# 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: Si&mple&, A&mple&, &UM&L &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: Si&mple&, A&mple&, &UM&L &P&rogramming &L&anguag&e&

- 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/1lO1FSV
        - 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
    - Umplification
- 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

## UMPLE Class Modeling

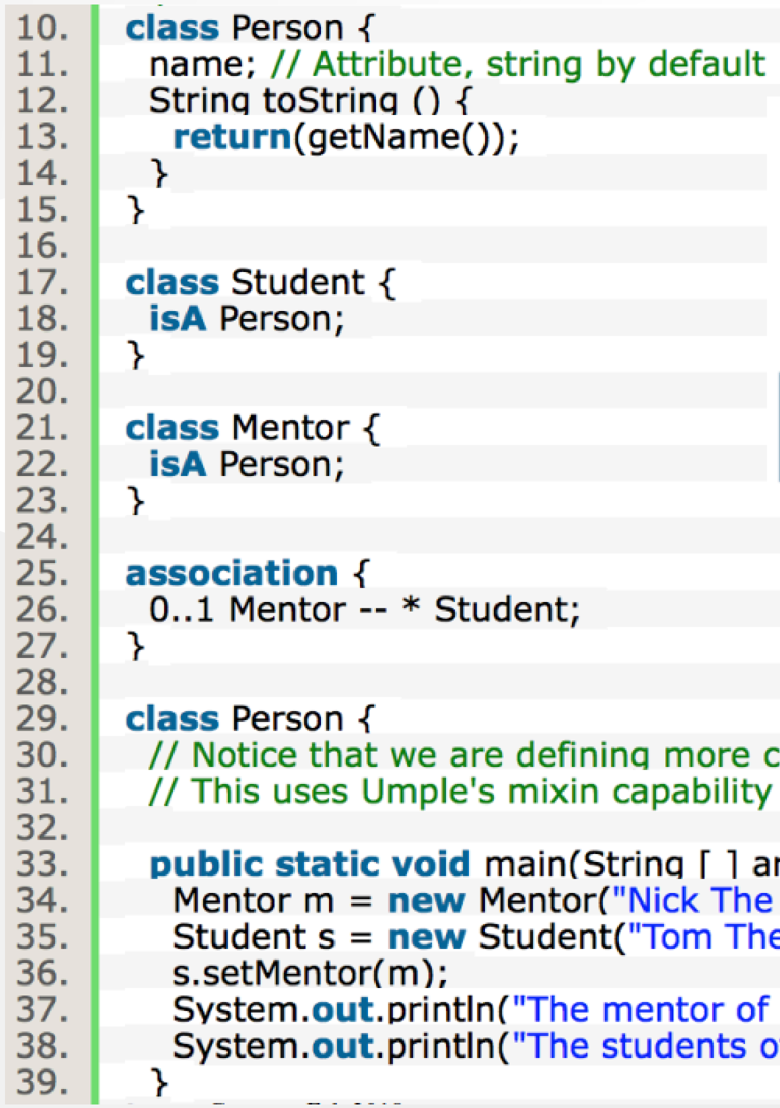
## UMPLE Class Models - Quick Overview

- 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**

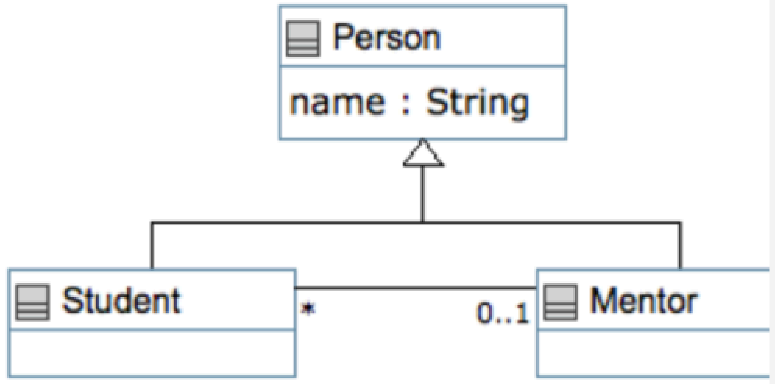
## Exercise: Compiling and changing a model

- 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



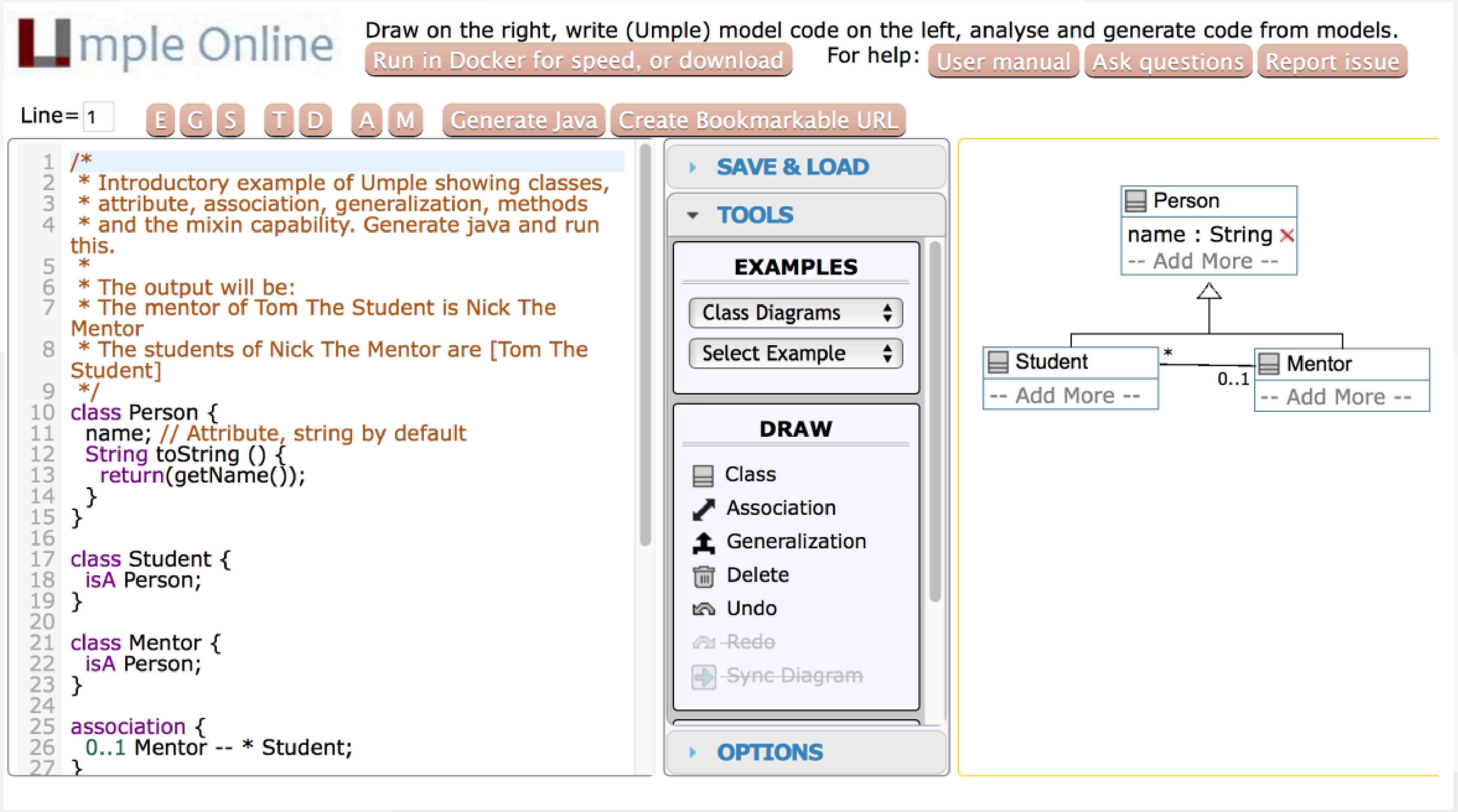
```
class Person {
  name; // Attribute, string by default
  String toString () {
    return(getName());
  }
}

class Student {
  isA Person;
}

class Mentor {
  isA Person;
}

association {
  0..1 Mentor -- * Student;
}

class Person {
  // Notice that we are defining more contents for Person
  // This uses Umple's mixin capability

  public static void main(String [ ] args) {
    Mentor m = new Mentor("Nick The Mentor");
    Student s = new Student("Tom The Student");
    s.setMentor(m);
    System.out.println("The mentor of "  + s  + " is " +  s.getMentor());
    System.out.println("The students of " +  m  + " are " +  m.getStudents()
  }
```



## Key tools:

- UMPLE Online
- Command-Line
- User Manual

## Hello World example 2 in UMPLEOnline


Umple Online
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
SAVE & LOAD
TOOLS
EXAMPLES
Class Diagrams
Select Example
DRAW
Class
Association
Generalization
Delete
Undo
~~Redo~~
~~Sync Diagram~~
OPTIONS
Person
name : String
-- Add More --
Student
-- Add More --
Mentor
-- Add More --
*
0..1

```
/*
 * Introductory example of Umple showing classes,
 * attribute, association, generalization, methods
 * and the mixin capability. Generate java and run
this.
 *
 * The output will be:
 * The mentor of Tom The Student is Nick The
Mentor
 * The students of Nick The Mentor are [Tom The
Student]
 */
class Person {
  name; // Attribute, string by default
  String toString () {
    return(getName());
  }
}

class Student {
  isA Person;
}

class Mentor {
  isA Person;
}

association {
  0..1 Mentor -- * Student;
}
```

## 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

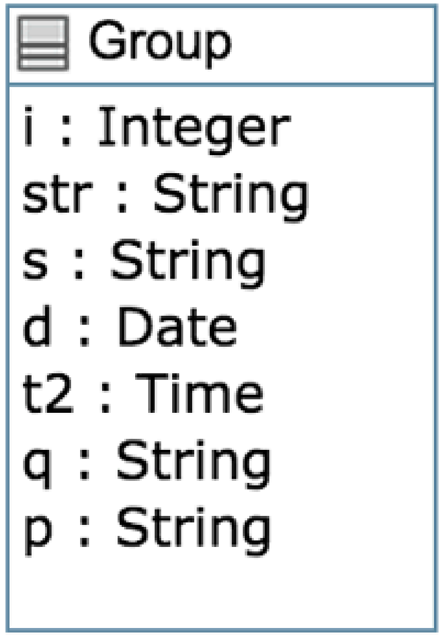
## Note in particular

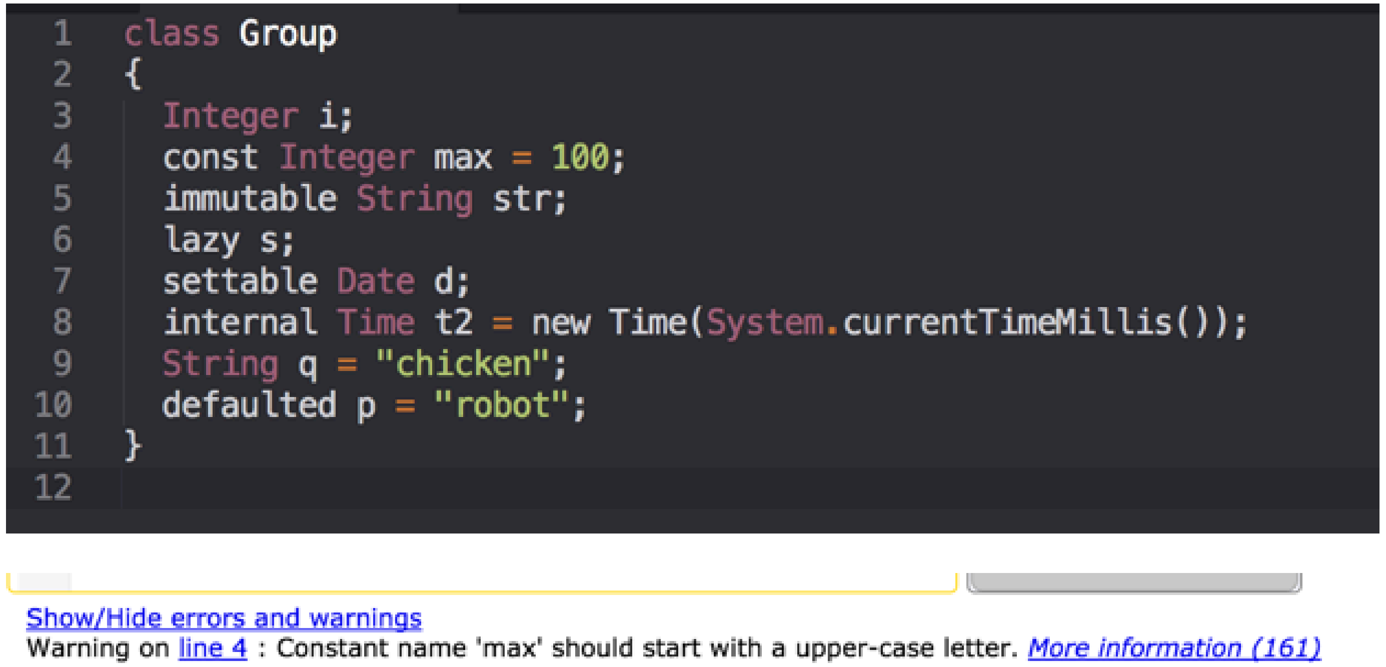
- Key sections:
  - attributes,
  - associations,
  - state machines
- Grammar
- Generated API
- Errors and warnings
- Editing pages in github

## UMPLE Attributes

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

## Attributes

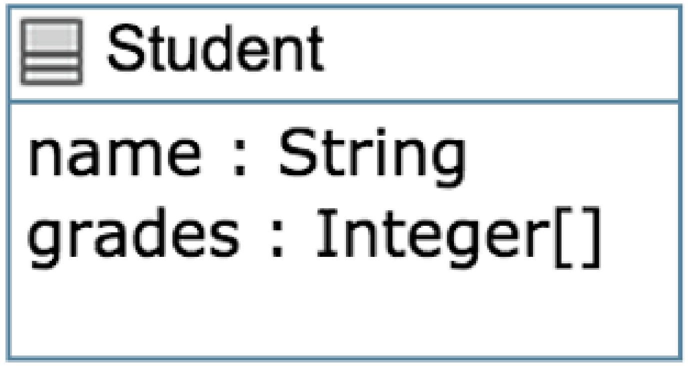


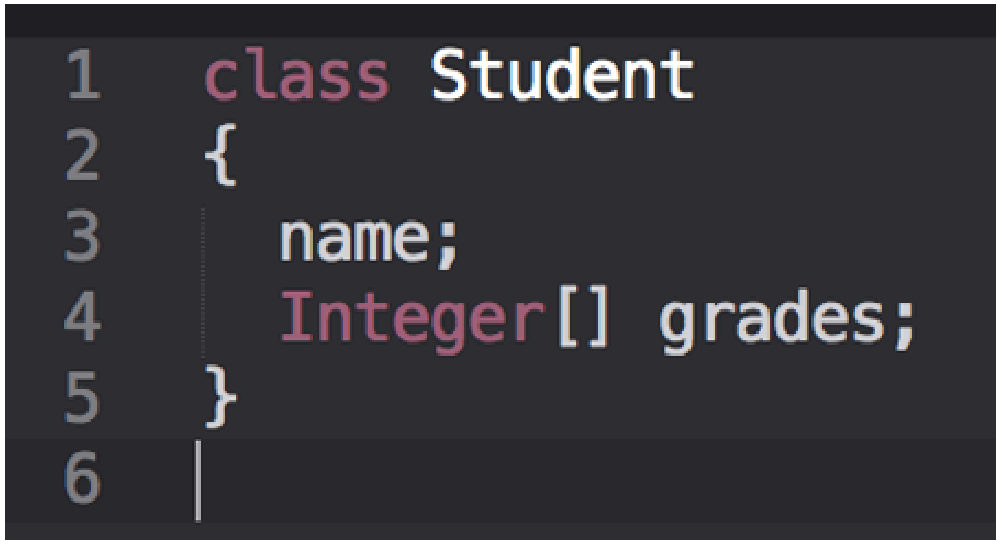


```
class Group
{
  Integer i;
  const Integer max = 100;
  immutable String str;
  lazy s;
  settable Date d;
  internal Time t2 = new Time(System.currentTimeMillis());
  String q = "chicken";
  defaulted p = "robot";
}

```

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





```
class Student
{
   name;
   Integer[] grades;
}

```

## 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";
```

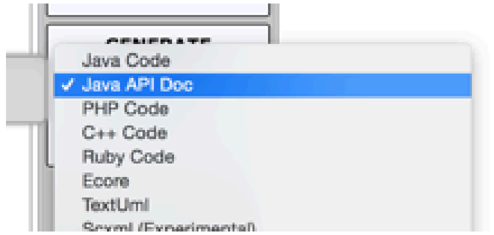
## Attributes

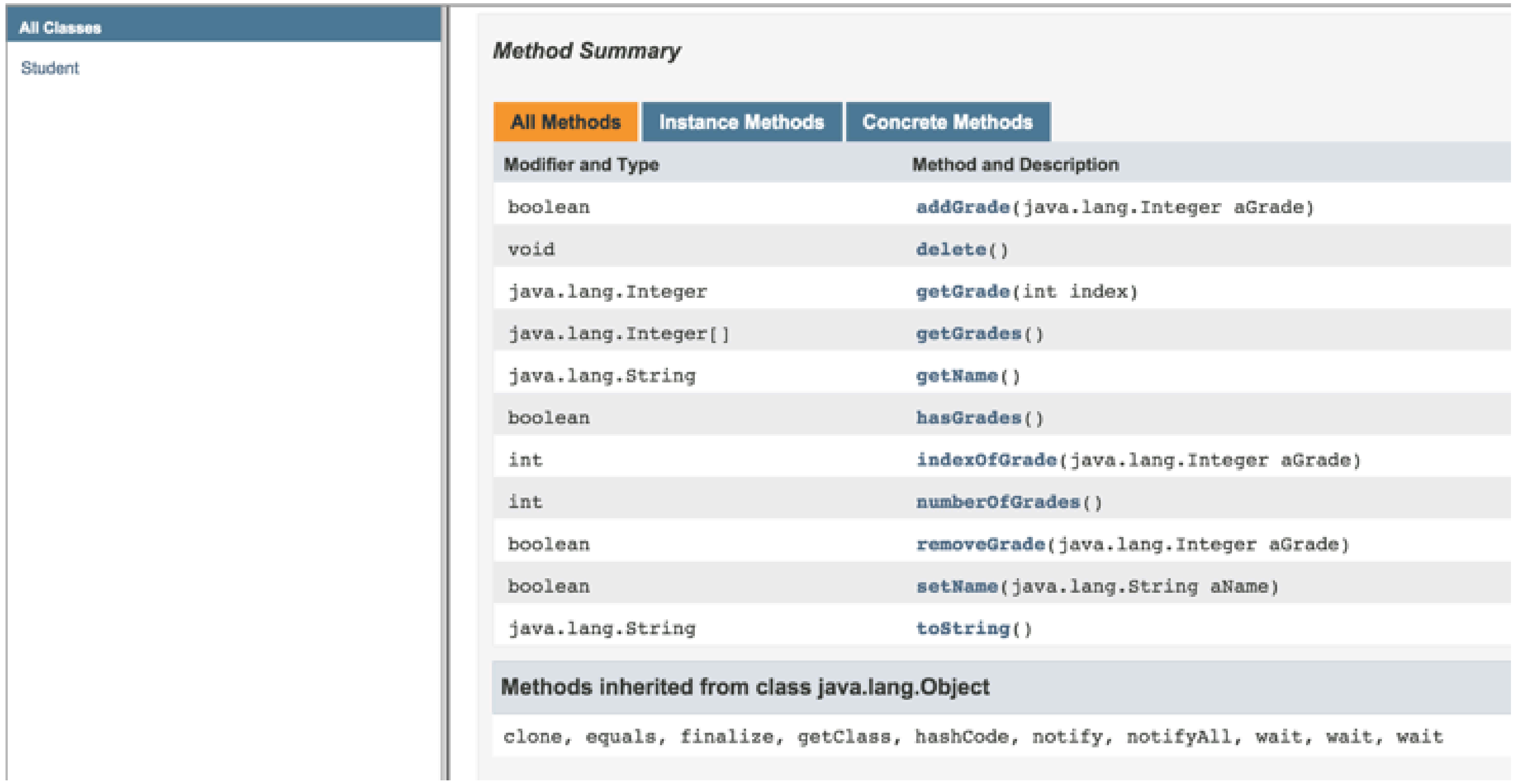
- As in UML, more abstract than instance variables
    - Always private by default
    - Should only be accessed get, set methods
    - Can be stereotyped (upcoming slides) to affect code generation
    - Can have aspects applied (discussed later)
    - Can be constrained (discussed later)

## 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)





All Classes

Student

*Method Summary*

All Methods | Instance Methods | Concrete Methods

| Modifier and Type | Method and Description |
|---|---|
| boolean | `addGrade(java.lang.Integer aGrade)` |
| void | `delete()` |
| java.lang.Integer | `getGrade(int index)` |
| java.lang.Integer[] | `getGrades()` |
| java.lang.String | `getName()` |
| boolean | `hasGrades()` |
| int | `indexOfGrade(java.lang.Integer aGrade)` |
| int | `numberOfGrades()` |
| boolean | `removeGrade(java.lang.Integer aGrade)` |
| boolean | `setName(java.lang.String aName)` |
| java.lang.String | `toString()` |

**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

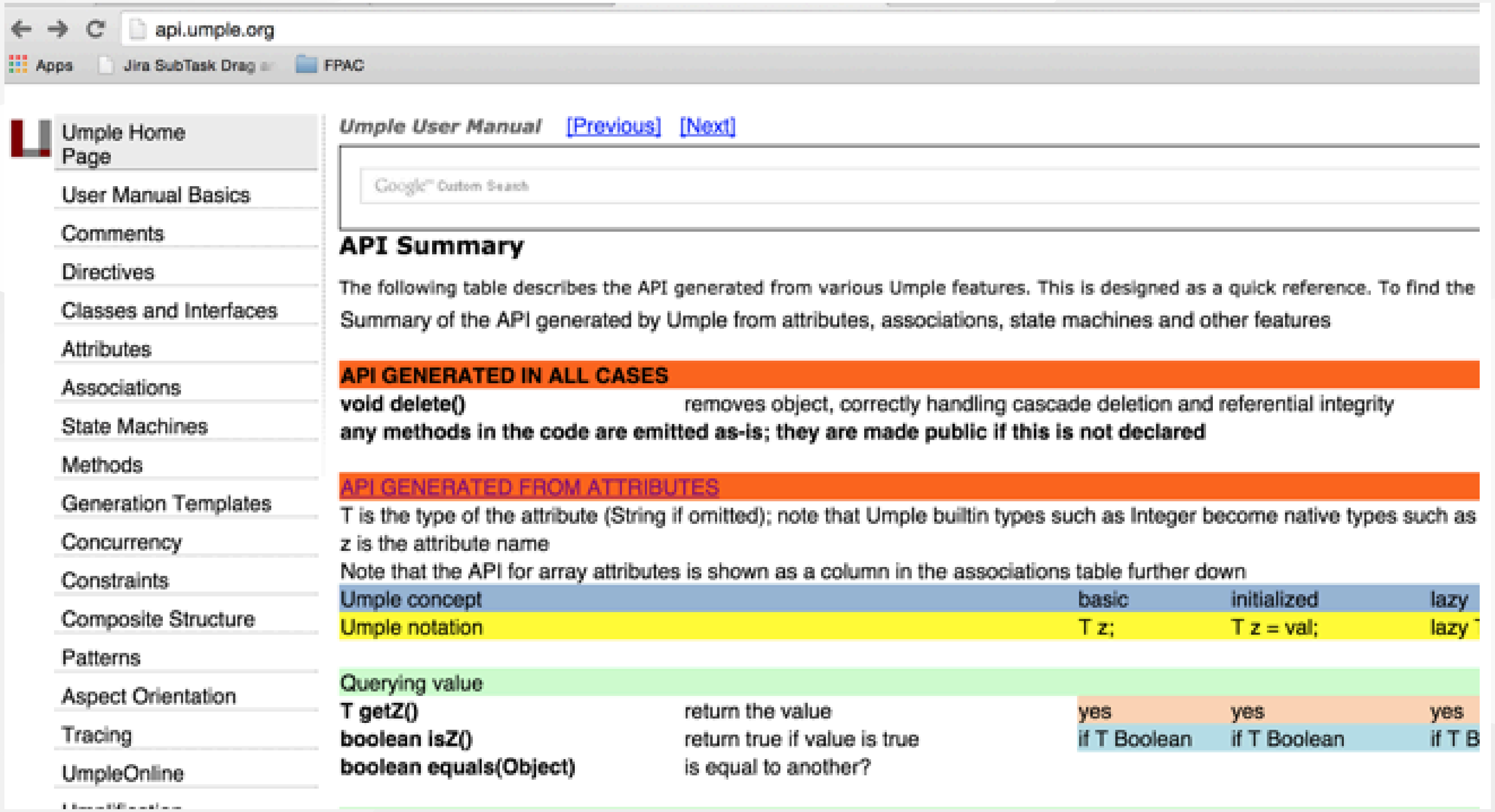
## Code Generation Patterns

- 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/



api.umple.org

Apps | Jira SubTask Drag | FPAC

Umple Home Page
User Manual Basics
Comments
Directives
Classes and Interfaces
Attributes
Associations
State Machines
Methods
Generation Templates
Concurrency
Constraints
Composite Structure
Patterns
Aspect Orientation
Tracing
UmpleOnline

*Umple User Manual* [Previous] [Next]

Google Custom Search

**API Summary**

The following table describes the API generated from various Umple features. This is designed as a quick reference. To find the

Summary of the API generated by Umple from attributes, associations, state machines and other features

**API GENERATED IN ALL CASES**

| | | | | |
|---|---|---|---|---|
| **void delete()** | removes object, correctly handling cascade deletion and referential integrity | | | |
| **any methods in the code are emitted as-is; they are made public if this is not declared** | | | | |

API GENERATED FROM ATTRIBUTES

T is the type of the attribute (String if omitted); note that Umple builtin types such as Integer become native types such as

z is the attribute name

Note that the API for array attributes is shown as a column in the associations table further down

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

## 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

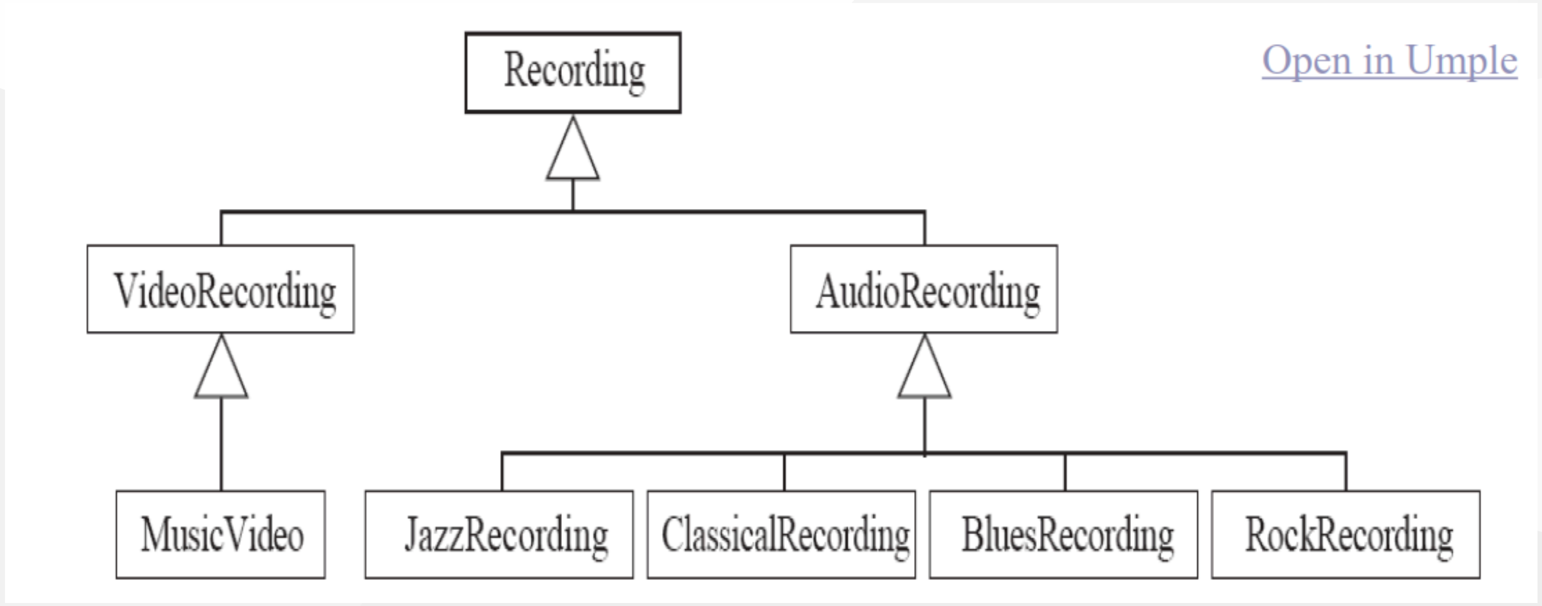
- UMPLE uses the `isA` keyword to indicate generalization
- Used to indicate `superclass`, used `trait`, implemented `interface`

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

## Avoiding unnecessary generalizations

Open in UMPLE

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



## 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

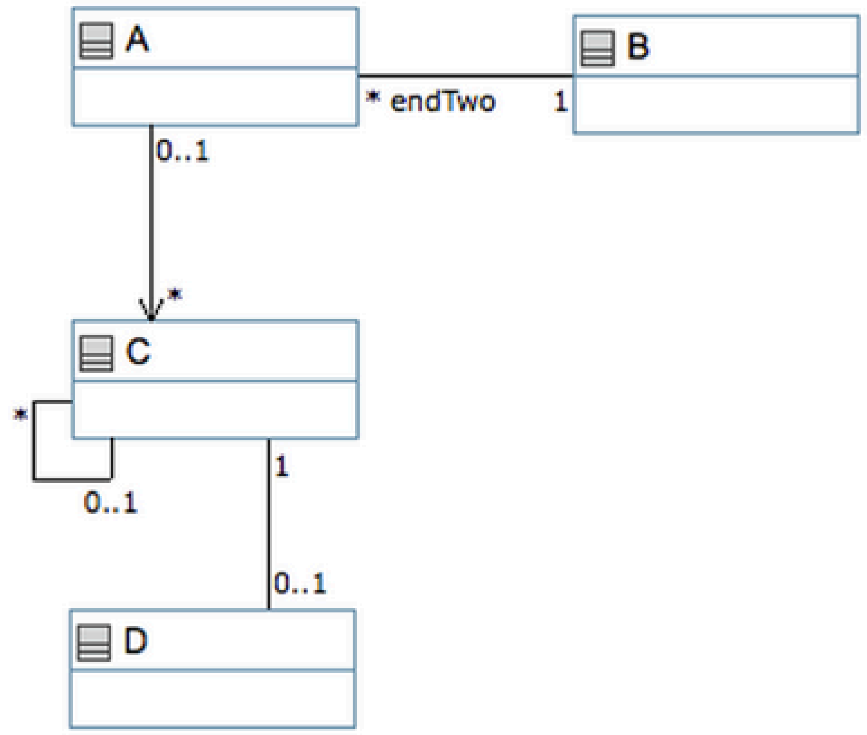
## User-written Methods in UMPLE

- 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 …*

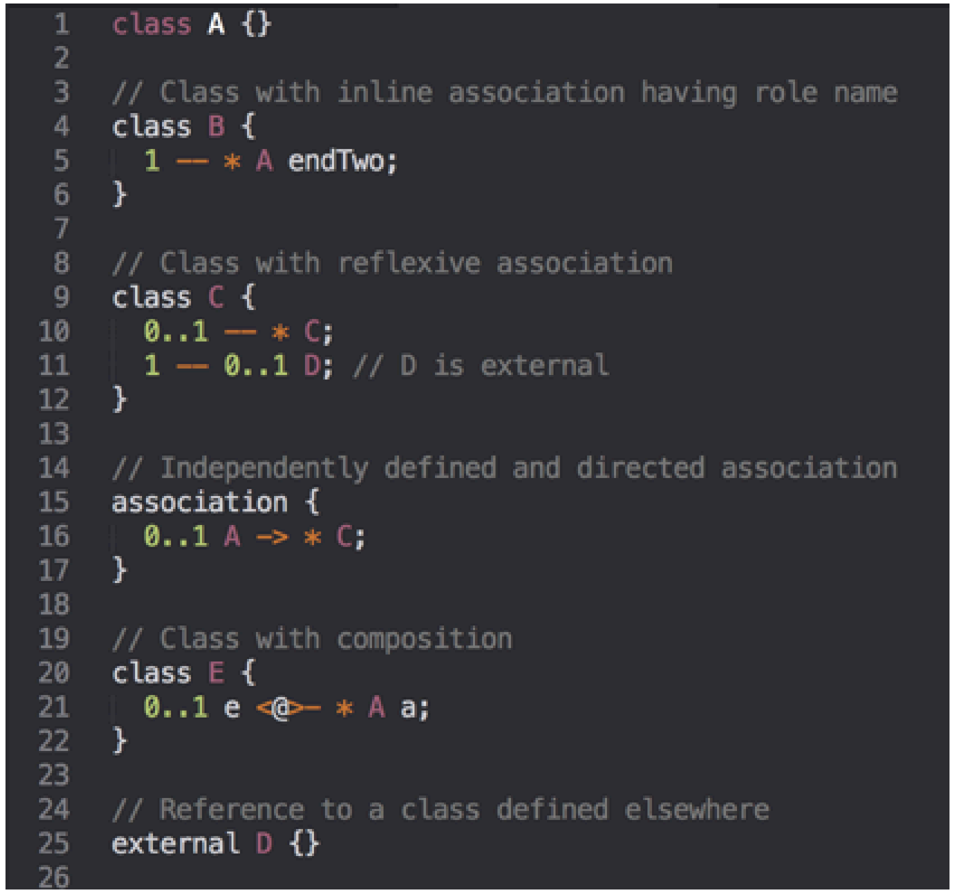
## UMPLE Associations

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

## Associations

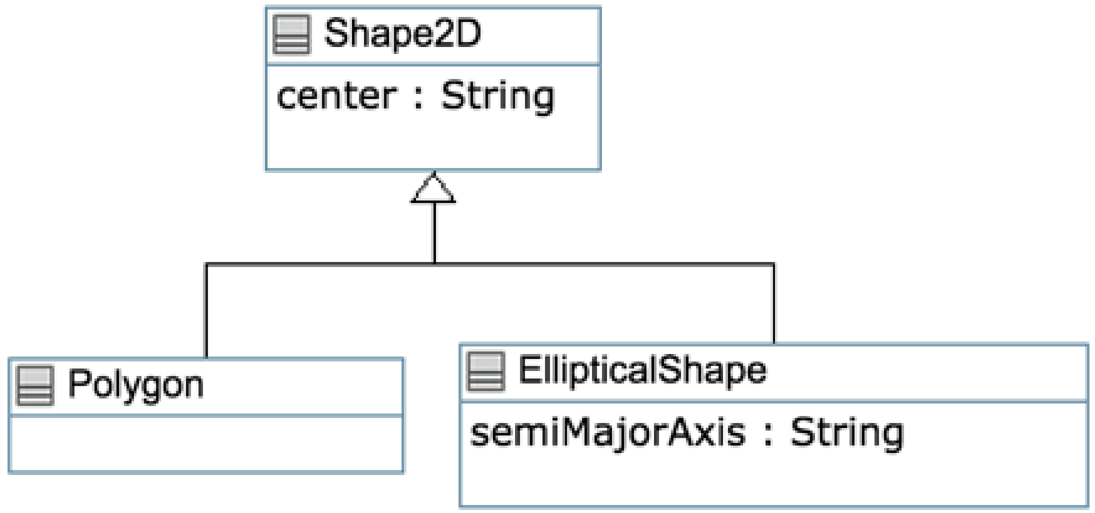


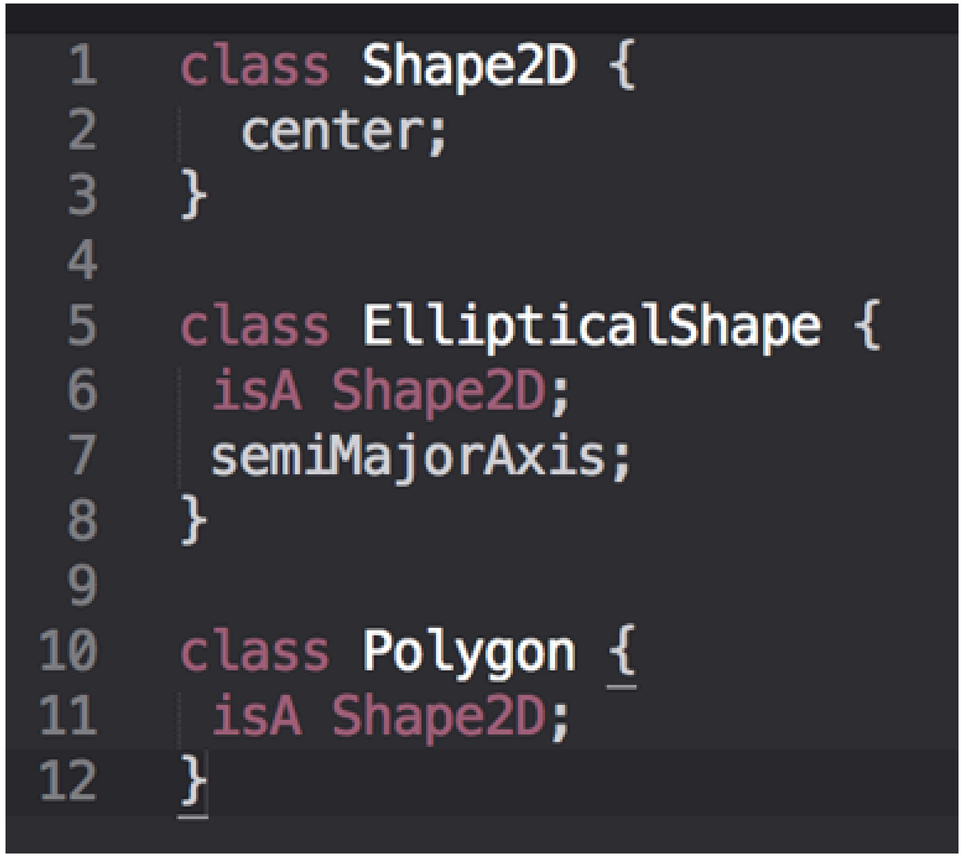
http://associations.umple.org/



```
class A {}

// Class with inline association having role name
class B {
  1 -- * A endTwo;
}

// Class with reflexive association
class C {
  0..1 -- * C;
  1 -- 0..1 D; // D is external
}

// Independently defined and directed association
association {
  0..1 A -> * C;
}

// Class with composition
class E {
  0..1 e <@>- * A a;
}

// Reference to a class defined elsewhere
external D {}

```

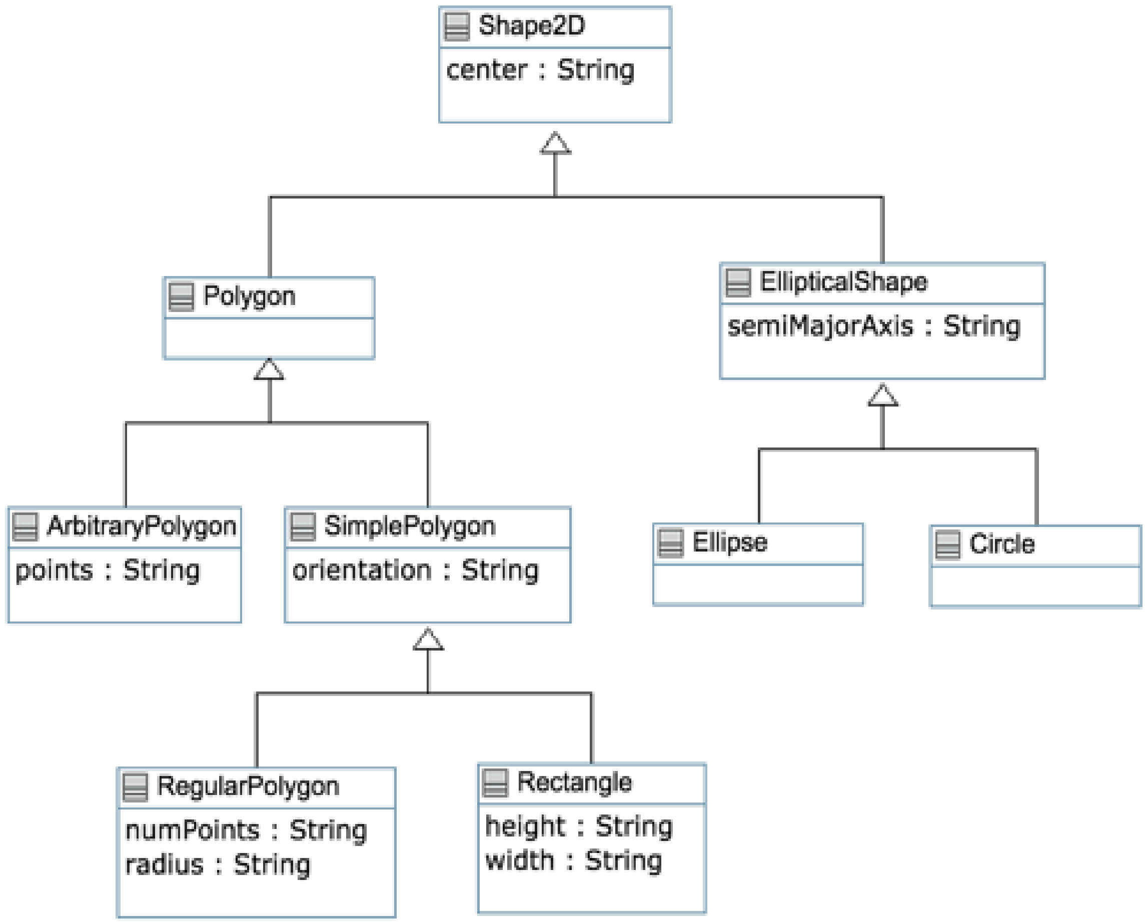
## Associations Exercise #1

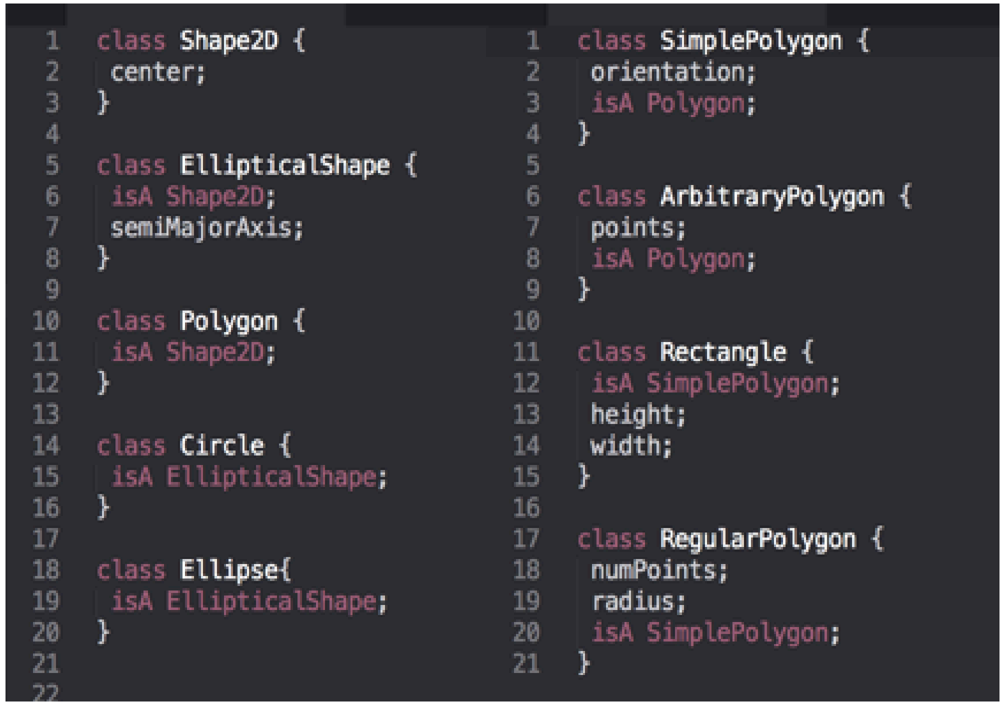




```
class Shape2D {
  center;
}

class EllipticalShape {
 isA Shape2D;
 semiMajorAxis;
}

class Polygon {
 isA Shape2D;
}
```

## Associations Exercise #2





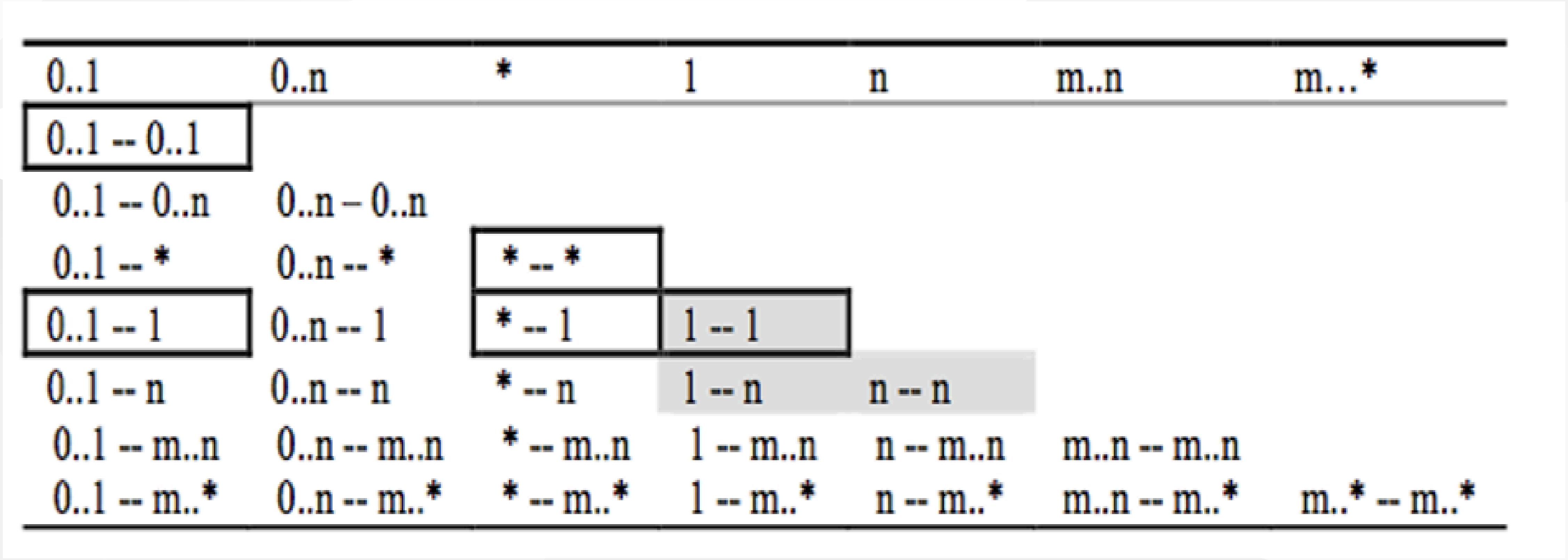
```
class Shape2D {
 center;
}

class EllipticalShape {
 isA Shape2D;
 semiMajorAxis;
}

class Polygon {
 isA Shape2D;
}

class Circle {
 isA EllipticalShape;
}

class Ellipse{
 isA EllipticalShape;
}
```

```
class SimplePolygon {
 orientation;
 isA Polygon;
}

class ArbitraryPolygon {
 points;
 isA Polygon;
}

class Rectangle {
 isA SimplePolygon;
 height;
 width;
}

class RegularPolygon {
 numPoints;
 radius;
 isA SimplePolygon;
}
```

## 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

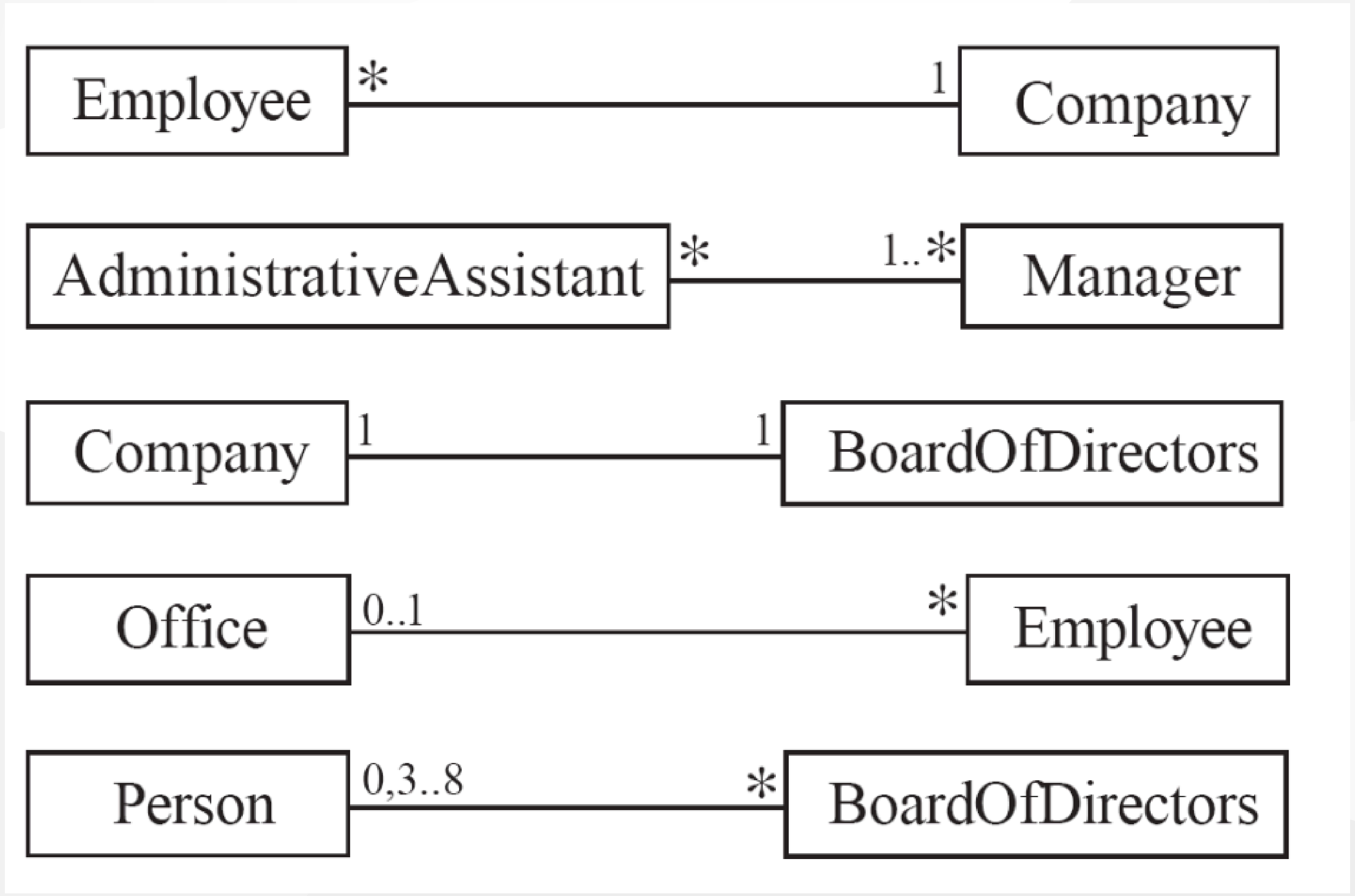
- Directional Associations

```
* -> 0..1, * -> 1, * -> *, * -> m..n, * - >n, *->m..* and*->0..n.
```

- Symmetric Reflexive

```
0..1, 0..n, *, 1, n, m..n,m..*
```

## Basic UML associations

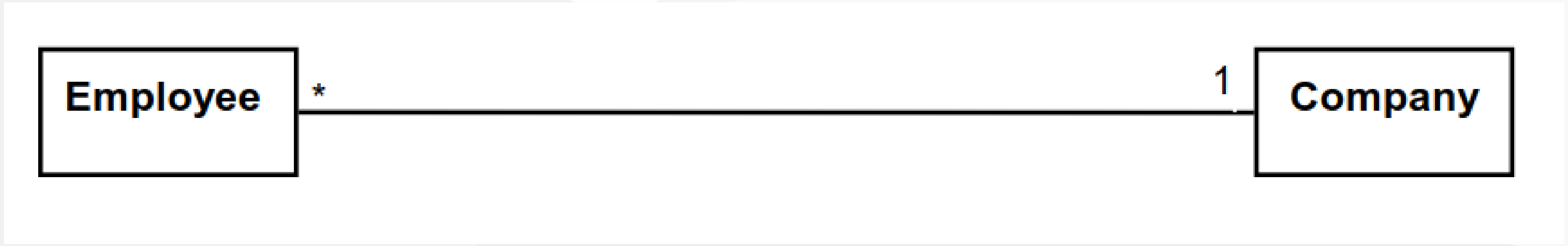


## Many-to-one associations (1)

```
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;
}
```

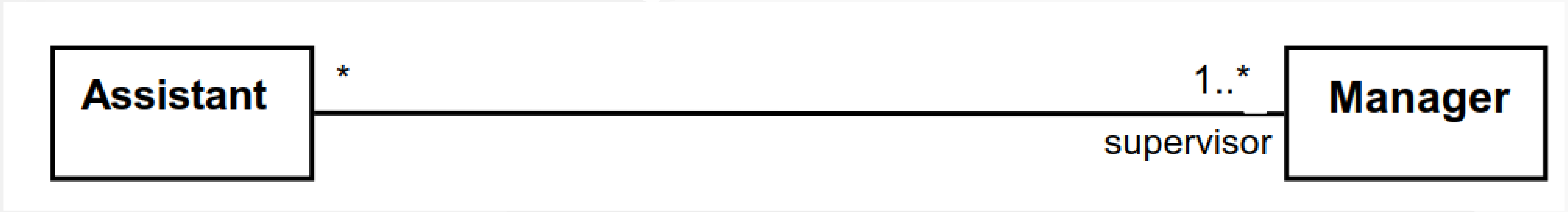
## Referential Integrity

- 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?

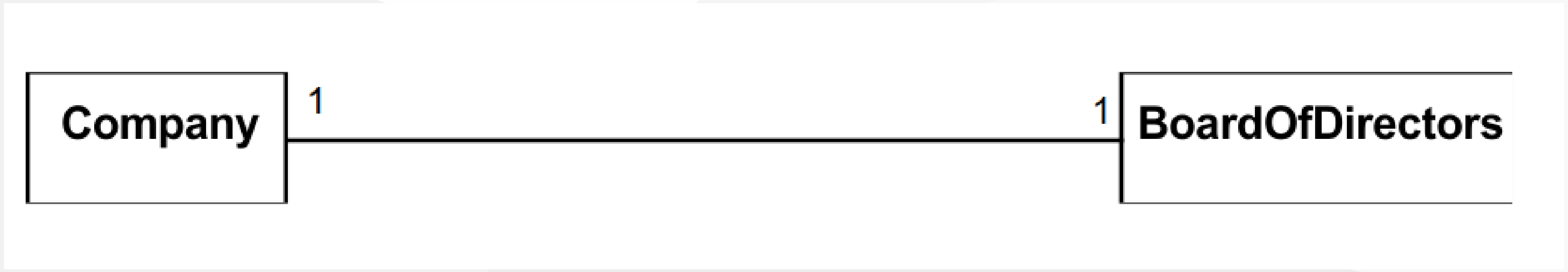
Open in UMPLE



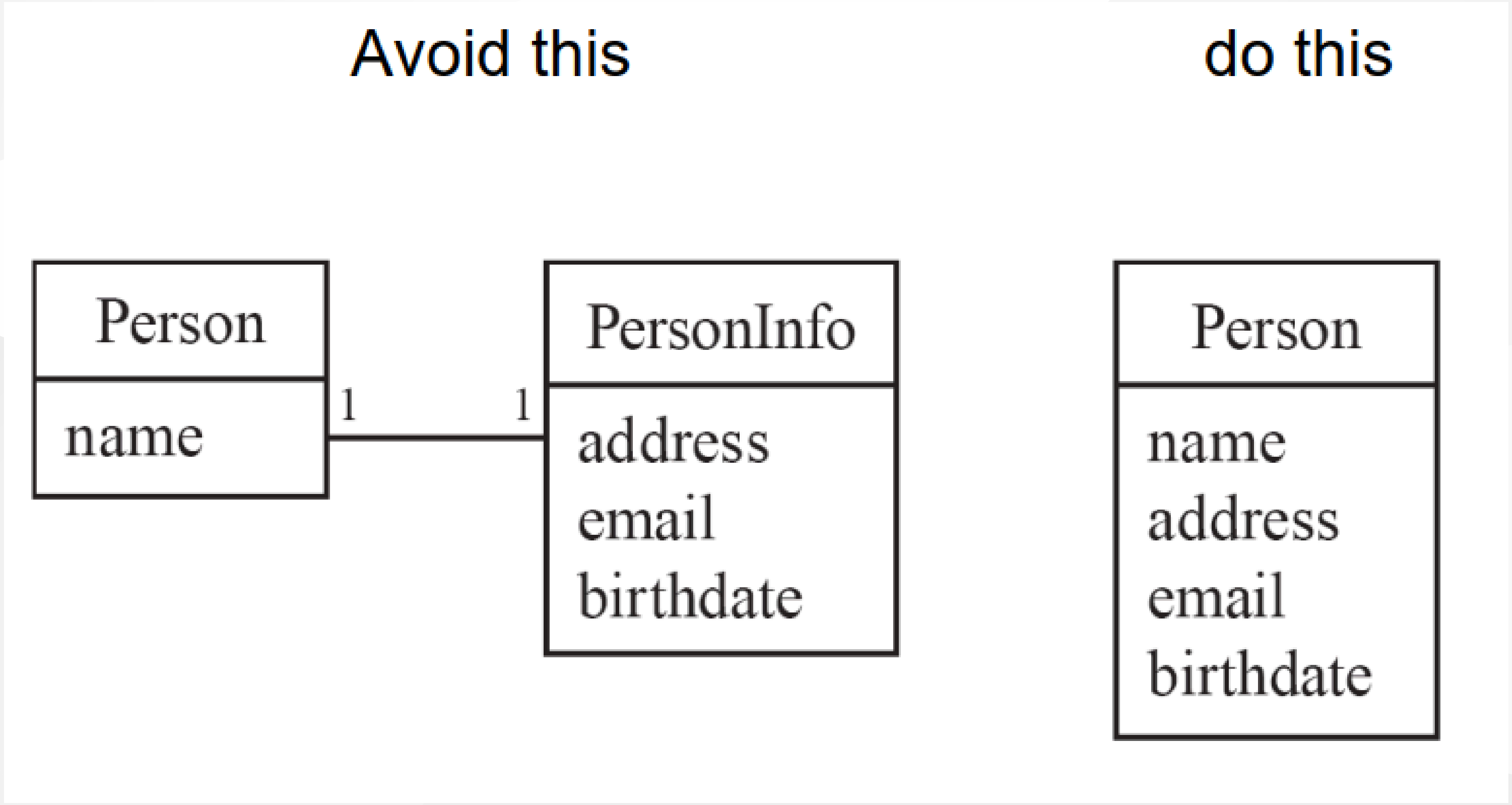
## One-to-One Associations (Use cautiously)

- 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

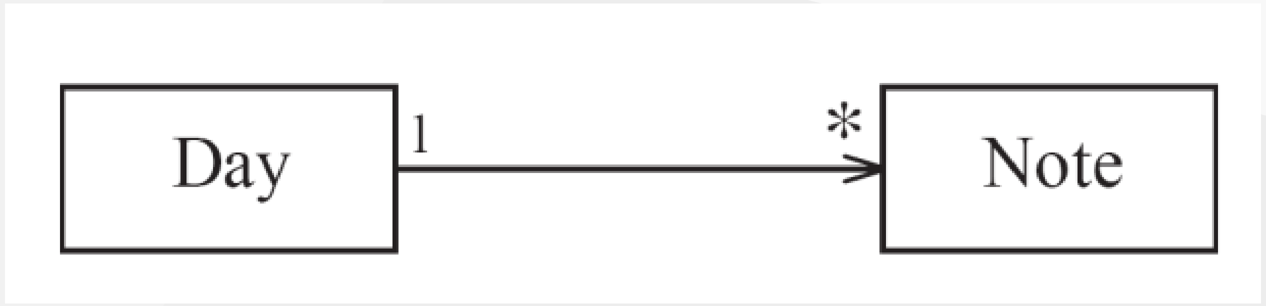


## Unidirectional Associations

- 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 is belongs to
  - Note remains 'uncoupled' and can be used in other contexts

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

Open in UMPLE

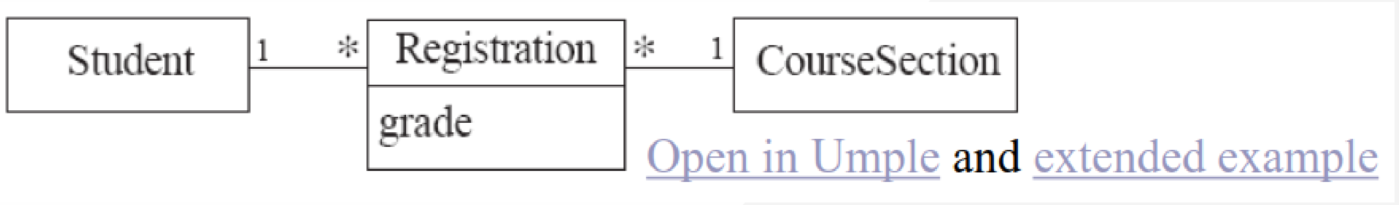


## Association Classes

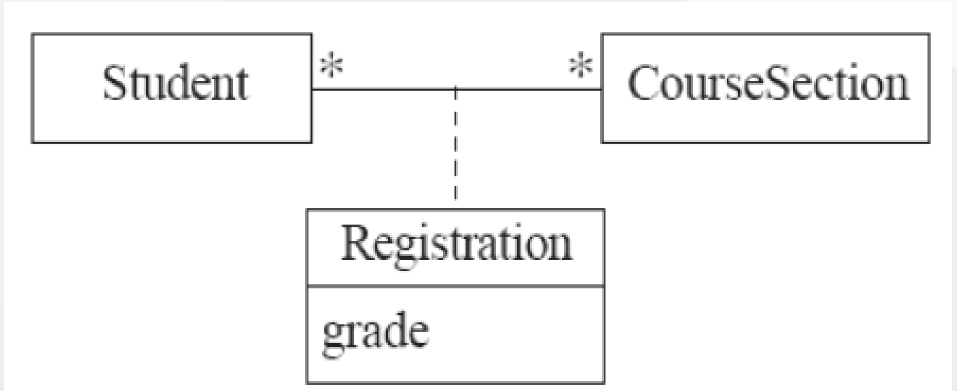
- Sometimes, an attribute that concerns two associated classes cannot be placed in either of the classes

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.)

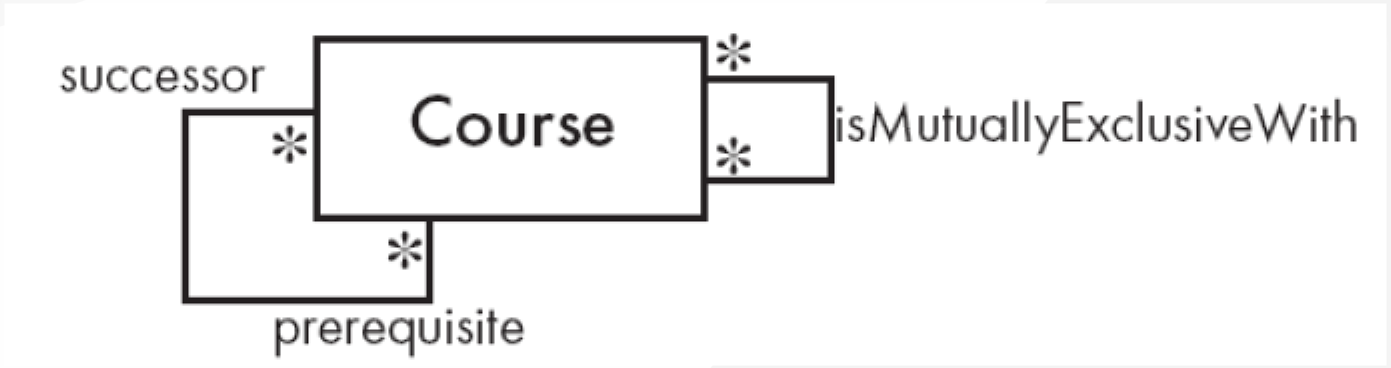
- UMPLE code

```
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

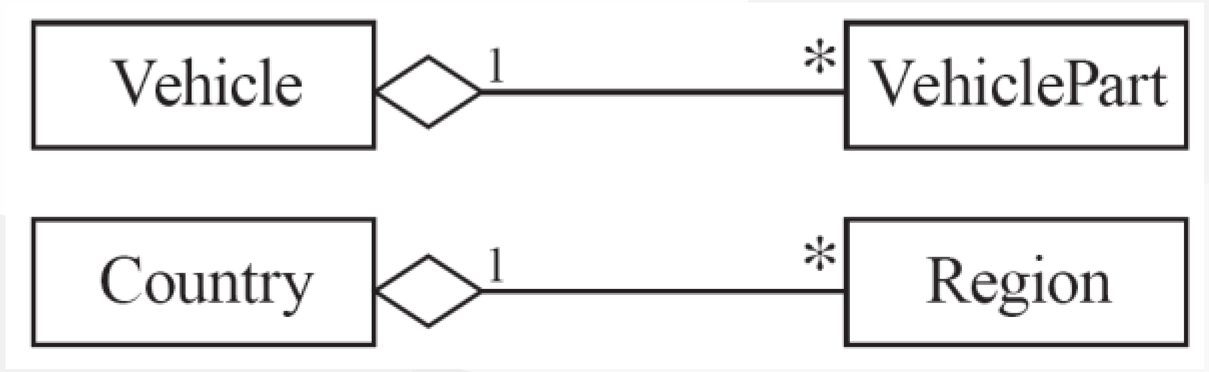
- The following are equivalent to allow flexibility:

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

}
```

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

}
```

## Aggregation

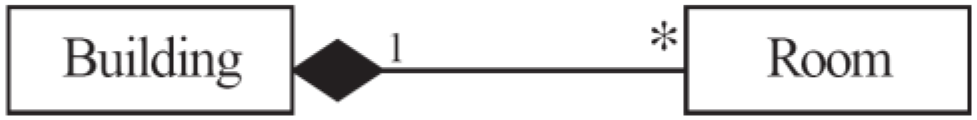
- Aggregations are ordinary associations that represent part-whole relationships.
  - The 'whole' side is often called the assembly or the aggregate
  - This is a shorthand for association named `isPartOf`
  - UMPLE has no special syntax currently



```
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

$End - Of - Week - 6$