In the name of God

Modeling with UML

Advanced topics in Software Engineering

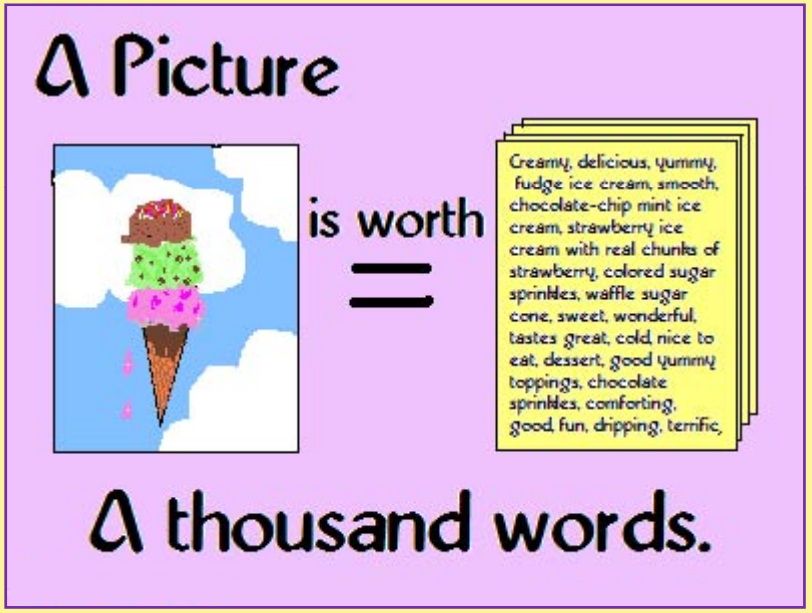
Leila Samimi-Dehkordi
PhD. Student
Department of Software Engineering
Faculty of Computer Engineering
University of Isfahan
Fall 2014
Email: *l.samimi@gmail.com*

Introduction

- UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- UML was created by Object Management Group (OMG).
- UML stands for **U**nified **M**odeling **L**anguage.
- UML is different from the other common programming languages like C++, Java, COBOL etc.
  - UML can be used to generate code in various languages.
- UML is a pictorial language used to make software blue prints.
- It is also used to model non software systems as well like process flow in a manufacturing unit etc.
- UML has a direct relation with object oriented analysis and design.

Goals

- *A picture is worth a thousand words.*
- to define some general purpose modeling language which all modelers can use
- simple to understand and use for anybody.



Conceptual Model

- A conceptual model can be defined as a model which is made of concepts and their relationships.
- A conceptual model is the first step before drawing a UML diagram.
- It helps to understand the entities in the real world and how they interact with each other.
- As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually.

Conceptual model of UML

- Conceptual model of UML can be mastered by learning the following three major elements:
  - UML building blocks
  - Rules to connect the building blocks
  - Common mechanisms of UML

Building blocks of UML

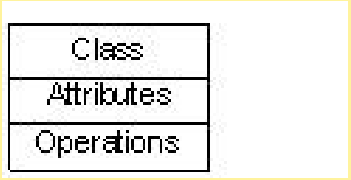
- The building blocks of UML can be defined as:
- Things:
  - They are the most important.
- Relationships:
  - They show how elements are associated with each other and this association describes the functionality of an application.
- Diagrams:
  - UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

Things

- Structural
  - They define the static part of the model and represent physical and conceptual elements.
- Behavioral
  - A behavioral thing consists of the dynamic parts of UML models.
- Grouping
  - They can be defined as a mechanism to group elements of a UML model together.
- Annotational
  - They can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

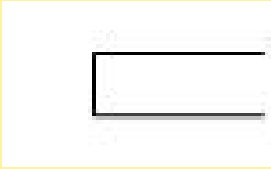
Structural Things

- **Class:** represents set of objects having similar responsibilities.



- **Interface:** defines a set of operations which specify the responsibility of a class.

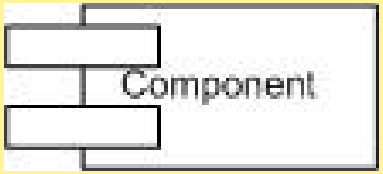
- **Collaboration:** defines interaction between elements.



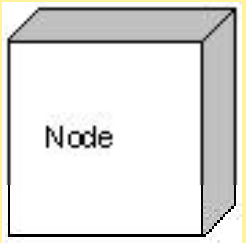
Structural Things (2)

- **Use case:** represents a set of actions performed by a system for a specific goal.

- **Component:** describes physical part of a system.



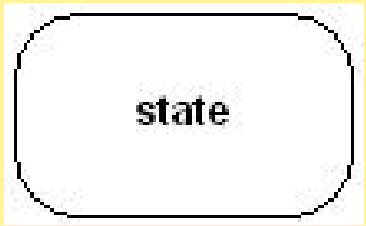
- **Node:** can be defined as a physical element that exists at run time.



Behavioral things

- **Interaction:** is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

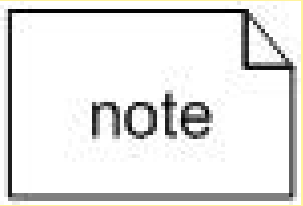
- **State machine:** defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



Grouping and Annotational Things

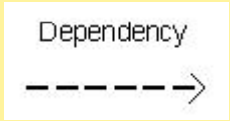
- **Package** is the only one **grouping** thing available for gathering structural and behavioral things.

- **Note** is the only one **Annotational** thing available.
  - A note is used to render comments, constraints etc of an UML element.



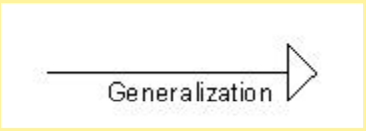
Relationship

- **Dependency** is a relationship between two things in which change in one element also affects the other one.



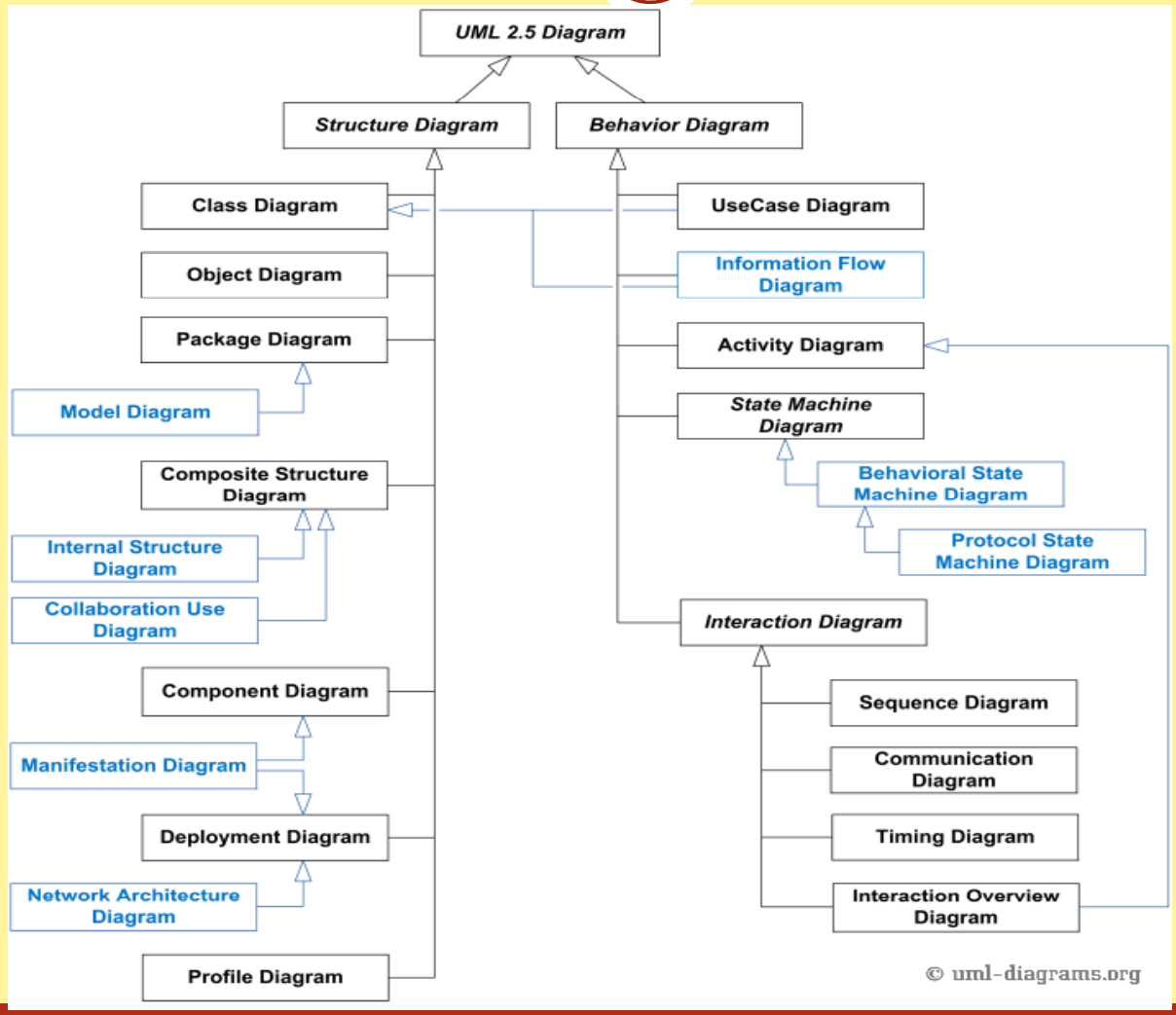
- **Association** is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

- **Generalization** can be defined as a relationship which connects a specialized element with a generalized element (inheritance).



- **Realization** can be defined as a relationship in which two elements are connected that one describes some responsibility which is not implemented and the other implements them.

UML diagrams

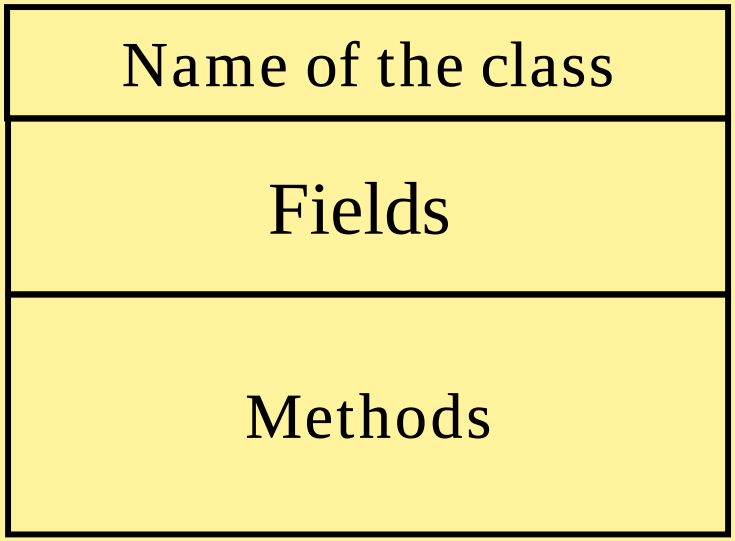


Class diagrams

- A class diagram shows classes, interfaces, and their relationships
- We'll cover most of classes and interfaces, and a few of the most important relationships

Classes

- A class is drawn as a rectangle with two or three compartments:



Field

- A Field (variable) is written as:

  visibility name : type

  where:
  - **+** means public visibility
  - **#** means protected visibility
  - **-** means private visibility
  - <*blank*> means default (package) visibility
- Example: +length:int

Field (2)

- Static variables are underlined
- An initial value can be shown with ***=value***
- Example:

  -numberOfEmployees:int=10

  means numberOfEmployees is:
  - private
  - static
  - integer
  - and has 10 as its initial value

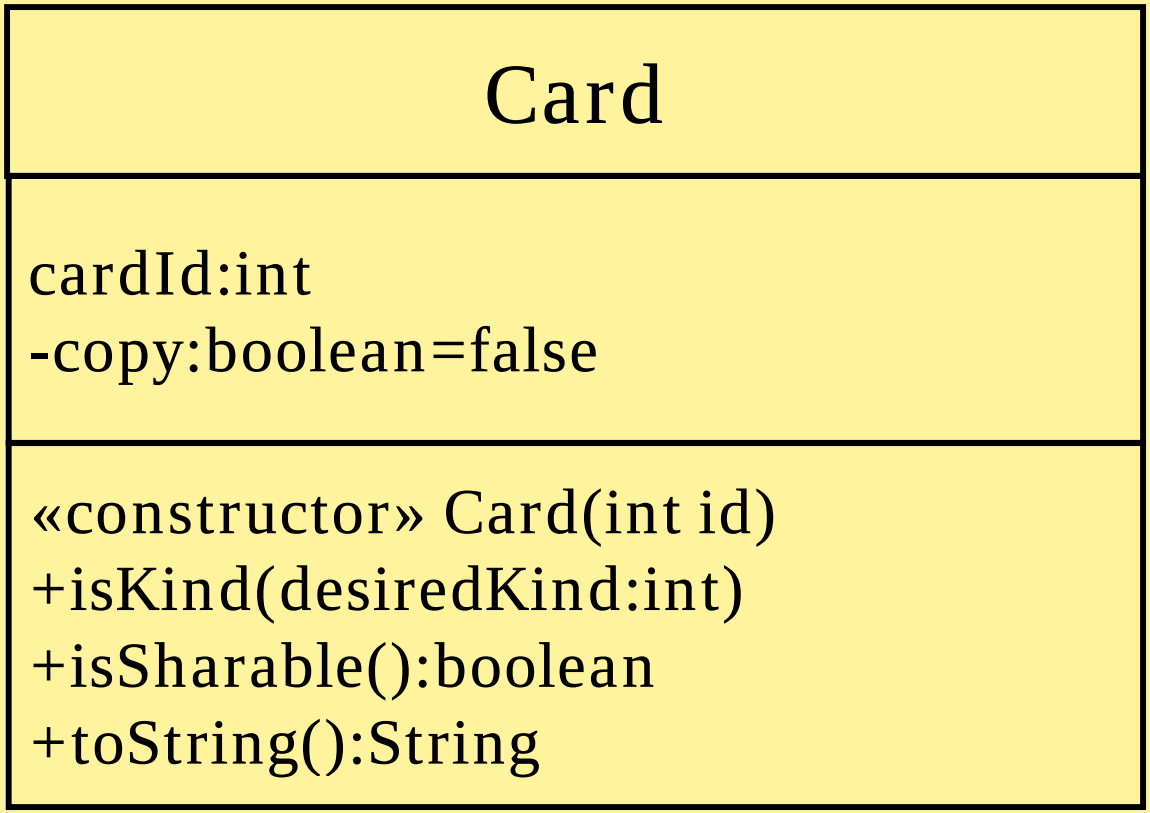
Methods

- Methods are written as:

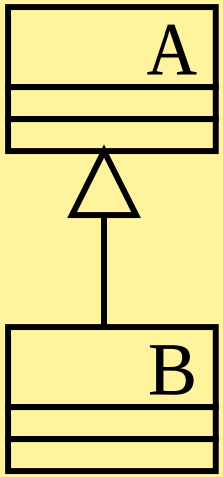
  visibility name (parameters) : returnType

  where
  - ***visibility*** uses the same syntax variables (+, -, #, blank)
  - parameters are given as ***name:type***
  - if the ***returnType*** is void, it is omitted
  - constructors are preceded by «constructor»
  - interfaces are preceded by «interface»
  - an ellipsis (...) indicates omitted methods

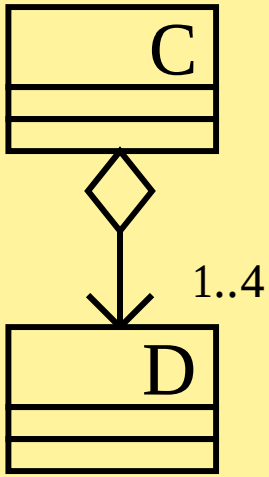
Example of a class



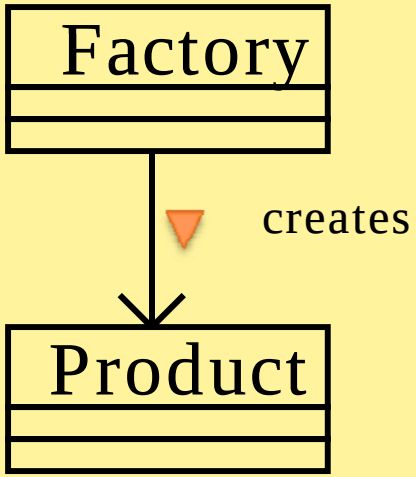
Types of relationships



Class B extends class A



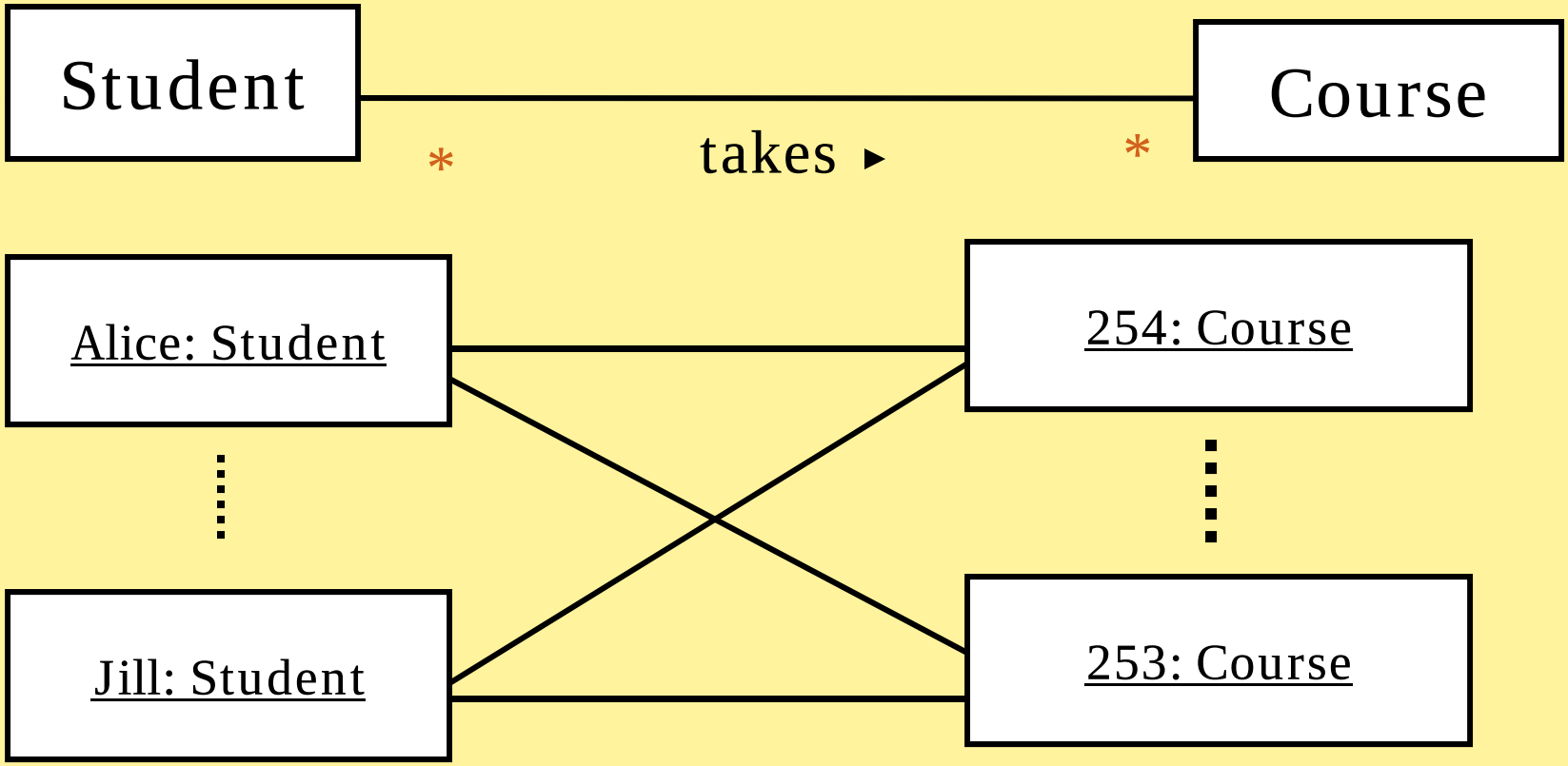
Class C contains 1 to 4 objects of class D



Other kinds of relations

Association - Multiplicity

- A Student can take many Courses and many Students can be enrolled in one Course.



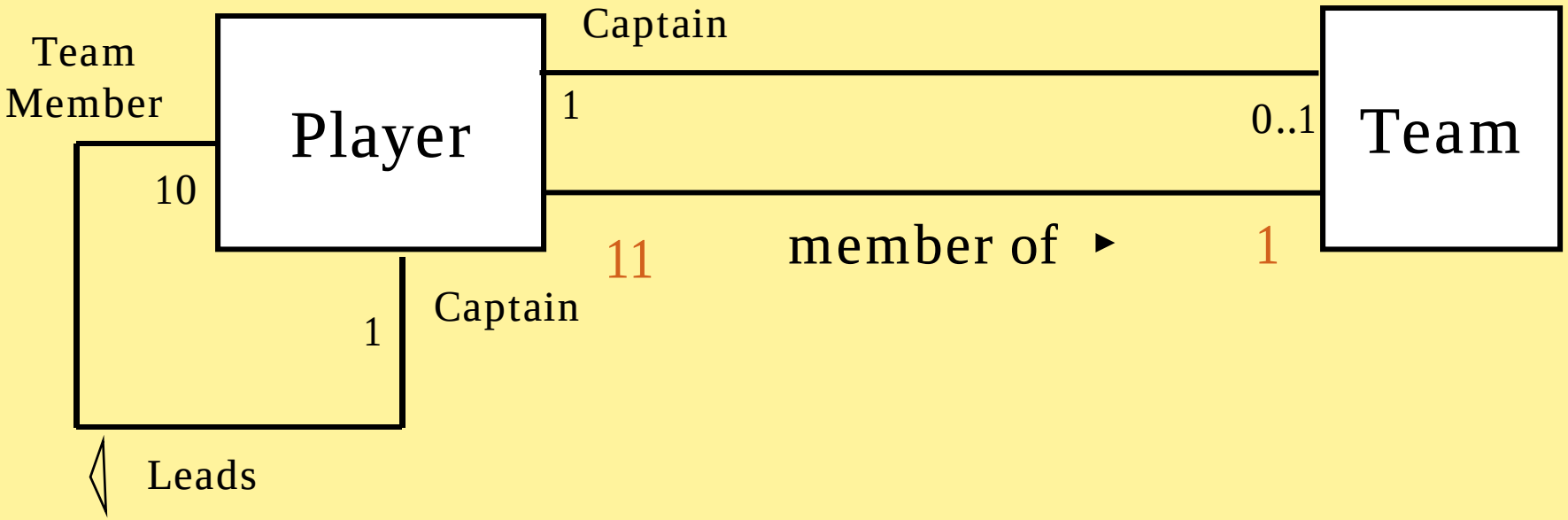
Notes of Multiplicity

- One class can be relate to another in a
  - One-to-one
  - One-to-many
  - One-to-one or more
  - One-to-zero or one
  - One-to-a bounded interval (one-to-two through twenty)
  - One-to-exactly n
  - One-to-a set of choices (one-to-five or eight)

- Multiplicity can be expressed as,
  - Exactly one - 1
  - Zero or one - 0..1
  - Many - 0..* or *
  - One or more - 1..*
  - Exact Number - e.g. 3..4 or 6
  - Or a complex relationship – e.g. 0..1, 3..4, 6..* would mean any number of objects other than 2 or 5

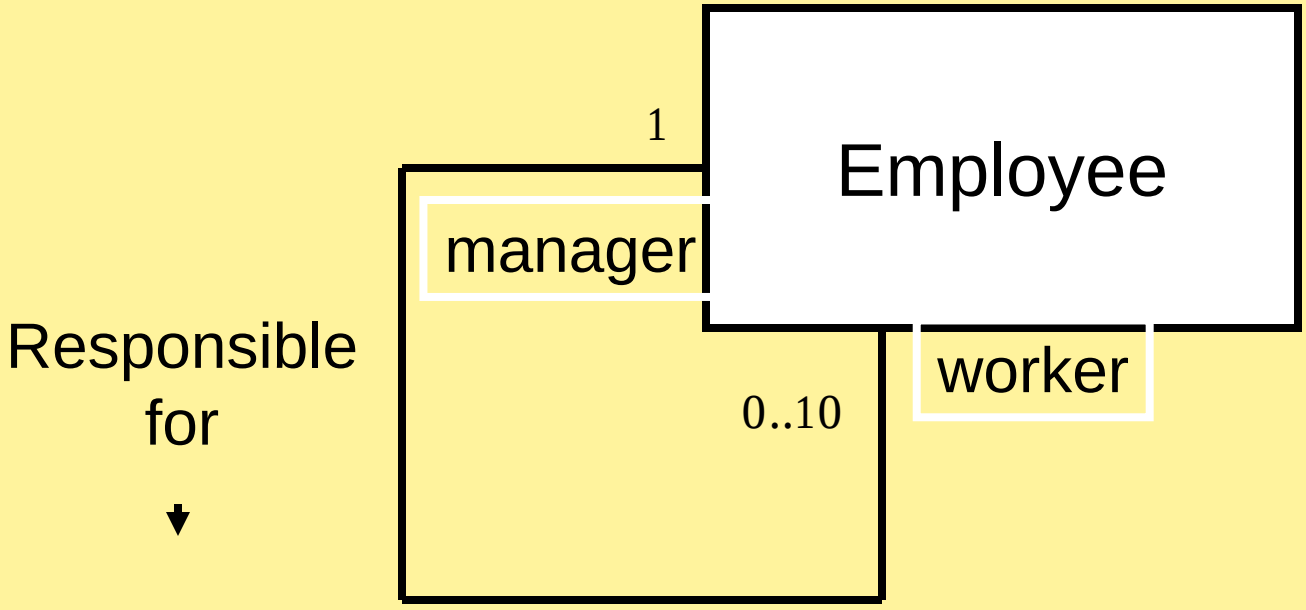
Association - Multiplicity

- A cricket team has 11 players. One of them is the captain.
- A player can play only for one Team.
- The captain leads the team members.



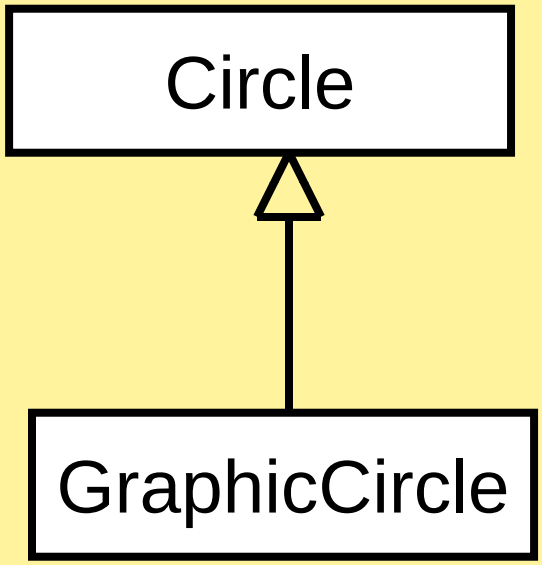
Self Association

- An association that connects a class to itself is called a self association.
- A Company has Employees.
- A single manager is responsible for up to 10 workers.



Generalization (Inheritance)

- Child class is a special case of the parent class

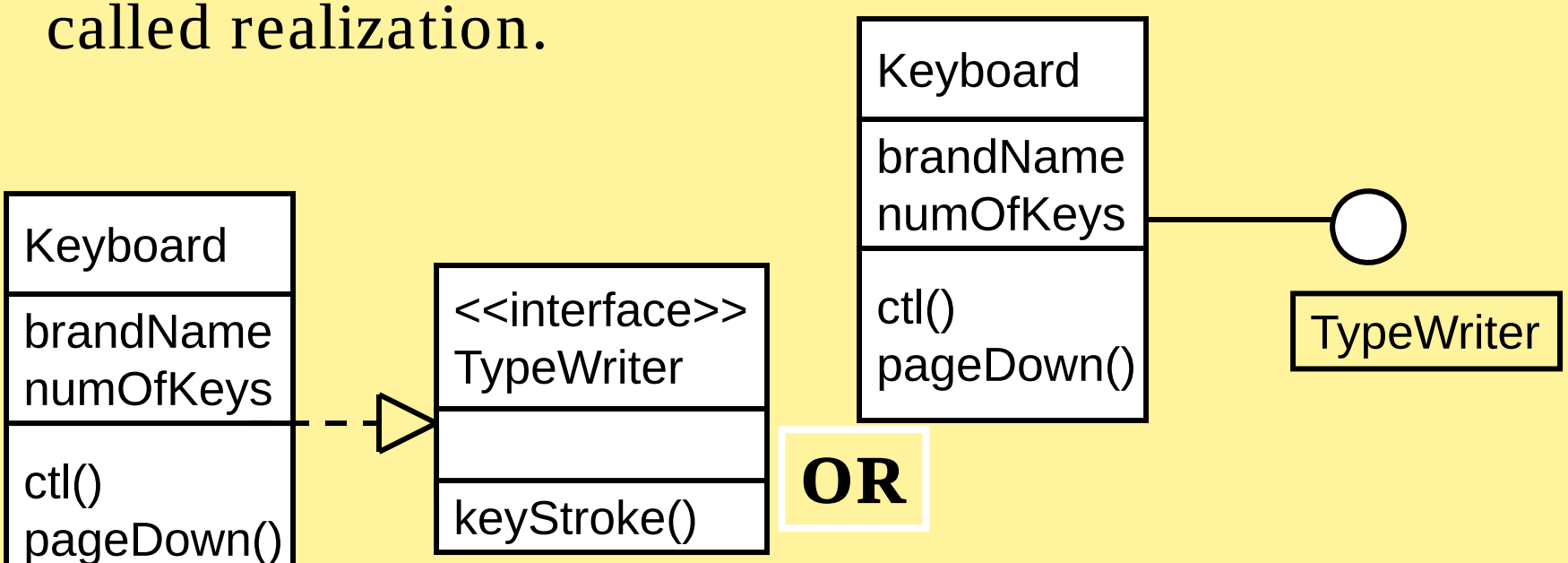



```
public class Circle {

}
```

```
public class GraphicCircle extends Circle {

}
```

Realization- Interface

- Interface is a set of operation the class carries out.
- The relationship between a class and an interface is called realization.



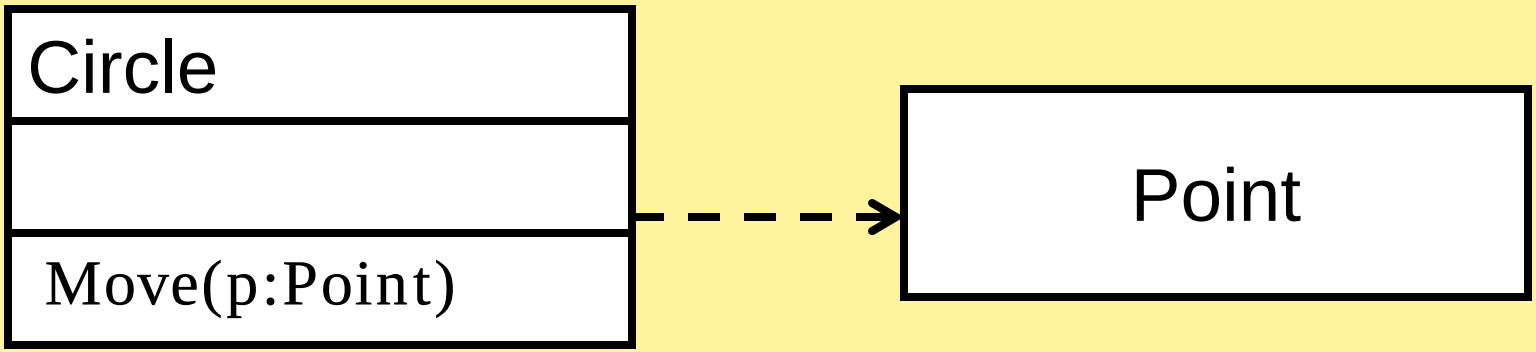
An interface does not have attributes. It Only has operations.

Realization - Implementation


```
public interface TypeWriter {
   void keyStroke()

}

public class KeyBoard implements TypeWriter {
   public void  keyStroke(){
    ……

   }
}
```

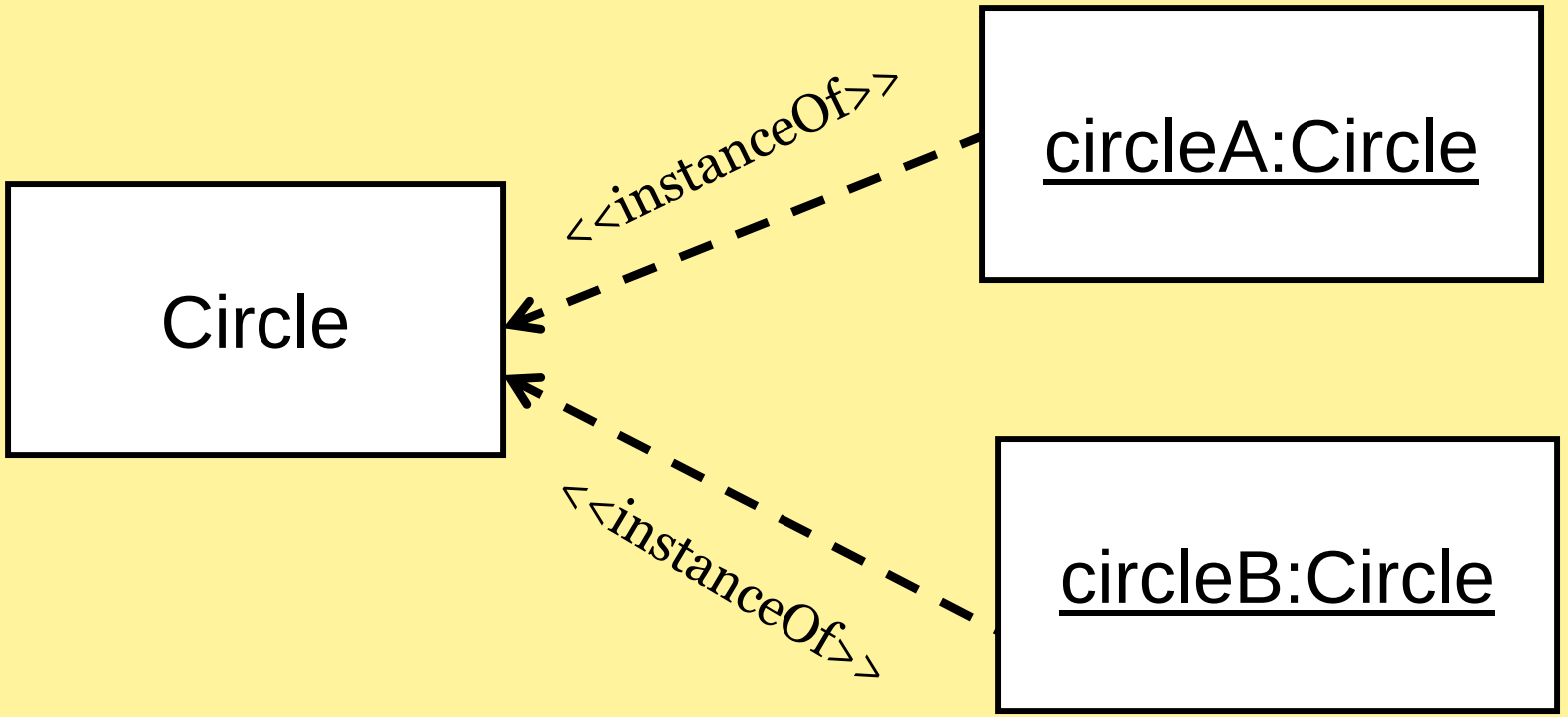
Dependency

- Change in specification of one class can change the other class. This can happen when one class is using another class.



Dependency (2)

- Dependency relationship can be used to show relationships between classes and objects.

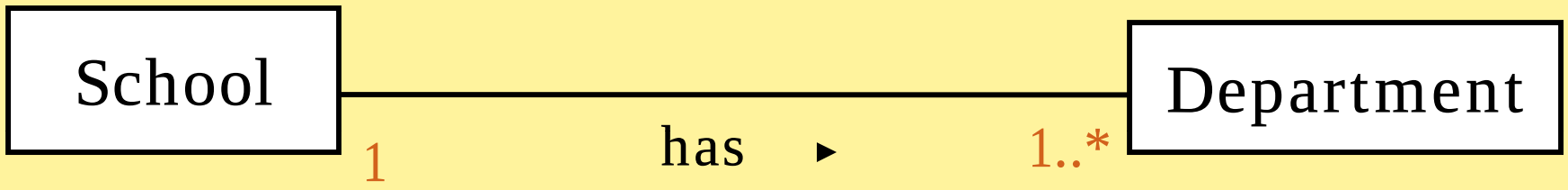


Class Diagram - Example

- Draw a class diagram for a information modeling system for a school.
  - School has one or more Departments.
  - Department offers one or more Subjects.
  - A particular subject will be offered by only one department.
  - Department has instructors and instructors can work for one or more departments.
  - Student can enroll in up to 5 subjects in a School.
  - Instructors can teach up to 3 subjects.
  - The same subject can be taught by different instructors.
  - Students can be enrolled in more than one school.

Class Diagram - Example

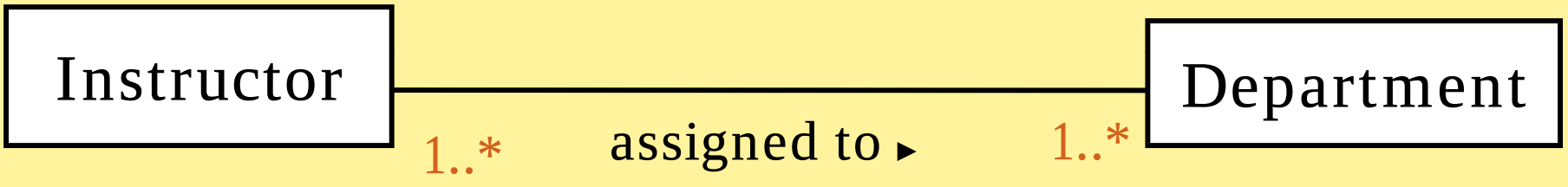
- School has one or more Departments.



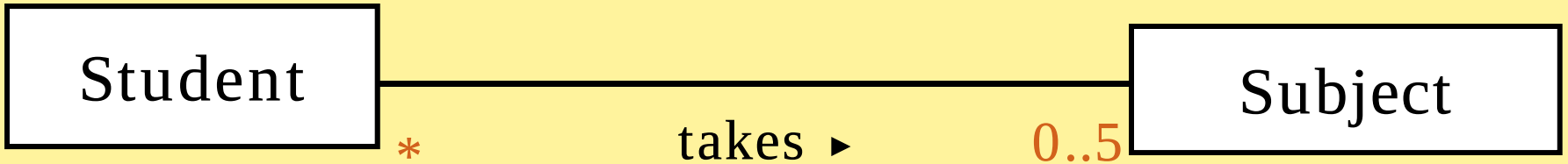
- Department offers one or more Subjects.
- A particular subject will be offered by only one department.

Class Diagram - Example

- Department has Instructors and instructors can work for one or more departments.



- Student can enroll in up to 5 Subjects.

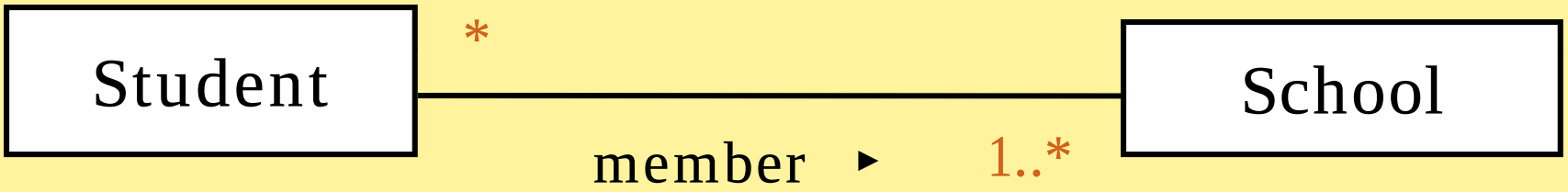


Class Diagram - Example

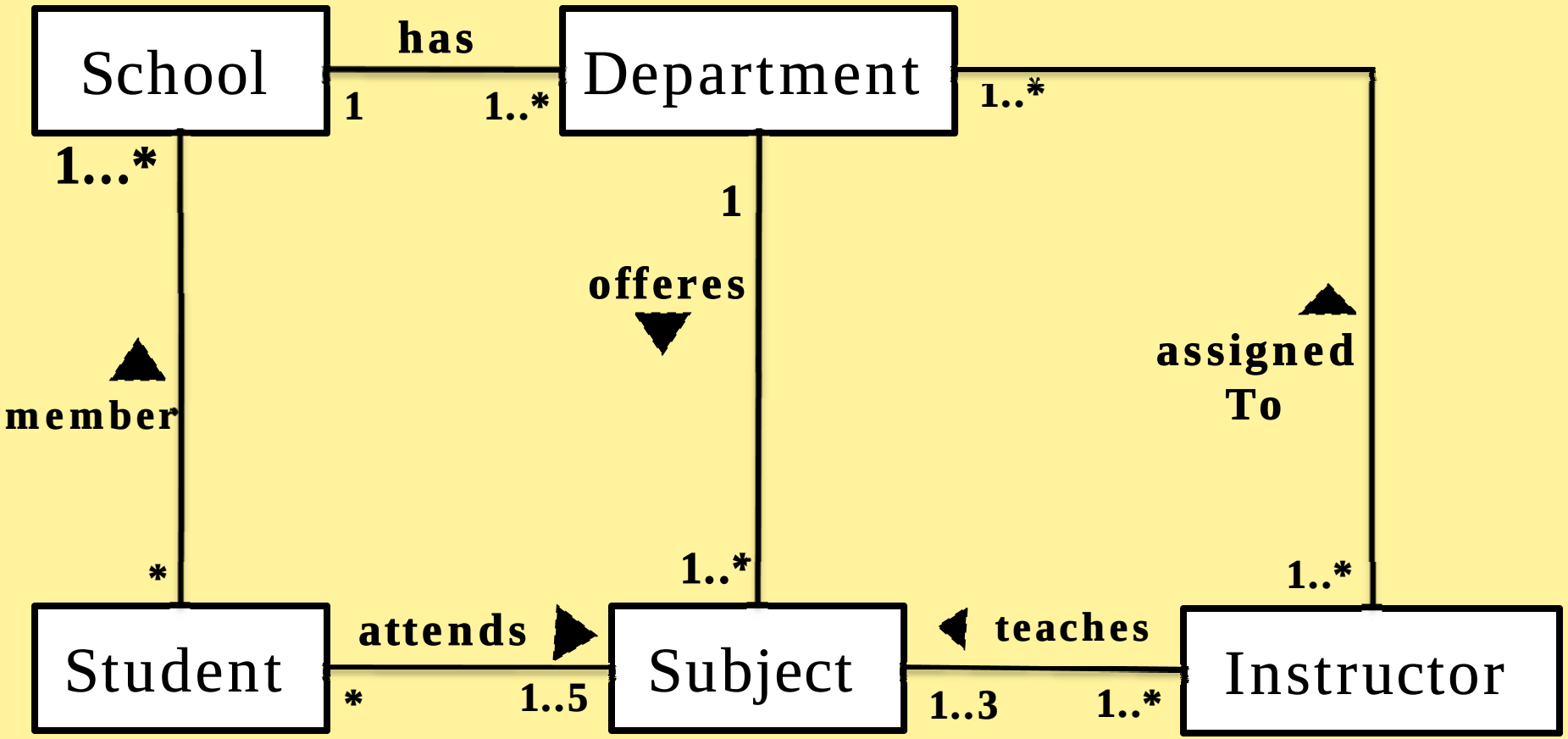
- Instructors can teach up to 3 subjects.
- The same subject can be taught by different instructors.

Class Diagram - Example

- Students can be enrolled in more than one school.



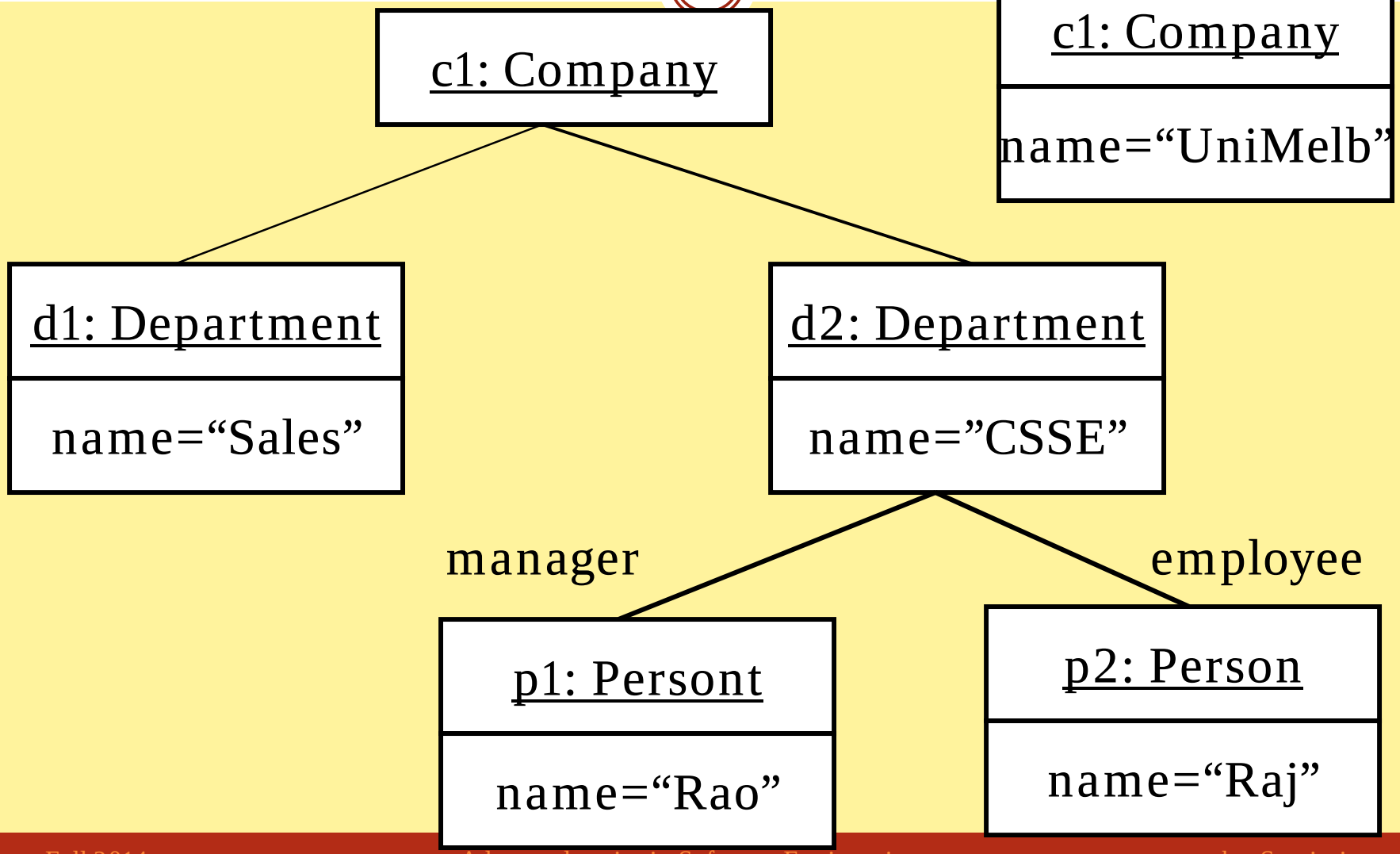
Class Diagram Example



Object Diagram

- Object Diagram shows the relationship between objects.
- It is a graph of instances, including objects and data values.
- A static object diagram is an instance of a class diagram;
  - it shows a snapshot of the detailed state of a system at a point in time.
- Unlike classes objects have a state.

Object Diagram - Example



UML tools

- Rational Rose is *the* "real world" standard; full round-trip code generation
  - Recently acquired by IBM (right under Microsoft's nose!)
- Together (from Borland) is a lot like Rational Rose
  - I haven't used it in about three years (since before Borland bought it from TogetherSoft)
- ArgoUML looks interesting (and is open source)
- BlueJ, of course, displays simple UML diagrams
- Drawing programs with UML support
  - Visio is a Microsoft tool
  - Dia is a freeware clone of Visio

Good reference

- A good reference for learning UML is:

  http://www.uml-diagrams.org/

Thanks for your attention