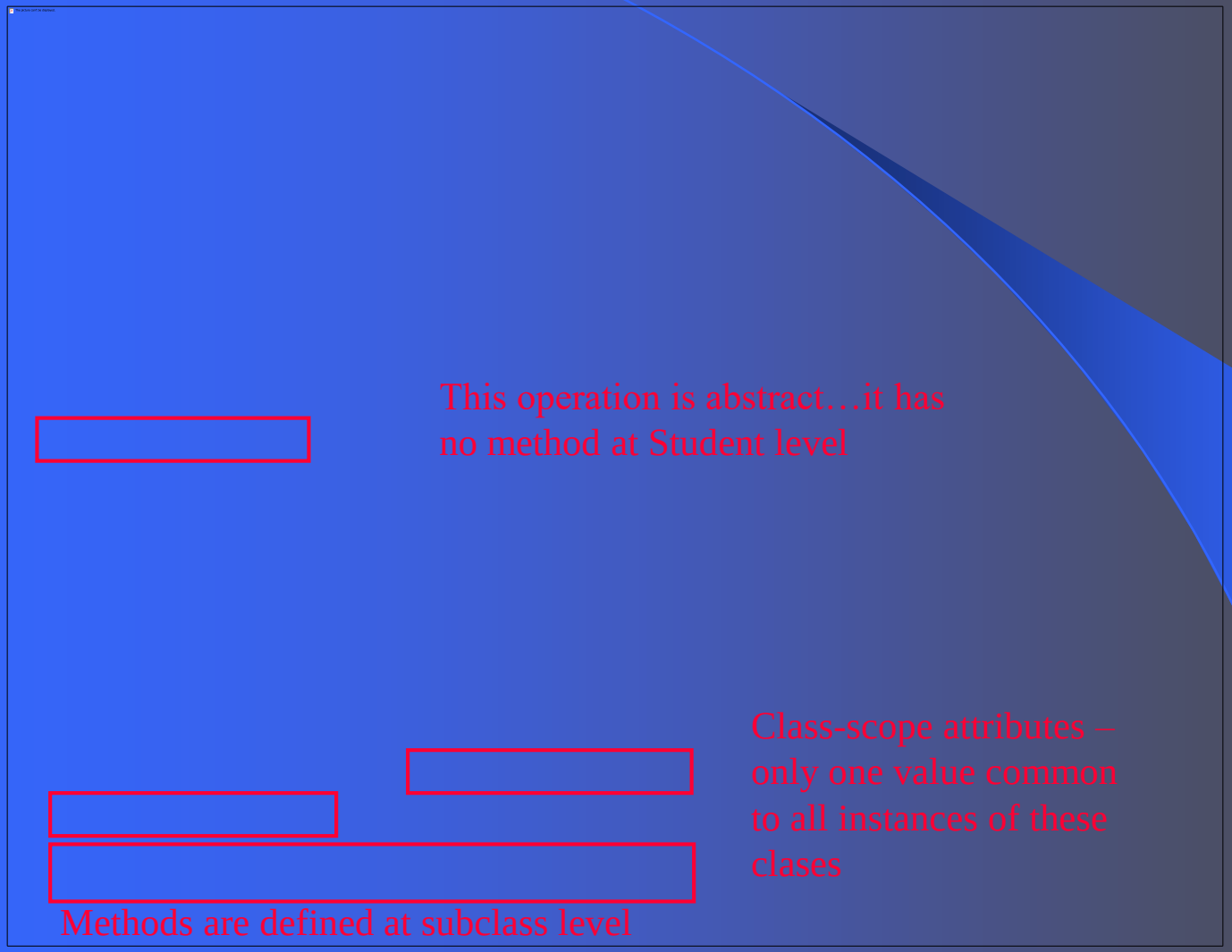
Polymorphism

Abstract Operation: Defines the form or protocol of the operation, but not its implementation.

Method: The implementation of an operation.

Polymorphism: The same operation may apply to two or more classes in different ways

Figure 14-11 – Polymorphism, abstract operation, class-scope attribute, and ordering

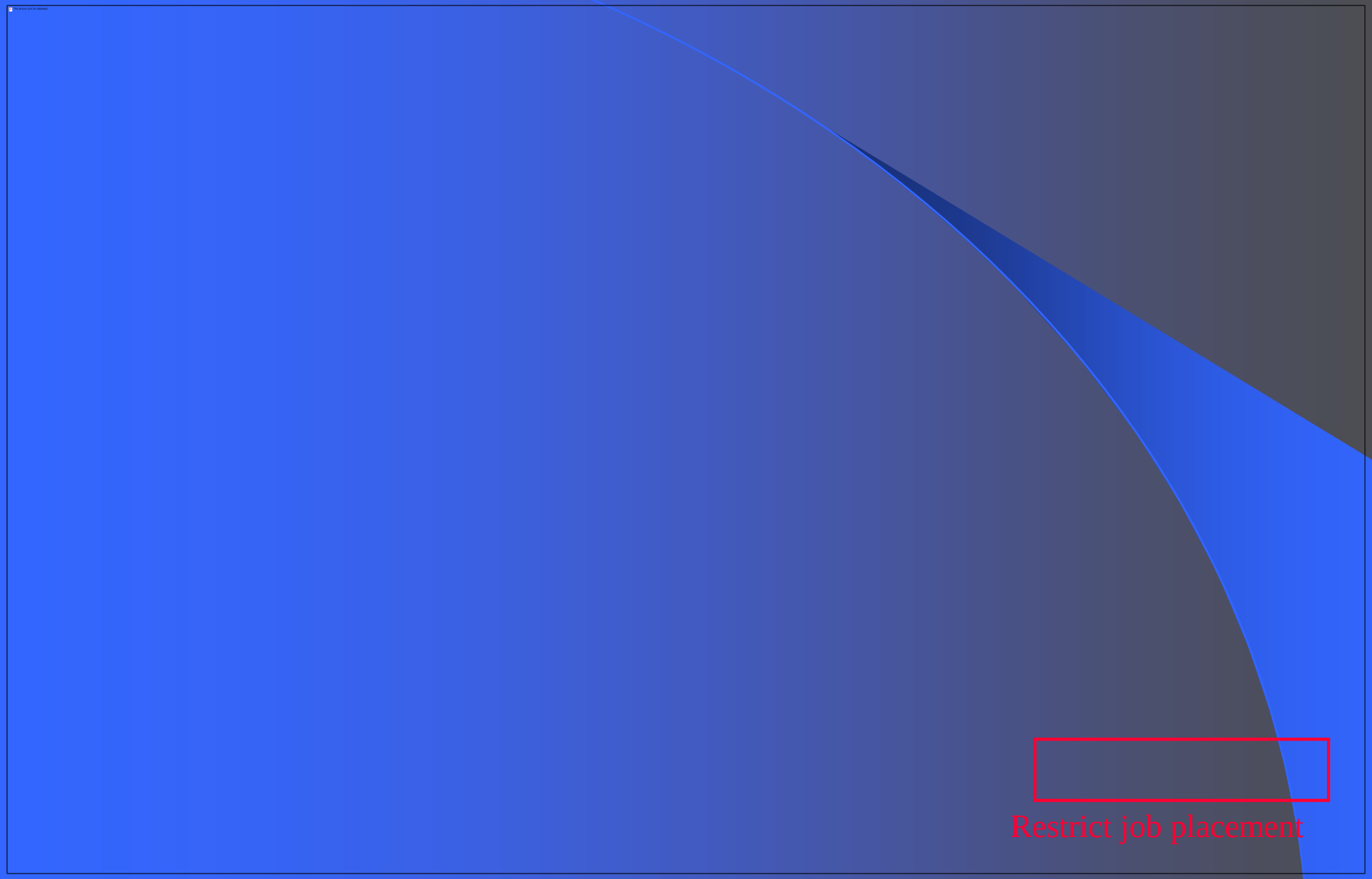


Overriding Inheritance

Overriding: The process of replacing a method inherited from a superclass by a more specific implementation of that method in a subclass.

- For Extension: add code.
- For Restriction: limit the method.
- For Optimization: improve code by exploiting restrictions imposed by the subclass.

Figure 14-12 – Overriding inheritance

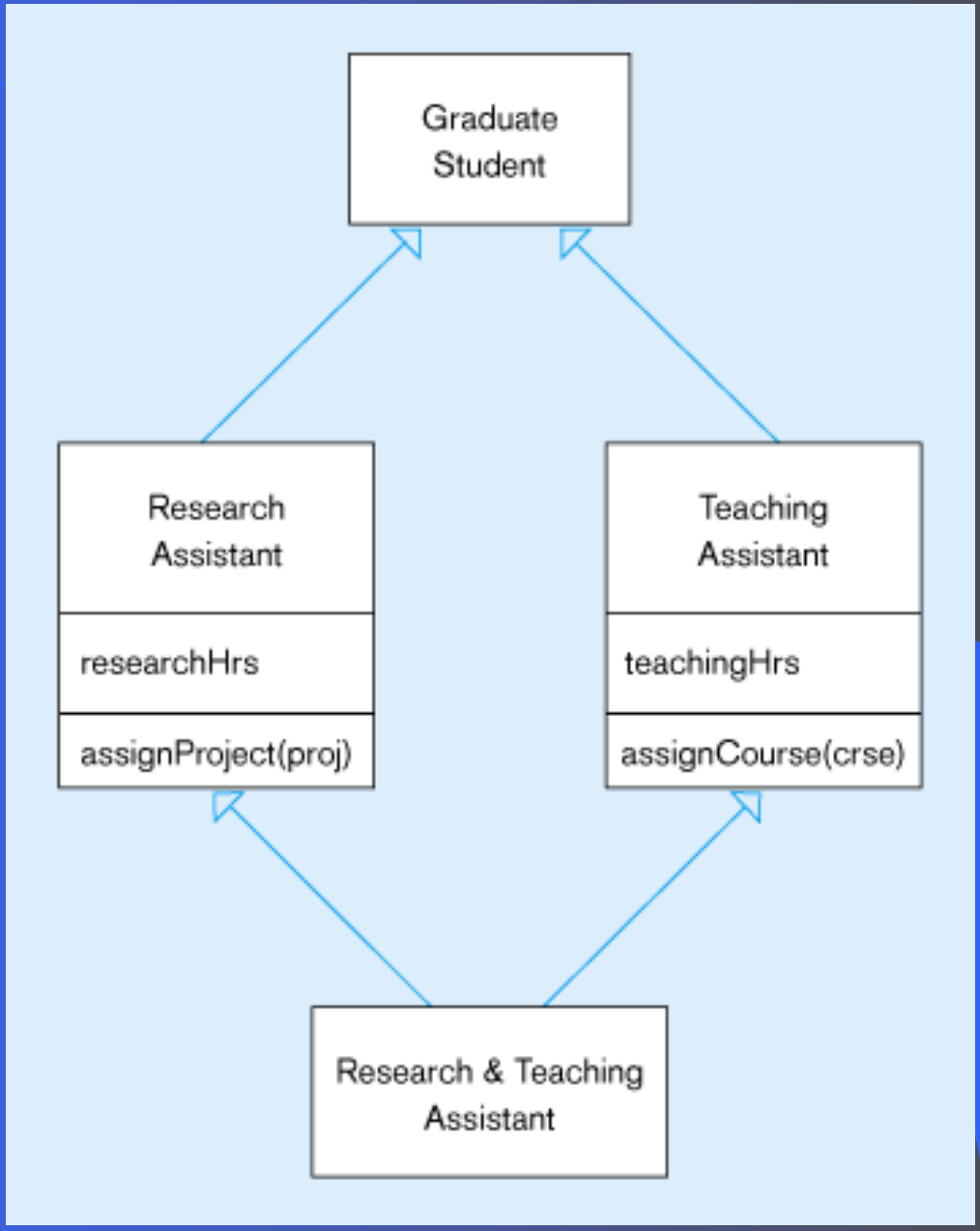


Multiple Inheritance

Multiple Classification: An object is an instance of more than one class.

Multiple Inheritance: A class inherits features from more than one superclass.

Figure 14-13
Multiple inheritance



Aggregation

Aggregation: A part-of relationship between a component object and an aggregate object.

Composition: A stronger form of aggregation in which a part object belongs to only one whole object and exists only as part of the whole object.

Recursive Aggregation: composition where component object is an instance of the same class as the aggregate object.

Figure 14-14 – Example aggregation

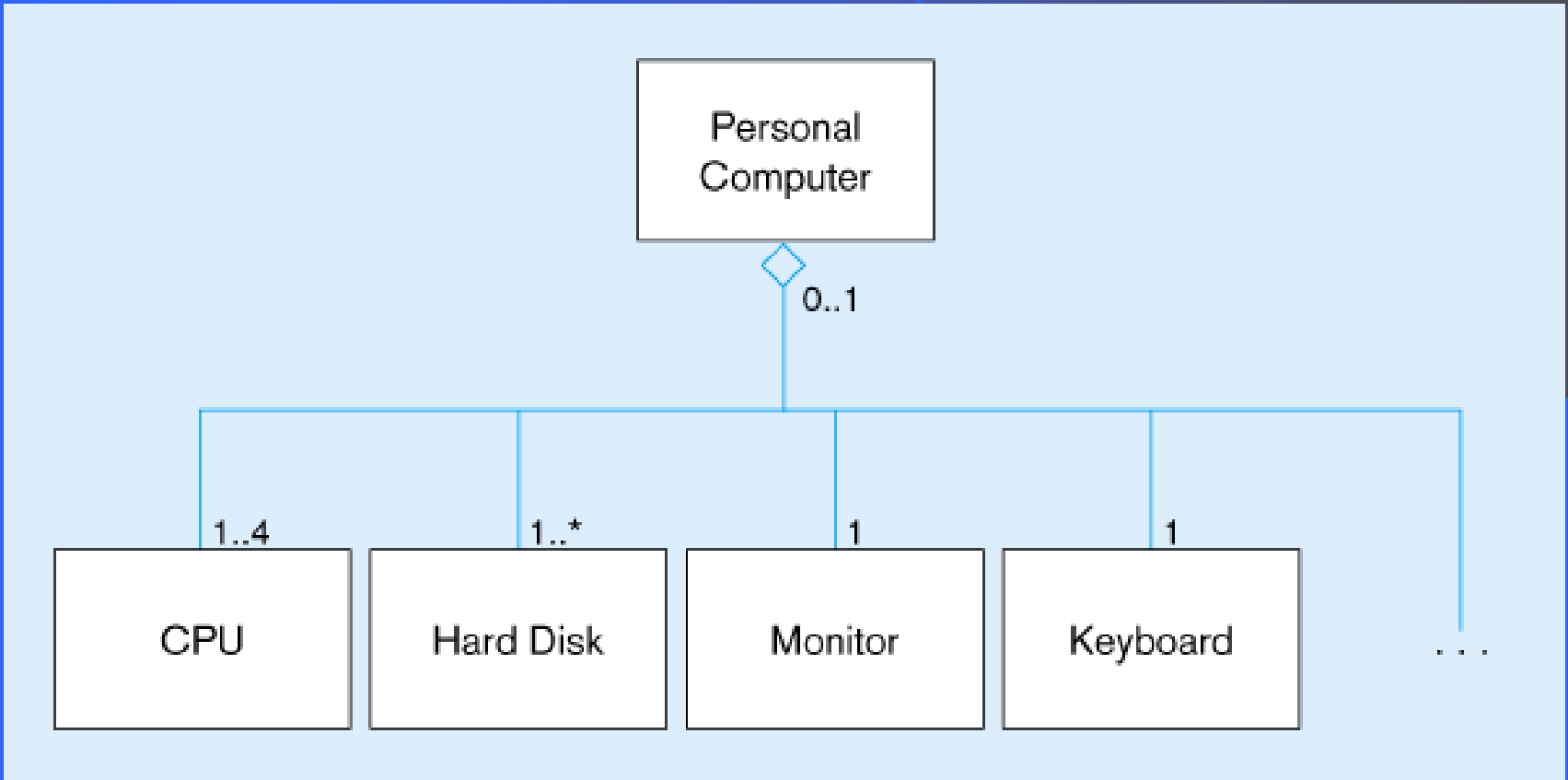
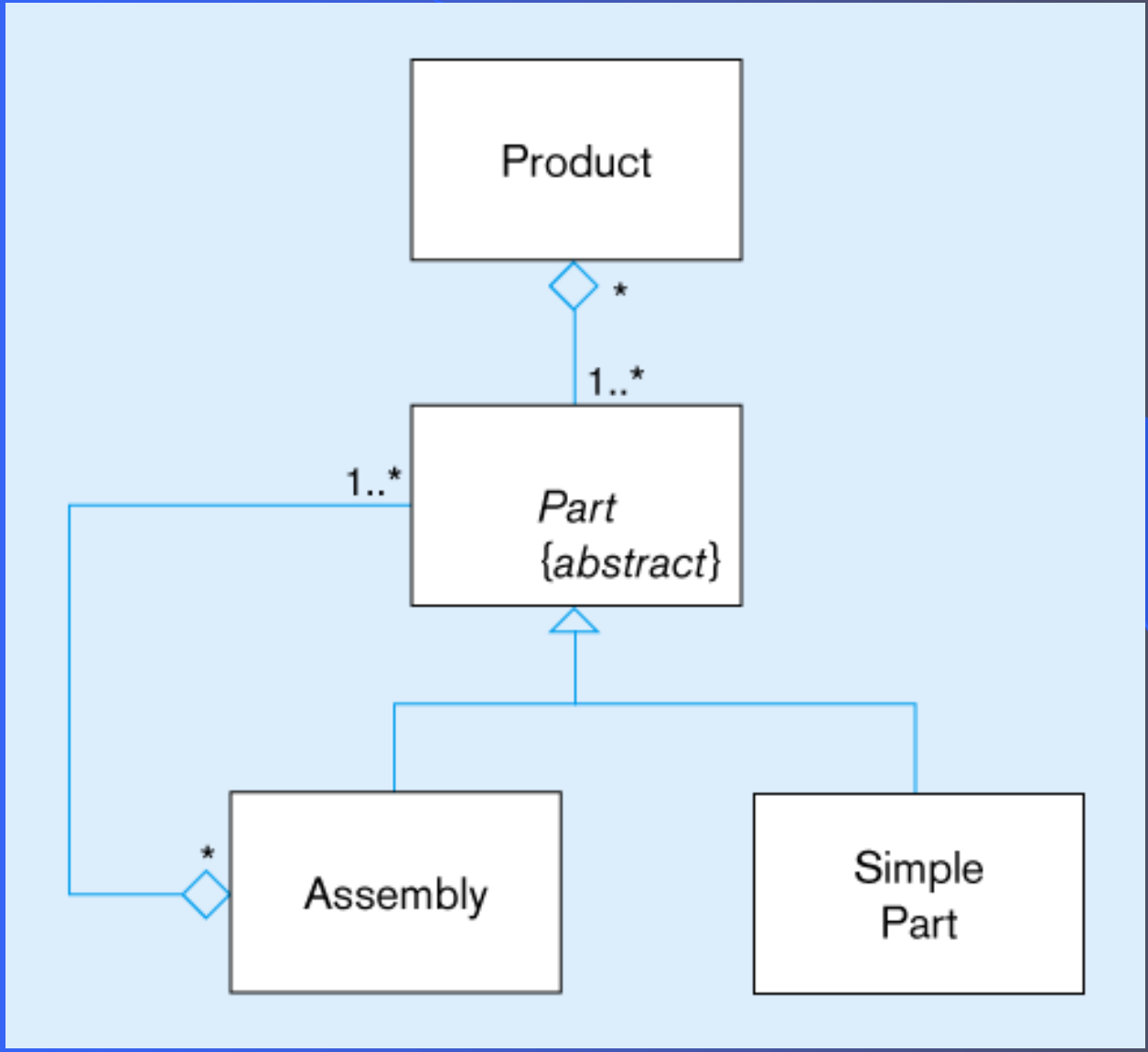


Figure 14-15 – Aggregation and Composition

(a) Class diagram

(b) Object diagram

Figure 14-16 – Recursive aggregation



Business Rules

See chapters 3 and 4

Implicit and explicit constraints on objects – for example:

- cardinality constraints on association roles
- ordering constraints on association roles

Business rules involving two graphical symbols:

- labeled dashed arrow from one to the other

Business rules involving three or more graphical symbols:

- note with dashed lines to each symbol

Figure 14-17 – Representing business rules

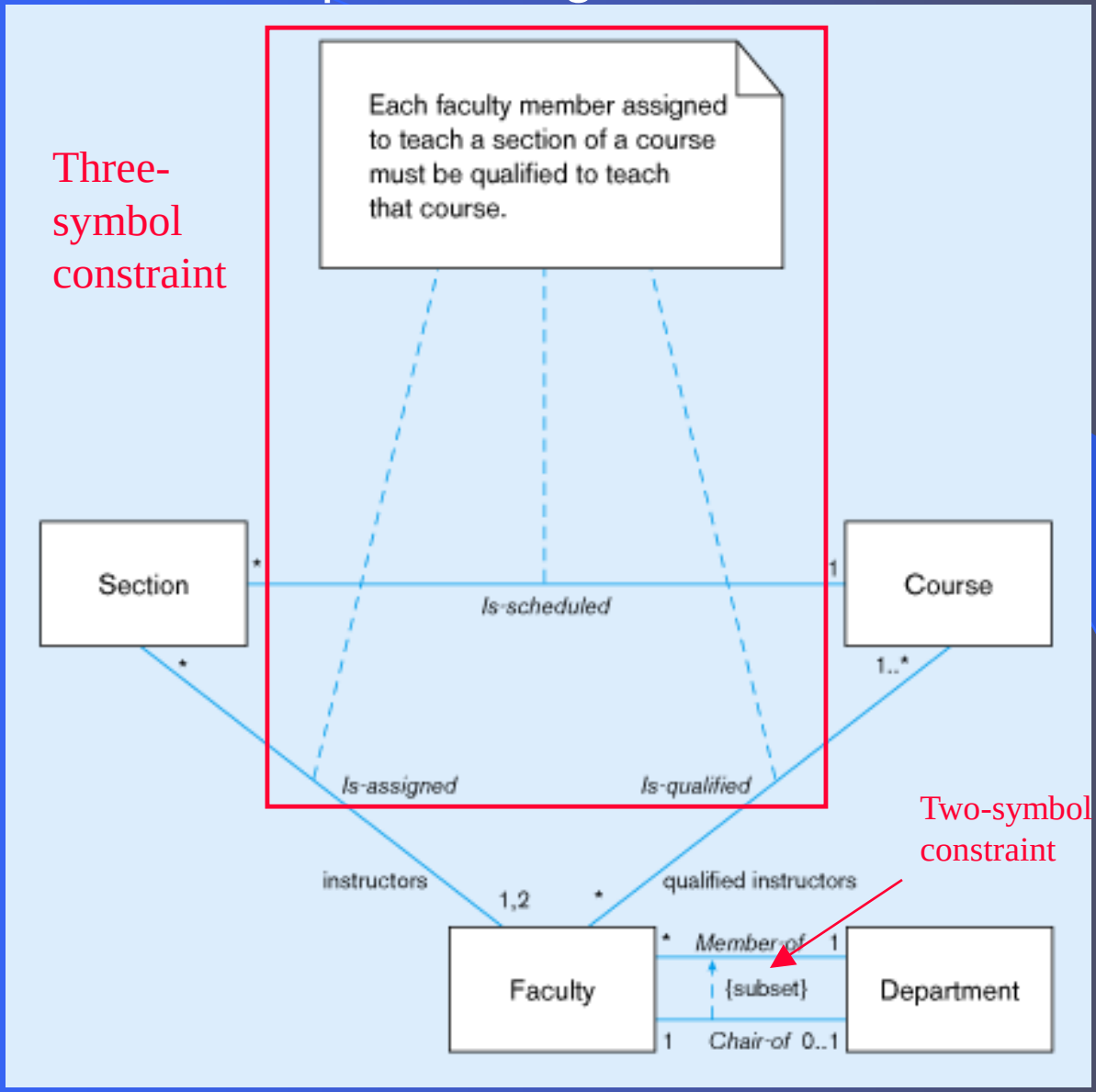


Figure 14-18 – Class diagram for Pine Valley Furniture Company