

CEN206 Object-Oriented Programming (formerly CE204)

Week-6 (UMPLE - Part 1)

Spring Semester, 2024-2025

Download [DOC-PDF](#), [DOC-DOCX](#), [SLIDE](#), [PPTX](#),



UMPLE

Common Scope

- What is UMPLE?
- What is its purpose?
- How to create a UML model with UMPLE?
- What is philosophy of UMPLE?

Common Scope

- How to use UMPLE?
 - UMPLE Online
 - Command-Line
 - Eclipse Plugin
 - Visual Studio Code Plugin

Common Scope

- How to learn UMPLE?
 - Online Documentations
 - Video Tutorials
 - UMPLE Community

Common Scope

- Overview of the basics of UMPLE
- Associations in UMPLE
- State machines in UMPLE
- Product lines in UMPLE: Mixins and Mixsets
- Other separation of concerns mechanisms: (Aspects and traits) and their code generation
- Other advanced features of UMPLE
- Hands-on exercise developing versions of a concurrent system using state machines and product lines.
- UMPLE as written in itself: A case study.

Common Scope

- Introduction:
- Overview of Model-Driven Development
 - Languages / Tools / Motivation for UMPLE
- Class Modeling
 - Tools / Attributes / Methods / Associations / Exercises / Patterns
- Modeling with State Machines
 - Basics / Concurrency / Case study and exercises
- Separation of Concerns in Models
 - Mixins / Aspects / Traits
- More Case Studies and Hands-on Exercises
 - UMPLE in itself / Real-Time / Data Oriented
- Conclusion

Outline - UMPLE Part 1

- Introduction to UMPLE
- Motivation for developing UMPLE
- Some key UMPLE innovations
- Using UMPLE
- UMPLE Philosophy
- UMPLE Class Modeling

Outline - UMPLE Part 1

- UMPLE Online Usage
- UMPLE Attributes
- UMPLE Generalization and interfaces
- UMPLE Methods
- UMPLE Associations

Introduction to UMPLE

UMPLE: Simple, A&Mple, &UML &P&rogramming &L&anguag&e&

- **Open source textual modelling tool set for 3 platforms**
 - Command line compiler
 - Web-based tool (UMPLEOnline) for demos and education
 - Eclipse plugin
- **Code generator for UML ++**
 - Infinitely nested state machines, with concurrency
 - Proper referential integrity and multiplicity constraints on associations
 - Traits, mixins, aspects for modularity
 - Text generation templates, patterns, traits
- **Pre-processor to add UML, patterns and other features on top of Java, PhP, C++ and other languages**

UMPLE: Simple, Ample, UML Programming Language

- Open source textual modeling tool and code generator
 - Adds modeling to Java, C++, PHP
 - A sample of features
 - Referential integrity on associations
 - Code generation for patterns
 - Blending of conventional code with models
 - Infinitely nested state machines, with concurrency
 - Separation of concerns for models: mixins, traits, mixsets, aspects
- Tools
 - Command line compiler
 - Web-based tool (UMPLEOnline) for demos and education
 - Plugins for Eclipse and other tools

What Are we Going to Learn About in This Tutorial? What Will You Be Able To Do?

- Modeling using **class diagrams**
 - Attributes, Associations, Methods, Patterns, Constraints
- Modeling using **state diagrams**
 - States, Events, Transitions, Guards, Nesting, Actions, Activities
 - Concurrency
- **Separation of Concerns** in Models
 - Mixins, Traits, Aspects, Mixsets
- Practice with a examples focusing on **state machines** and **product lines**
- Building a complete system in UMPLE

What Technology Will You Need?

- As a minimum: Any web browser.
- For a richer command-line experience
 - A computer (laptop) with Java 8-14 JDK
 - Mac and Linux are the easiest platforms, but Windows also will work
 - Download UMPLE Jar at <http://dl.UMPLE.org>
- You can also run UMPLE in Docker: <http://docker.UMPLE.org>

Key Websites

- Entry-point: <https://www.UMPLE.org>
 - Everything you need to get started with UMPLE
- Github: <https://github.com/UMPLE/UMPLE>
 - Source code and examples for UMPLE
- UMPLE Online: <https://try.UMPLE.org>
 - Online application for UMPLE

Key Websites (Another way)

The UMPLEOnline web interface is at try.UMPLE.org

The user manual is at manual.UMPLE.org

The UMPLE home page is at www.UMPLE.org

UMPLE download page: dl.UMPLE.org

Motivation for developing UMPLE

Motivation for developing UMPLE (1)

Designers want the best combination of features:

- Textual editing and blending with other languages
- Ability to use in an agile process
 - Write tests, continuous integration, versioning
 - Combine the best of agility and modeling
- Excellent code generation
 - A complete generation of real systems (including itself)
- Multi-platform (command line, Eclipse, VsCode, Web)
- Practical and easy to use for developers
 - Including great documentation
- Open source

Motivation for developing UMPLE (2)

Many existing tools:

- Lacked in usability
 - Awkward to edit diagrams
 - Many steps to do a task
 - Lengthy learning process
- Lack in ongoing support
- Could be enhanced by us perhaps, but we would be tied to key decisions (e.g. Eclipse-only)

Some key UMPLE innovations

Some key UMPLE innovations

- Model is code
 - Traditional code is embedded in model
- No need to edit generated code
 - No *round-trip engineering*

Using UMPLE

Using UMPLE

- We will mostly be using
 - UMPLEonline
 - In a web browser: <http://try.UMPLE.org>
 - Or in Docker: <http://docker.UMPLE.org>
 - UMPLE on the command line: <http://dl.UMPLE.org>
 - Needs Java 8 JDK on the command line:
 - <http://bit.ly/1IO1FSV>
 - Java 9 works well too

Docker Container Experimental

```
mkdir ~/src && cd ~/src && git clone git@github.com:UMPLE/UMPLE.git
```

```
docker run -i -t -v `pwd`:/src UMPLE/UMPLE:0.4.0 bash
```

Using UMPLE

- Optional:
 - UMPLE in Eclipse
 - <https://github.com/UMPLE/UMPLE/wiki/InstallEclipsePlugin>
 - cmake and gcc for compiling C++ code

UMPLE Philosophy

UMPLE Philosophy 1-4

- P1. Modeling is programming and vice versa
- P2. An UMPLE programmer should never need to edit generated code to accomplish any task.
- P3. The UMPLE compiler can accept and generate code that uses nothing but UML abstractions.
 - The above is the inverse of the following
- P4. A program without UMPLE features can be compiled by an UMPLE compiler.
 - e.g. input Java results in the same as output

UMPLE Philosophy 5-8

- P5. A programmer can incrementally add UMPLE features to an existing program
 - Umplication
- P6. UMPLE extends the base language in a minimally invasive and safe way.
- P7. UMPLE features can be created and viewed diagrammatically or textually
- P8. UMPLE goes beyond UML

UML Class Modeling

UMPLE Class Models - Quick Overview

- Key elements:
 - Classes
 - Attributes
 - Associations
 - Generalizations
 - Methods
- We will look at all these using examples via UMPLE ONLINE
- UMPLE code/models are stored in files with suffix **.ump**

Exercise: Compiling and changing a model

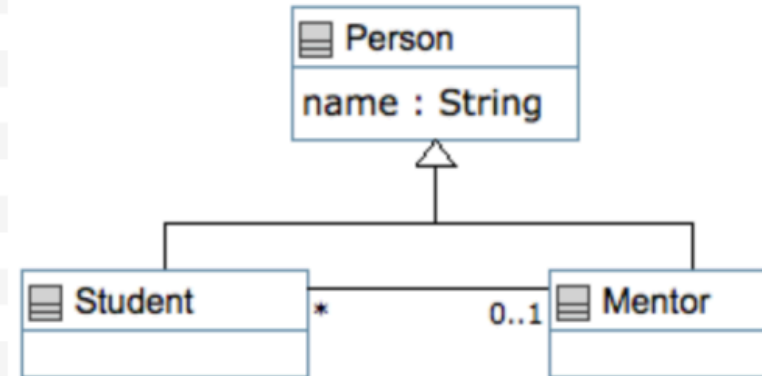
- Look at the example at the bottom of
 - <http://helloworld.UMPLE.org> (also on next slide)
 - Observe: attribute, association, class hierarchy, mixin
- Click on Load the above code into UMPLEOnline
 - Observe and modify the diagram
 - Add an attribute
 - Make a multiplicity error, then undo
 - Generate code and take a look
 - Download, compile and run if you want

Hello World Example 2 in the User Manual

```

10. class Person {
11.     name; // Attribute, string by default
12.     String toString () {
13.         return(getName());
14.     }
15. }
16.
17. class Student {
18.     isA Person;
19. }
20.
21. class Mentor {
22.     isA Person;
23. }
24.
25. association {
26.     0..1 Mentor -- * Student;
27. }
28.
29. class Person {
30.     // Notice that we are defining more contents for Person
31.     // This uses Uml's mixin capability
32.
33.     public static void main(String [ ] args) {
34.         Mentor m = new Mentor("Nick The Mentor");
35.         Student s = new Student("Tom The Student");
36.         s.setMentor(m);
37.         System.out.println("The mentor of " + s + " is " + s.getMentor());
38.         System.out.println("The students of " + m + " are " + m.getStudents());
39.     }


```



Key tools:

- UMPLE Online
- Command-Line
- User Manual

Hello World example 2 in UMPLEOnline



Draw on the right, write (Umple) model code on the left, analyse and generate code from models.
[Run in Docker for speed, or download](#) For help: [User manual](#) [Ask questions](#) [Report issue](#)

Line= 1
E G S T D A M
Generate Java
Create Bookmarkable URL

```

1  /*
2  * Introductory example of Umple showing classes,
3  * attribute, association, generalization, methods
4  * and the mixin capability. Generate java and run
5  * this.
6  * The output will be:
7  * The mentor of Tom The Student is Nick The
8  * Mentor
9  * The students of Nick The Mentor are [Tom The
10 * Student]
11 */
12 class Person {
13     name; // Attribute, string by default
14     String toString () {
15         return(getName());
16     }
17 }
18 class Student {
19     isA Person;
20 }
21 class Mentor {
22     isA Person;
23 }
24 association {
25     0..1 Mentor -- * Student;
26 }
27 
```

▶ SAVE & LOAD

▼ TOOLS

EXAMPLES

Class Diagrams ▾

Select Example ▾

DRAW

Class

Association

Generalization

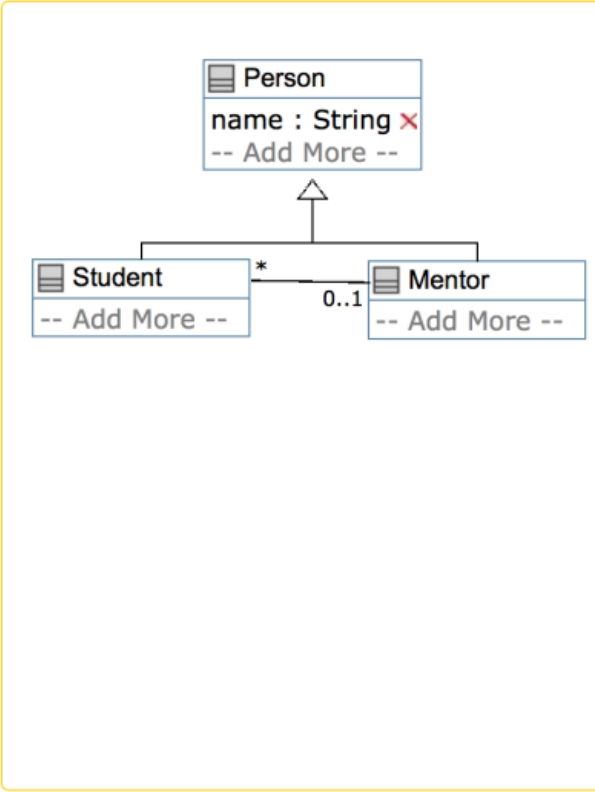
Delete

Undo

-Redo

-Sync-Diagram

▶ OPTIONS



```

classDiagram
    class Person {
        name : String
    }
    class Student {
    }
    class Mentor {
    }
    Person <|-- Student
    Person <|-- Mentor
    Student "*" -- "0..1" Mentor
    
```

Exploration of UMPLEOnline

- Explore class diagram examples
- Options
 - T or Control-t (hide and show text)
 - D or Control-d (hide and show diagram)
 - A, M to hide and show attributes, methods
 - Default diagram types
 - G / Control-g (Graphviz), S / Control-s (State Diagram)
 - E / Control-e (Editable class diagram)
- Generate code and look at the results
 - In UMPLE you never should modify generated code
 - It is designed to be readable for educational purposes

Use of the UMPLEOnline Docker image

- UMPLE's server can handle 80,000 transactions per hour
 - Code generations, edits
- But needs a good Internet connection
(sometimes hundreds of students have assignments due)
- To maximize speed of UMPLEOnline run it in your local machine:
 - Follow the instructions at <http://docker.UMPLE.org>

Demo of compiling on the command line

- To compile on the command line you will need Java 8
- Download UMPLE from <http://dl.UMPLE.org>
- Basic compilation

```
java -jar UMPLE.jar model.ump
```

- Help for features and commands

```
java -jar UMPLE.jar --help
```

- To generate and compile the java to a final system

```
java -jar UMPLE.jar model.ump -c -
```

Quick walkthrough of the user manual

- <http://manual.UMPLE.org>

Note in particular

- Key sections:
 - attributes,
 - associations,
 - state machines
- Grammar
- Generated API
- Errors and warnings
- Editing pages in github

UMPLE Attributes

- More than just variables
 - <http://attributes.UMPLE.org>

Attributes

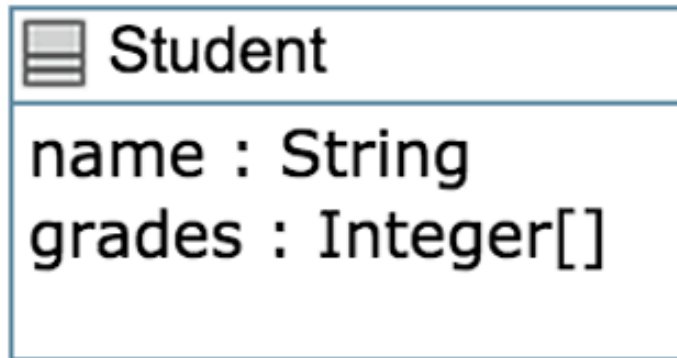
Group
i : Integer
str : String
s : String
d : Date
t2 : Time
q : String
p : String

```
1 class Group
2 {
3     Integer i;
4     const Integer max = 100;
5     immutable String str;
6     lazy s;
7     settable Date d;
8     internal Time t2 = new Time(System.currentTimeMillis());
9     String q = "chicken";
10    defaulted p = "robot";
11 }
12
```

[Show/Hide errors and warnings](#)

Warning on [line 4](#) : Constant name 'max' should start with a upper-case letter. [More information \(161\)](#)

Attributes Exercise #1



```
1  class Student
2  {
3      name;
4      Integer[] grades;
5  }
6  |
```


Attributes

- "*Instance variables*"
 - Part of the state of an object
 - Simple data that will always be present in each instance
- Specified like a Java or C++ field or member variable
- But, intended to be more abstract!
 - **Example**, with an initial value

```
a = "init value";
```

Attributes

- As in UML, more abstract than instance variables
 - Always private by default
 - Should only be accessed get, set methods
 - Can be stereotyped (upcoming slides) to affect code generation
 - Can have aspects applied (discussed later)
 - Can be constrained (discussed later)

Code generation from attributes

- Default code generation
 - Generates a `getName()` and `setName()` method for `name`
 - `public`
- Creates an arguments in the class constructor by default
- An attribute is `private` to the class by default
 - *Should only be accessed get, set methods*

Code Generation (JavaDocs)

GENERATE

- Java Code
- Java API Doc
- PHP Code
- C++ Code
- Ruby Code
- Ecore
- TextUml
- Sevml (Experimental)

All Classes

Student

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type		Method and Description
boolean		<code>addGrade(java.lang.Integer aGrade)</code>
void		<code>delete()</code>
java.lang.Integer		<code>getGrade(int index)</code>
java.lang.Integer[]		<code>getGrades()</code>
java.lang.String		<code>getName()</code>
boolean		<code>hasGrades()</code>
int		<code>indexOfGrade(java.lang.Integer aGrade)</code>
int		<code>numberOfGrades()</code>
boolean		<code>removeGrade(java.lang.Integer aGrade)</code>
boolean		<code>setName(java.lang.String aName)</code>
java.lang.String		<code>toString()</code>

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Code Generation Patterns

- Attributes
 - Set/Get (UB = 1)
 - Add/Remove/NumberOf/IndexOf/Get (UB > 1)
 - Lazy immutability
 - Default values
 - Constants
 - Before / After cod

UB = upper bound

Code Generation Patterns

- Associations
 - Set/Get (UB = 1)
 - Add/Remove/NumberOf/IndexOf/Get (UB > 1)
 - Referential Integrity
 - Multiplicity Constraints
 - 42 different cases

UB = upper bound

Code Generation (Semantics)

- <http://api.UMPLE.org/>

The screenshot shows the 'API Summary' page on api.umples.org. The page title is 'Umples User Manual' with links for '[Previous]' and '[Next]'. A search bar is present. The main content is titled 'API Summary' and explains that the following table describes the API generated from various Umples features. It lists 'API GENERATED IN ALL CASES' and 'API GENERATED FROM ATTRIBUTES'. A table summarizes the API for array attributes, showing the Umples concept, notation, and querying value for different attribute types (basic, initialized, lazy).

Umples concept	basic	initialized	lazy
Umples notation	T z;	T z = val;	lazy
Querying value			
T getZ()	return the value	yes	yes
boolean isZ()	return true if value is true	if T Boolean	if T Boolean
boolean equals(Object)	is equal to another?		if T B

UMPLE builtin datatypes

```
String // (default if none specified)
Integer
Float
Double
Boolean
Time
Date
```

- The above will generate appropriate code in Java, C++ etc.
 - e.g. Integer becomes int
- Other (native) types can be used but without guaranteed correctness

Attribute stereotypes (1)

- Code generation can be controlled through stereotypes:
 - lazy - **don't add a constructor argument**

```
lazy b; // sets it to null, 0, "" depending on type
```

- Defaulted – *can be reset*

```
defaulted s = "def"; // resettable to the default
```

Attribute stereotypes (2)

- autounique – provide a unique value to each instance

```
autounique x; // sets attribute to 1, 2, 3 ...
```

- internal – don't generate any methods

```
internal i; // doesn't generate any get/set either
```

Immutability

- Useful for objects where you want to guarantee no possible change once created
 - e.g. a geometric point
- Generate a constructor argument and get method but no set method

```
immutable String str;
```

- No constructor argument, but allows setting just once.

```
lazy immutable z;
```

Lets explore attributes by example

- Go to
 - <http://attributes.UMPLE.org>

Derived attributes

- These generate a get method that is calculated.

```
class Point
{
// Cartesian coordinates
Float x;
Float y;

// Polar coordinates
Float rho =
{Math.sqrt(Math.pow(getX(), 2) + Math.pow(getY(), 2))}
Float theta =
{Math.toDegrees(Math.atan2(getY(),getX()))}

}
```

Multi-valued attributes

- Limit their use. Associations are generally better.

```
class Office {  
Integer number;  
Phone[] installedTelephones;  
}
```

```
class Phone {  
String digits;  
String callerID;  
}
```

Keys

- Enable UMPLE to generate an `equals()` and a `hashCode()` method

```
class Student {  
  Integer id;  
  name;  
  key { id }  
}
```

- The user manual has a sports team example showing keys on associations too
- Note how this feature is not inherited from UML

UMPLE Generalization and interfaces

Generalization in UMPLE

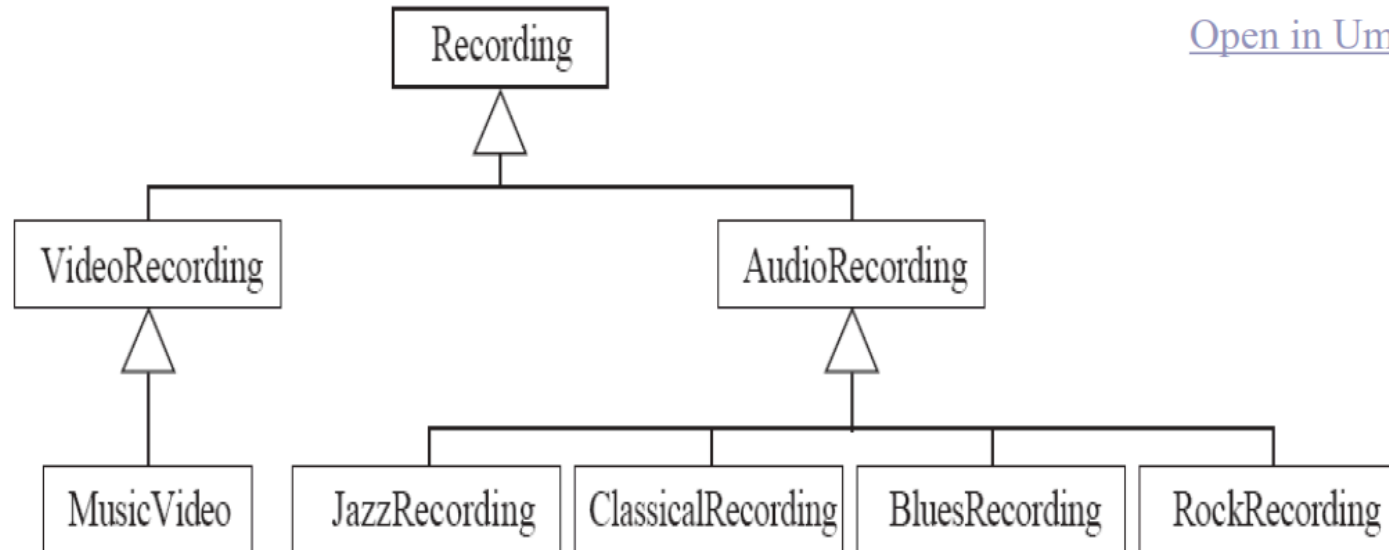
- UMPLE uses the `isA` keyword to indicate generalization
- Used to indicate `superclass`, used `trait`, implemented `interface`

```
class Shape {  
  colour;  
}  
class Rectangle {  
  isA Shape;  
}
```

Avoiding unnecessary generalizations

Open in UMPLE

- Inappropriate hierarchy of Classes
- What should the model be?



[Open in Umples](#)

Interfaces

- Declare signatures of a group of methods that must be implemented by various classes
- Also declared using the keyword `isA`
- Essentially the same concept as in Java
- *Let's explore examples in the user manual ...*

UMPLE Methods

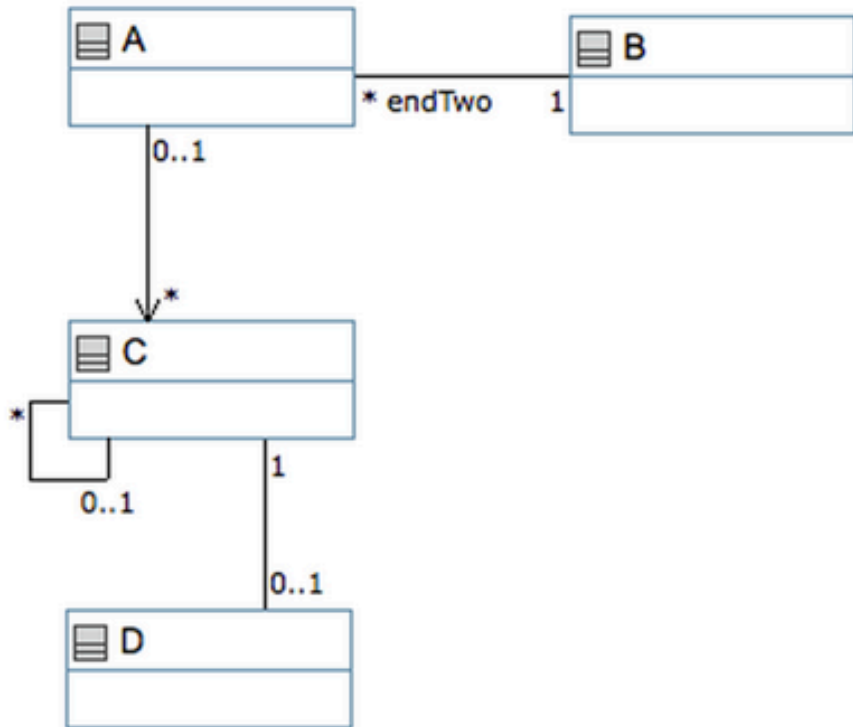
User-written **Methods** in UMPLE

- Methods can be added to any UMPLE code.
- UMPLE parses the signature only; the rest is passed to the generated code.
- You can specify different bodies in different languages
- *We will look at examples in the user manual ...*

UMPLE Associations

- <http://associations.UMPLE.org>
 - Notice the inline and independent state machines

Associations

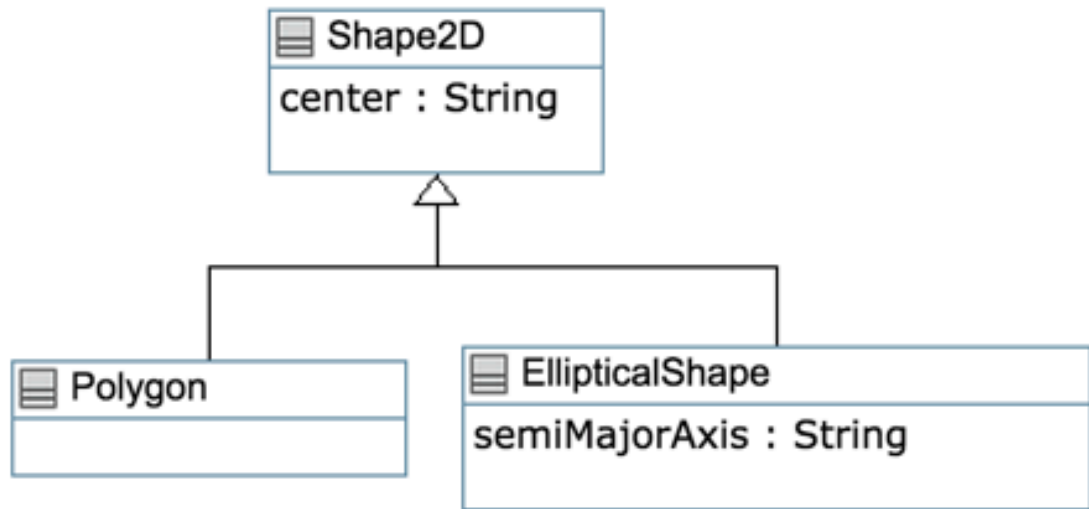


<http://associations.umple.org/>

```

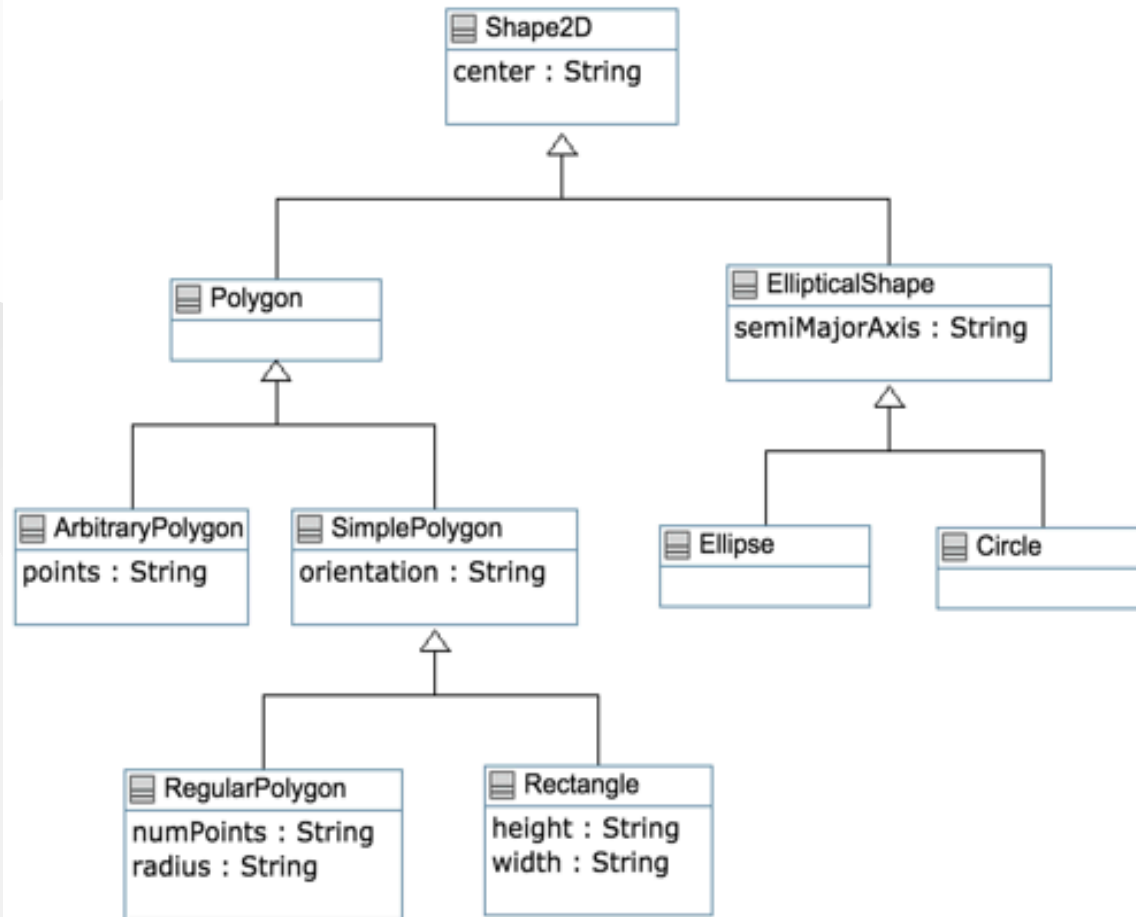
1 class A {}
2
3 // Class with inline association having role name
4 class B {
5     1 -- * A endTwo;
6 }
7
8 // Class with reflexive association
9 class C {
10    0..1 -- * C;
11    1 -- 0..1 D; // D is external
12 }
13
14 // Independently defined and directed association
15 association {
16     0..1 A -> * C;
17 }
18
19 // Class with composition
20 class E {
21     0..1 e @> * A a;
22 }
23
24 // Reference to a class defined elsewhere
25 external D {}
26
    
```

Associations Exercise #1



```
1 class Shape2D {
2     center;
3 }
4
5 class EllipticalShape {
6     isA Shape2D;
7     semiMajorAxis;
8 }
9
10 class Polygon {
11     isA Shape2D;
12 }
```


Associations Exercise #2



```

1 class Shape2D {
2   center;
3 }
4
5 class EllipticalShape {
6   isA Shape2D;
7   semiMajorAxis;
8 }
9
10 class Polygon {
11   isA Shape2D;
12 }
13
14 class Circle {
15   isA EllipticalShape;
16 }
17
18 class Ellipse{
19   isA EllipticalShape;
20 }
21
22
23 class SimplePolygon {
24   orientation;
25   isA Polygon;
26 }
27
28 class ArbitraryPolygon {
29   points;
30   isA Polygon;
31 }
32
33 class Rectangle {
34   isA SimplePolygon;
35   height;
36   width;
37 }
38
39 class RegularPolygon {
40   numPoints;
41   radius;
42   isA SimplePolygon;
43 }
44
45 
```

Associations

- Describe how instances of classes are linked at runtime
 - Bidirectional `--` or
 - Unidirectional `->`
- Multiplicity:
 - Bounds on the number of linked instances
- `*` Or `0..*` \longrightarrow 0 or more
- `1..*` \longrightarrow 1 or more
- `1` \longrightarrow Exactly 1
- `2` \longrightarrow Exactly 2
- `1..3` \longrightarrow Between 1 and 3
- `0..2` \longrightarrow Up to 2

Association Relationships

0..1	0..n	*	1	n	m..n	m...*
0..1 -- 0..1						
0..1 -- 0..n	0..n -- 0..n					
0..1 -- *	0..n -- *	* -- *				
0..1 -- 1	0..n -- 1	* -- 1	1 -- 1			
0..1 -- n	0..n -- n	* -- n	1 -- n	n -- n		
0..1 -- m..n	0..n -- m..n	* -- m..n	1 -- m..n	n -- m..n	m..n -- m..n	
0..1 -- m..*	0..n -- m..*	* -- m..*	1 -- m..*	n -- m..*	m..n -- m..*	m..* -- m..*

Association Relationships

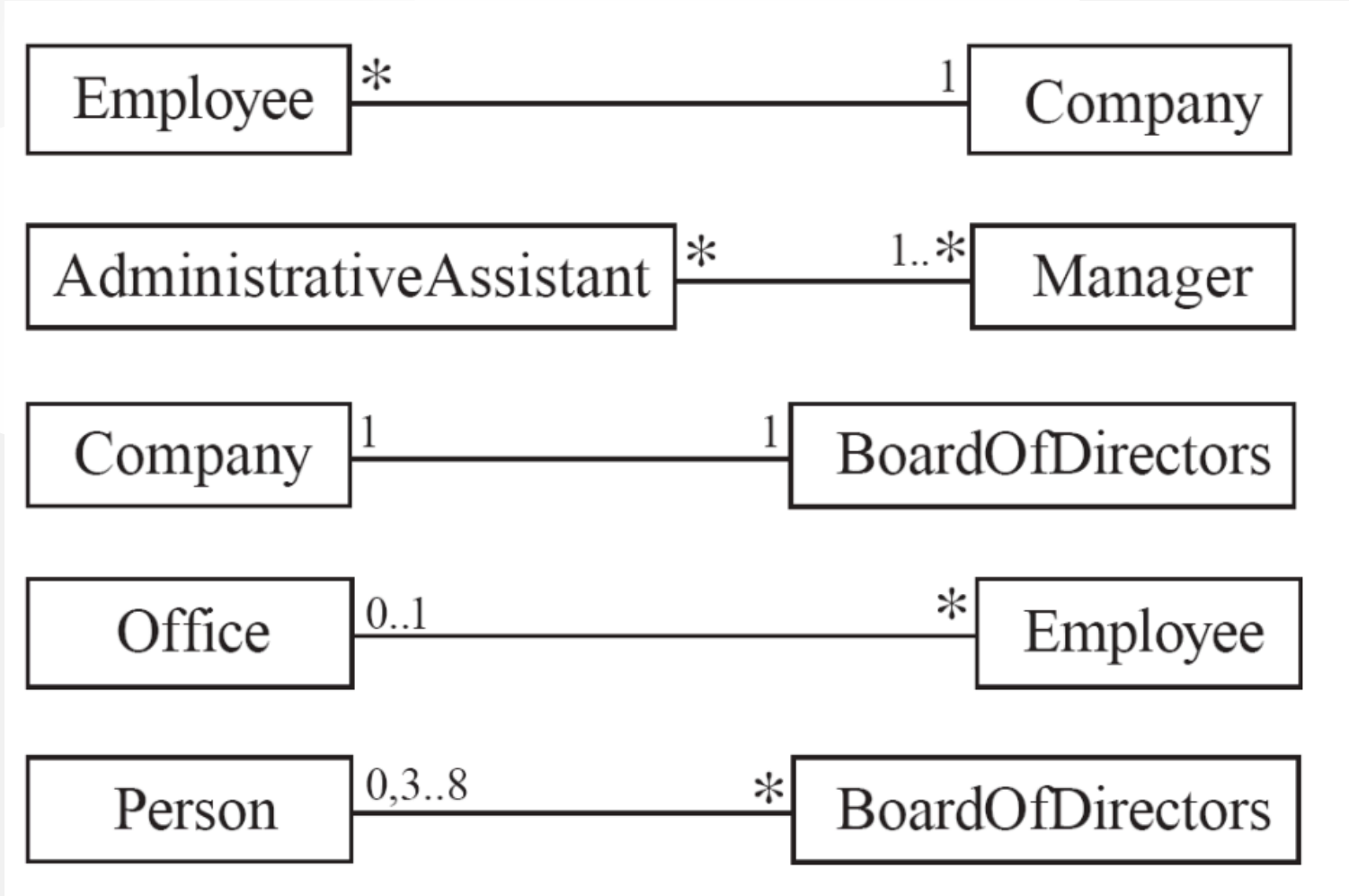
- Directional Associations

$* \rightarrow 0..1$, $* \rightarrow 1$, $* \rightarrow *$, $* \rightarrow m..n$, $* \rightarrow n$, $* \rightarrow m..*$ and $* \rightarrow 0..n$.

- Symmetric Reflexive

$0..1$, $0..n$, $*$, 1 , n , $m..n$, $m..*$

Basic UML associations

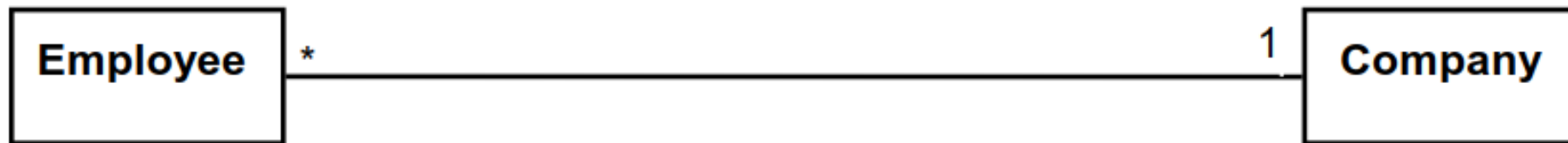


Many-to-one associations (1)

```
class Employee {  
  id;  
  firstName;  
  lastName;  
}  
  
class Company {  
  name;  
  1 -- * Employee;  
}
```

Many-to-one associations (2)

- A company has many employees,
- An employee can only work for one company.
 - This company will not store data about the moonlighting activities of employees!
- A company can have zero employees
 - E.g. a 'shell' company
- It is not possible to be an employee unless you work for a company
- Let's draw and write this in UMPLEOnline:



Role names (optional, in most cases)

- Allow you to better label either end of an association

```
class Person{
id;
firstName;
lastName;
}

class Company {
name;
1 employer -- * Person employee;
}
```

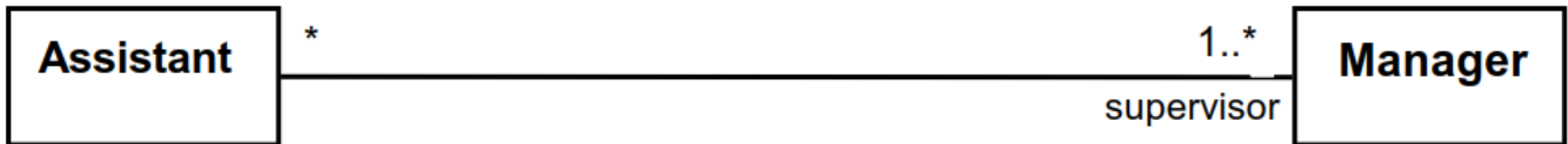

Referential Integrity

- When an instance on one side of the association changes
 - The linked instances on the other side know ...
 - And vice-versa
- This is standard in UMPLE associations, which are bidirectional

Many-to-Many Associations

- An assistant can work for many managers
- A manager can have many assistants
- Assistants can work in pools working for several managers
- Managers can have a group of assistants
- Some managers might have zero assistants.
- Is it possible for an assistant to have, perhaps temporarily, zero managers?

[Open in UMPLE](#)



One-to-One Associations (Use cautiously)

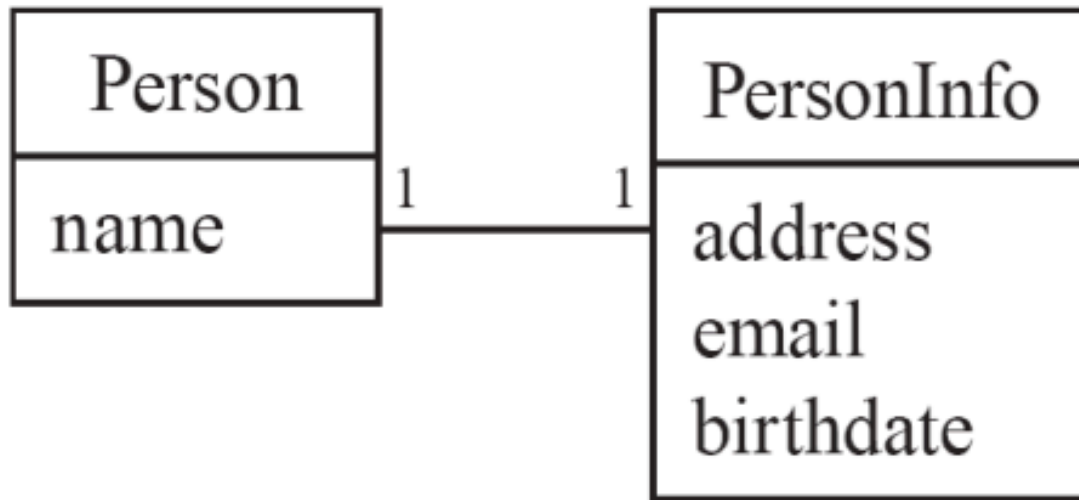
- For each company, there is exactly one board of directors
- A board is the board of only one company
- A company must always have a board
- A board must always be of some company

[Open in UMPLE](#)

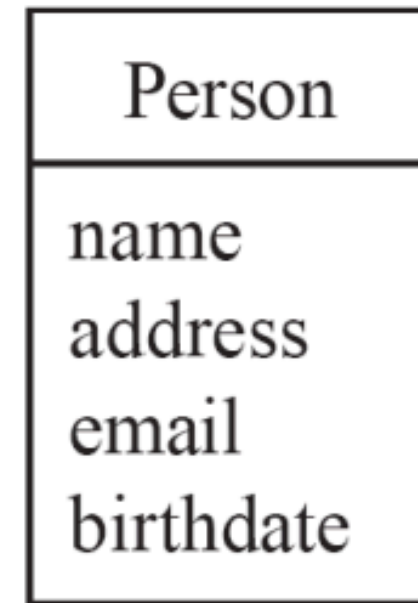


Typical erroneous use of one-to-one

Avoid this



do this

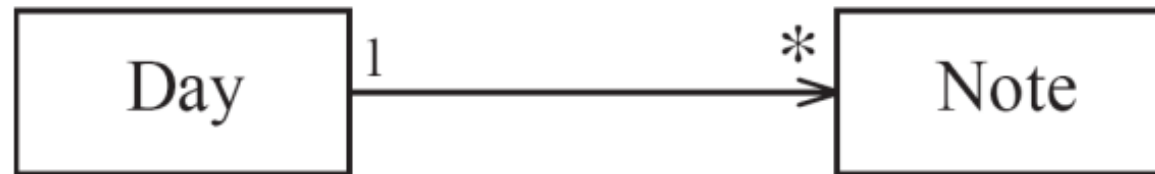


Unidirectional Associations

- Associations are by default bi-directional
- It is possible to limit the direction of an association by adding an arrow at one end
- In the following unidirectional association
 - A Day knows about its notes, but a Note does not know which Day it belongs to
 - Note remains 'uncoupled' and can be used in other contexts

```
class Day {  
  * -> 1 Note;  
}  
class Note {}
```

[Open in UMPLE](#)



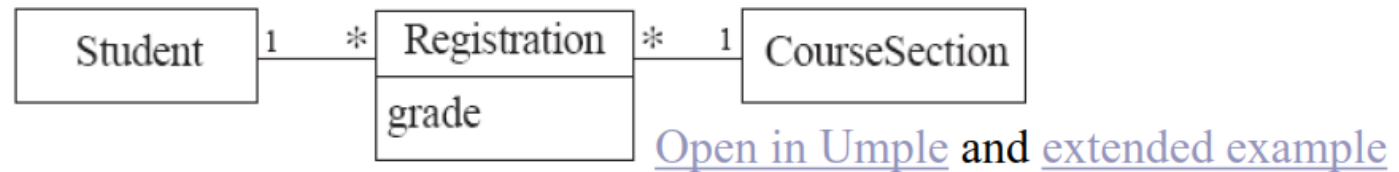
Association Classes

CEN206 Object-Oriented Programming (formerly CE204)

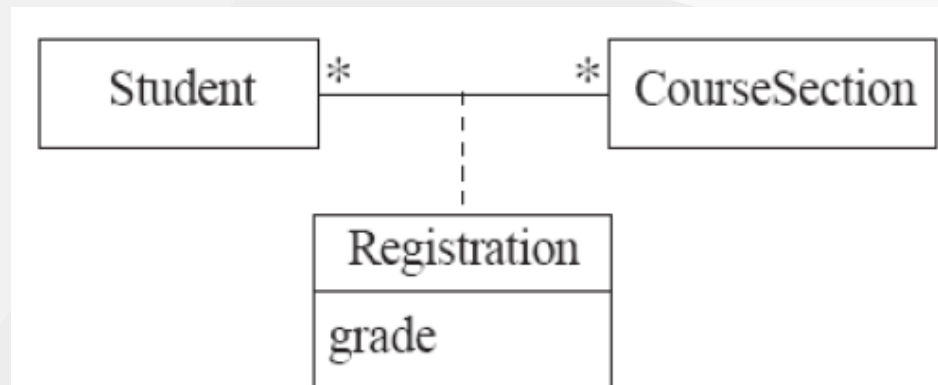
- Sometimes, an attribute that concerns two associated classes cannot be placed in either of the classes

Open in UMPLE

and [Extended Example](#)



- The following are nearly equivalent
 - The only difference:
 - in the association class there can be only a single registration of a given Student in a CourseSection



Association classes (cont.)

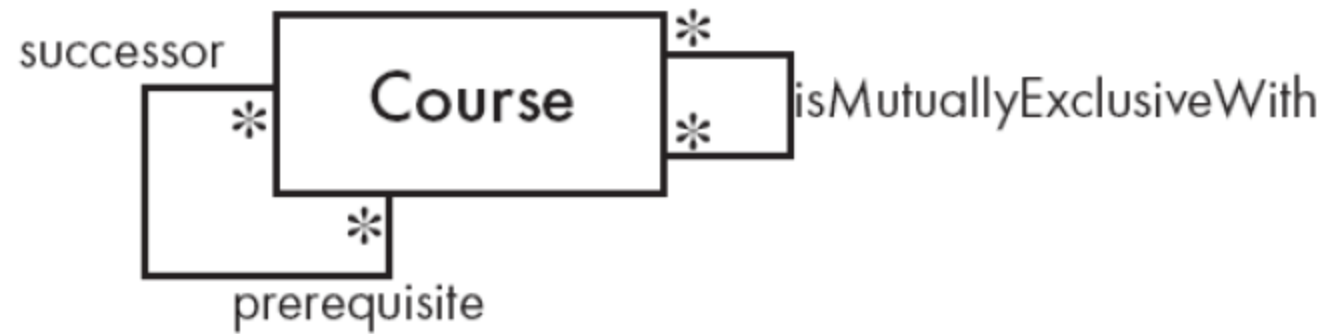
- UMPLE code

```
class Student {}  
class CourseSection {}  
associationClass Registration {  
* Student;  
* CourseSection;  
}
```

- Open in UMPLEOnline, and then generate code

Reflexive Associations

- An association that connects a class to itself



Open in UMPLE

```
class Course {  
* self isMutuallyExclusiveWith; // Symmetric  
}  
  
association {  
* Course successor -- * Course prerequisite;  
}
```


Inline vs. Standalone associations

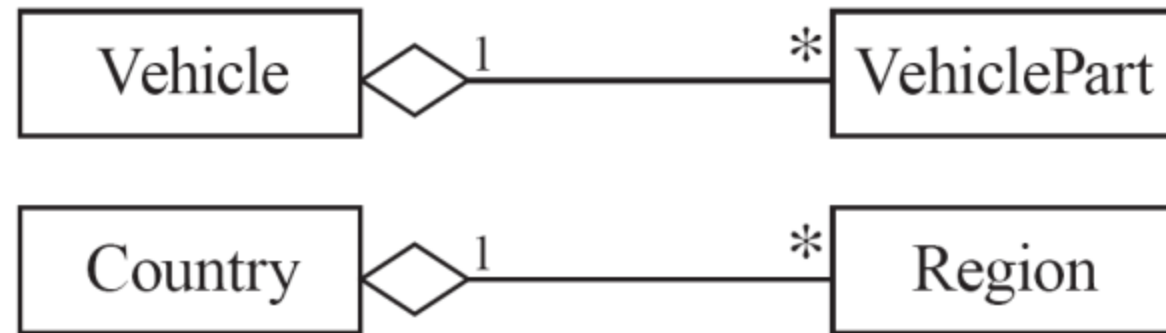
- The following are equivalent to allow flexibility:

```
class X {}  
class Y {  
1 -- * X;  
  
}
```

```
class X {}  
class Y {}  
association {  
1 Y -- * X;  
  
}
```

Aggregation

- Aggregations are ordinary associations that represent part-whole relationships.
 - The 'whole' side is often called the assembly or the aggregate
 - This is a shorthand for association named `isPartOf`
 - UMPLE has no special syntax currently



```
class Vehicle {
1 whole -- * VehiclePart part;
}
class VehiclePart{
}
```

Composition

- A composition is a strong kind of aggregation
 - If the aggregate is destroyed, then the parts are destroyed as well



```
class Building {
1 <@>- * Room;
}
class Room{
}
```

Sorted Associations

- Order objects in the association according to a specific key

```
class Academy {  
1 -- * Student registrants sorted {id};  
}  
  
class Student {  
Integer id;  
name;  
}
```

- We will look at a more complete example in the User Manual

A final word on associations

- More help and examples are in the user manual online at
 - <http://associations.UMPLE.org>

References

- [UMPLE Tutorials](#)
- [UMPLE Github](#)
- [UMPLE Online](#)
- [UMPLE Documentation](#)
- [UMPLE CSI5112– February 2018](#)
- [UMPLE Tutorial: Models 2020 Web](#)
- [UMPLE Tutorial: Models 2020 Pdf](#)

References

- [Getting Started in UMPLE](#)
- [Experiential Learning for Software Engineering Using Agile Modeling in UMPLE \(Youtube\)](#)
- [Experiential Learning for Software Engineering Using Agile Modeling in UMPLE \(Slide\)](#)
- [Tomassetti Code Generation](#)

End – Of – Week – 6