Software Engineering

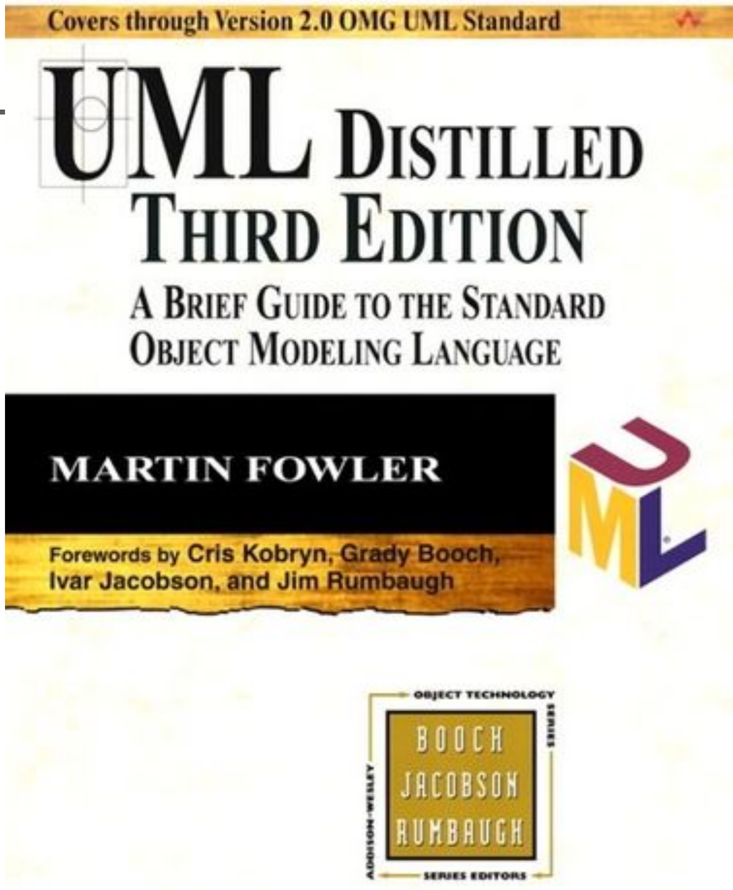
Lesson #03 - Practice

Lesson #03 - Practice

Lesson #03 - SE - Practice

References:

UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition, Martin Fowler, 2004, Addison-Wesley Professional;


Covers through Version 2.0 OMG UML Standard
UML Distilled
Third Edition
A Brief Guide to the Standard Object Modeling Language
Martin Fowler
Forewords by Cris Kobryn, Grady Booch, Ivar Jacobson, and Jim Rumbaugh
Object Technology Series
Booch Jacobson Rumbaugh
Series Editors
Addison-Wesley

Agenda: Lesson #03 - Software Engineering - Practice

Chapter #01: Introduction to UML

Chapter #02: Development Processes

Agenda: Lesson #03 - Software Engineering - Practice

Agenda: Lesson #03 - Software Engineering - Practice

What Is the UML?

UML

The Unified Modeling Language (UML) is a family of graphical notations, backed by single meta-model, that help in describing and designing software systems, particularly software systems built using the object-oriented (OO) style

UNIFIED
MODELING
LANGUAGE ™

What Is the UML?

UML

The UML is a relatively open standard, controlled by the Object Management Group (OMG), an open consortium of companies

The OMG was formed to build standards that supported interoperability, specifically the interoperability of object-oriented systems

The OMG is perhaps best known for the CORBA standards

What Is the UML?

UML

The UML was born out of the unification of the many object-oriented graphical modeling languages that thrived in the late 1980s and early 1990s

UNIFIED MODELING LANGUAGE™

What Is the UML?

UML

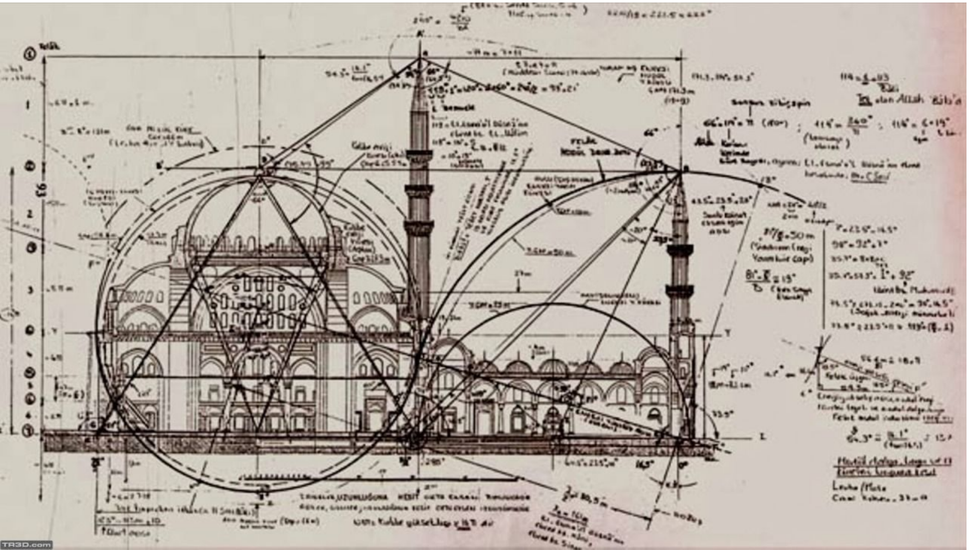
UML as sketch

UNIFIED MODELING LANGUAGE™

What Is the UML?

UML

UML as blueprint

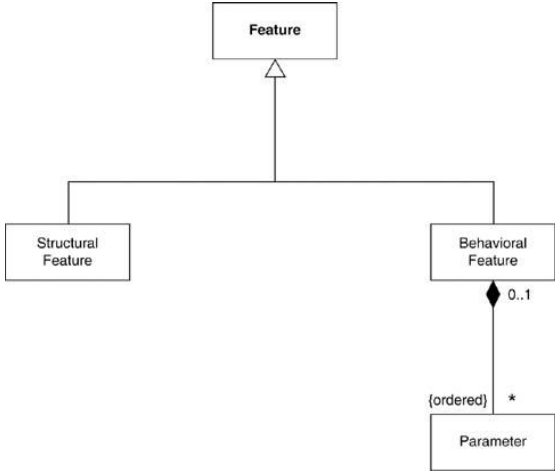


UNIFIED MODELING LANGUAGE™ UML®

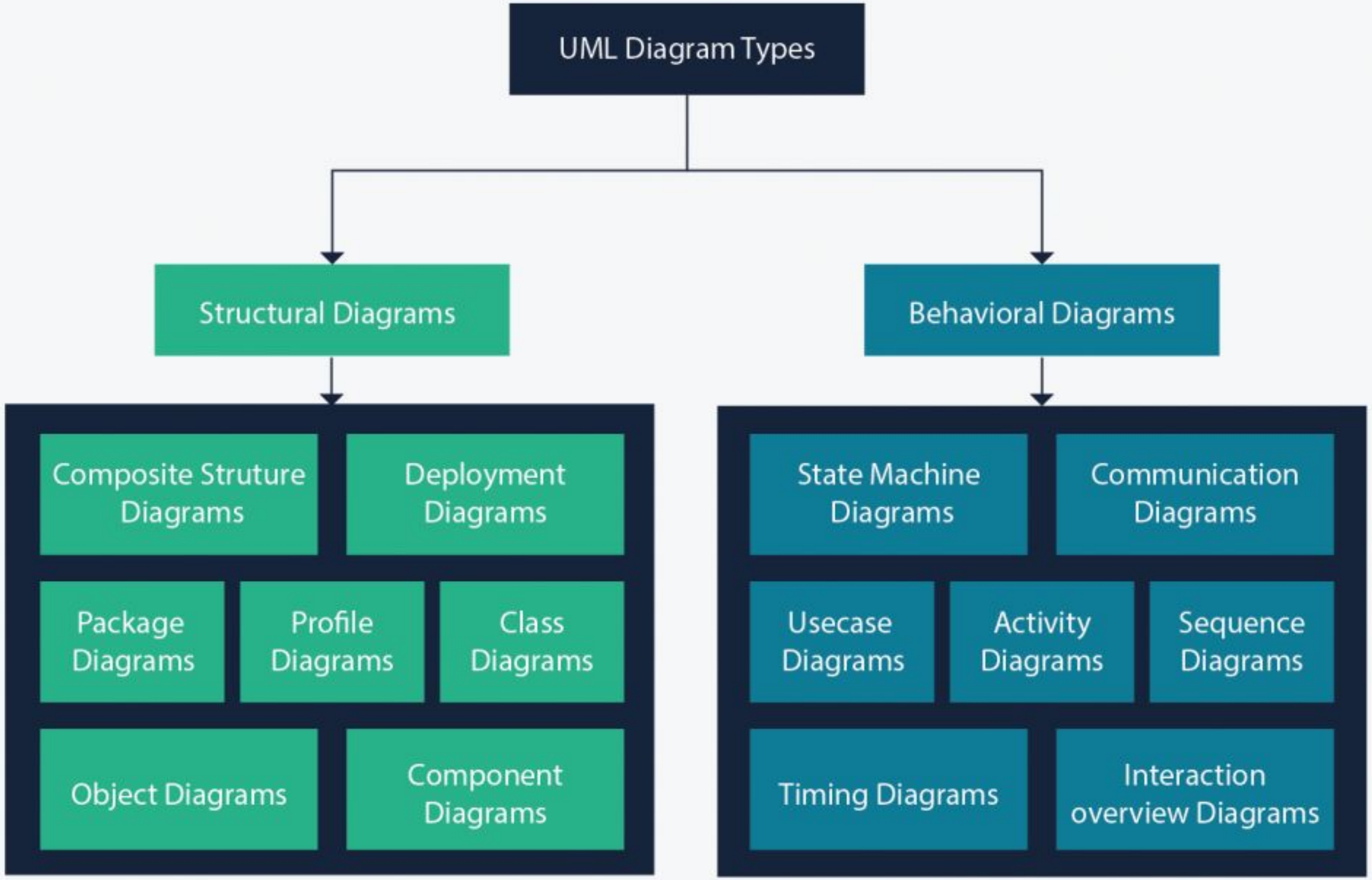
What Is the UML?

UML

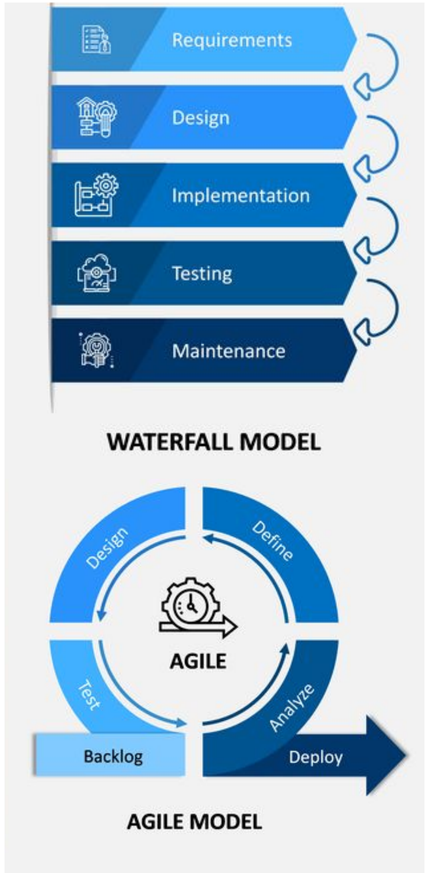
The UML, in its current state, defines a notation and a meta-model. The notation is the graphical stuff you see in models; it is the graphical syntax of the modeling language

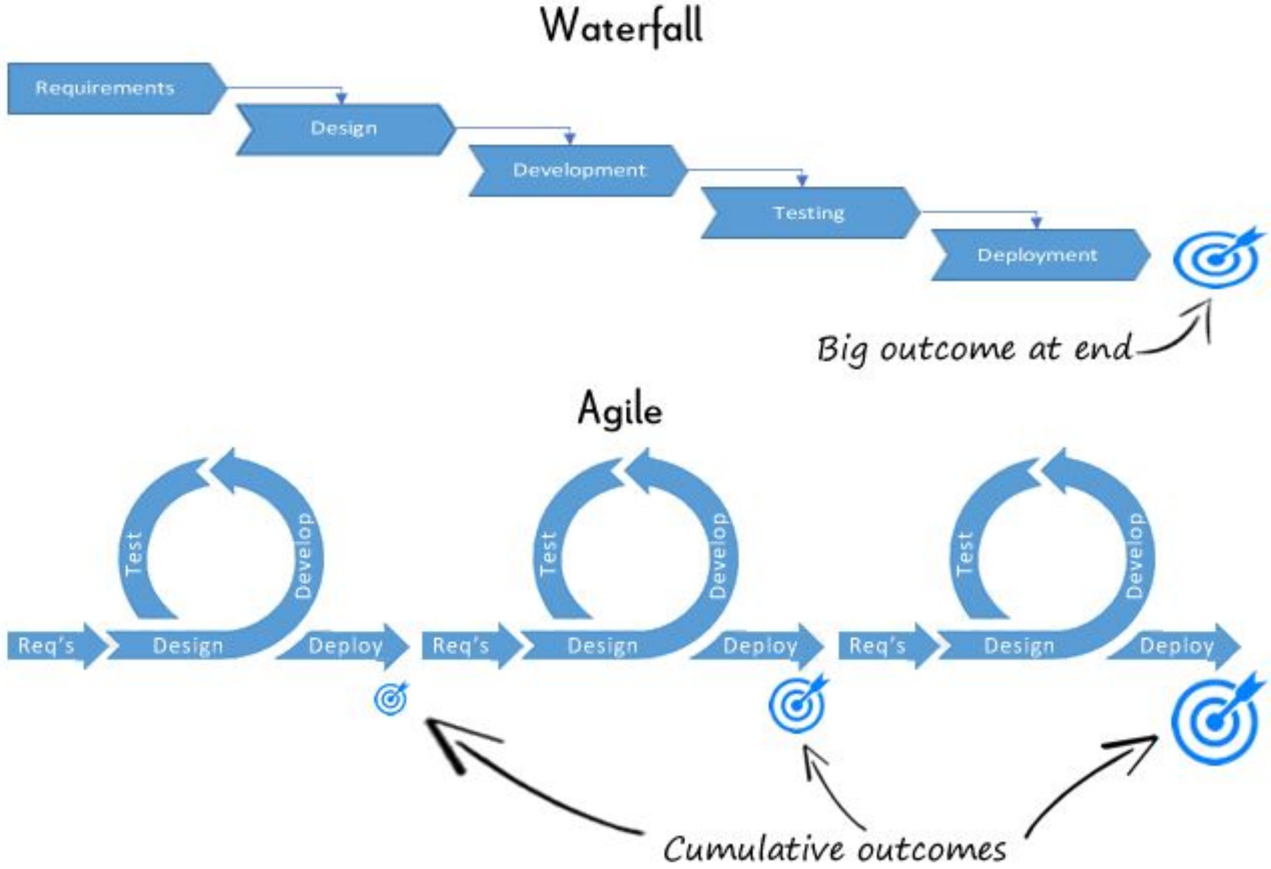


UNIFIED MODELING LANGUAGE™


UML Diagram Types
Structural Diagrams
Behavioral Diagrams
Composite Struture Diagrams
Deployment Diagrams
Package Diagrams
Profile Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
State Machine Diagrams
Communication Diagrams
Usecase Diagrams
Activity Diagrams
Sequence Diagrams
Timing Diagrams
Interaction overview Diagrams

What is the UML?





Fitting a UML to a Project

Using the UML doesn't necessarily simply developing documents or feeding a complex CASE tool

Many people draw UML diagrams on whiteboards only during a meeting to help communicate their ideas

Fitting a UML to a Project

Requirements Analysis

- Use cases
- A class diagram
- An activity diagram
- A state diagram

What is the UML?

Fitting a UML to a Project

Design

- Class diagram
- Package diagram
- Deployment diagram
- Sequence diagram
- A state diagram

Fitting a UML to a Project

Documentation

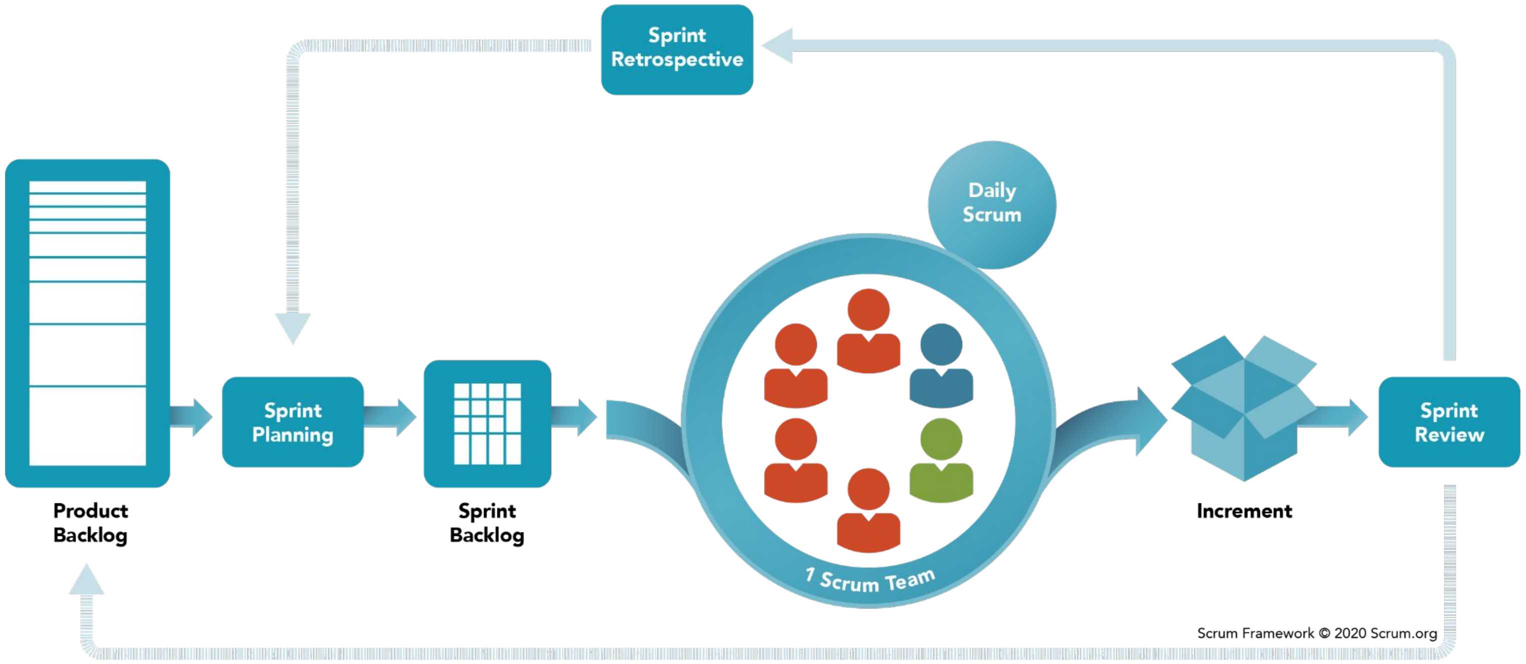
Once you have built the software, you can use the UML to help document what you have done

UML diagrams useful for getting an overall understanding of a system

What is the UML?

Choosing a Development Process

SCRUM



Agenda: Lesson #03 - Software Engineering - Practice

The Essentials

The class diagram is not only widely used but also subject to the greatest range of modeling concepts

Although the basic elements are needed by everyone, the advanced concepts are used less often

The Essentials

A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them

Class diagrams also show the properties and operations of a class and the constraints that apply to the way objects are connected

The UML uses the term feature as a general term that covers properties and operations of a class

Properties

Properties represent structural features of a class

As a first approximation, you can think of properties as corresponding to fields in a class

The reality is rather involved, as we shall see, but that's a reasonable place to start

Properties

Properties are a single concept, but they appear in two quite distinct notations: attributes and associations

Although they look quite different on a diagram, they are really the same thing

Properties

Attributes

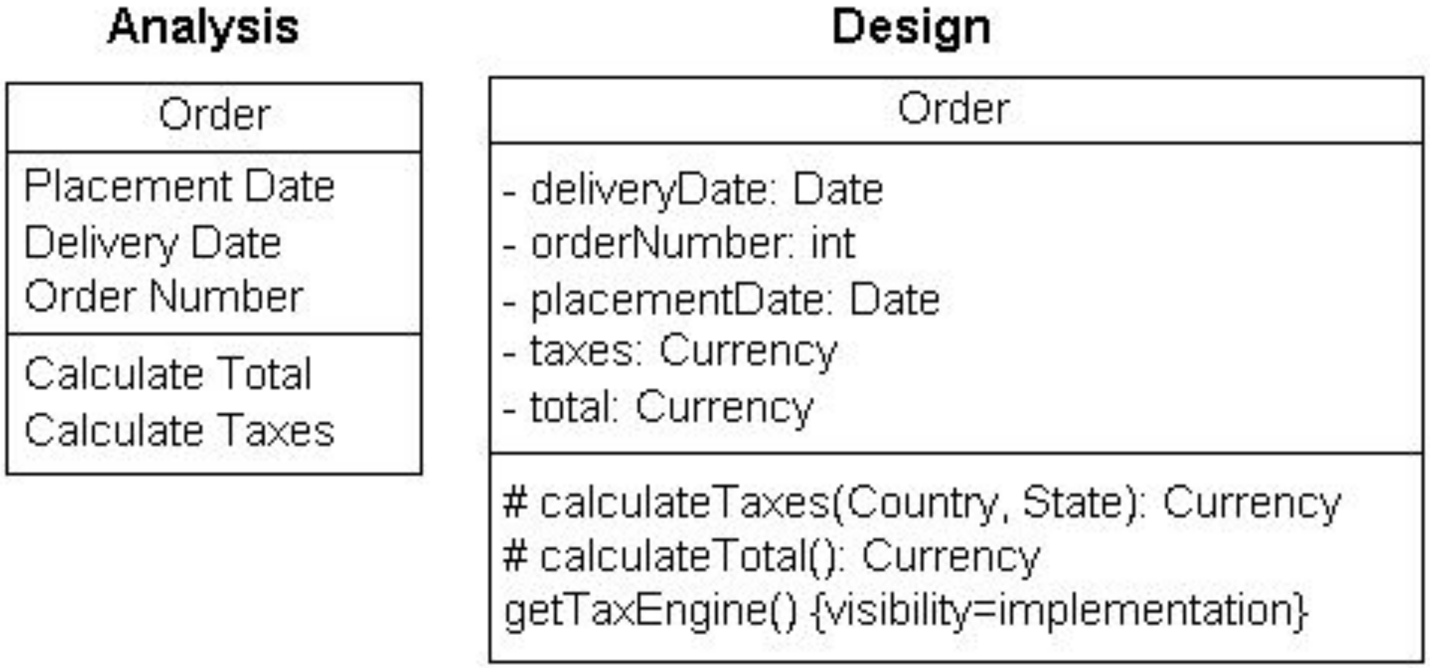
The attribute notation describes a property as a line of text within the class box itself

The full form of an attribute is:
visibility name: type multiplicity = default {property-string}

An example of this is:
name: String [1] = "Untitled" {readOnly}

Properties

Attributes



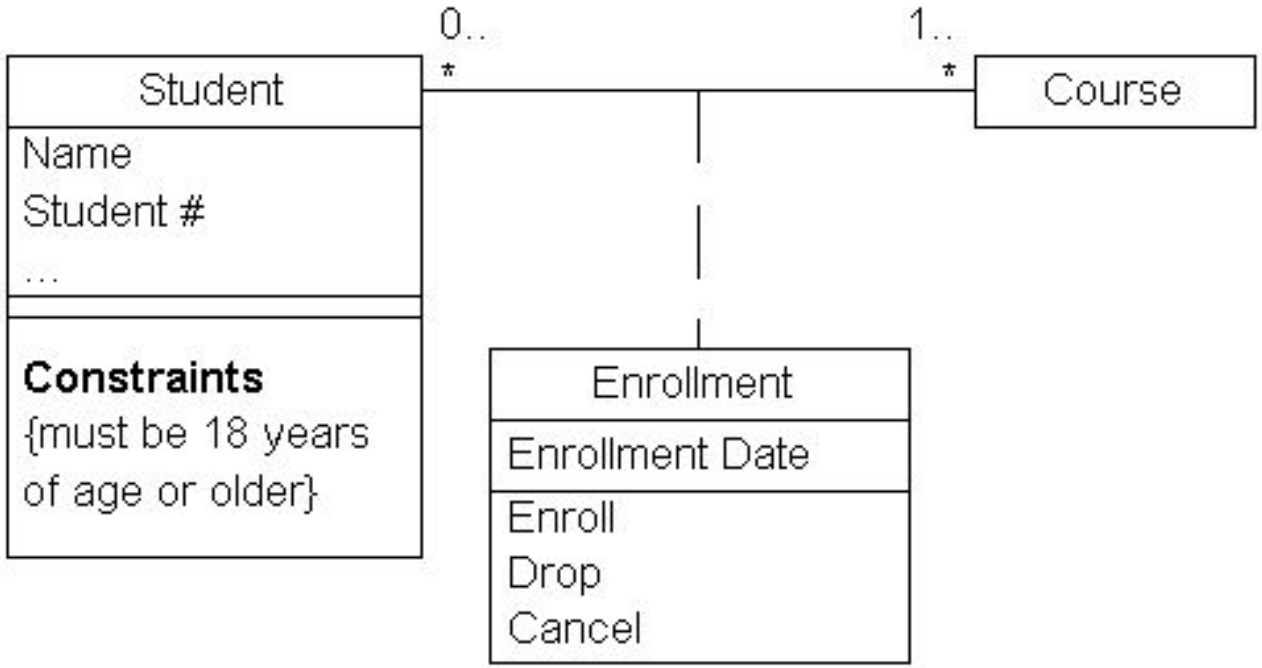
Properties

Associations

- An association is a solid line between two classes, directed from the source class to the target class
- The name of the property goes at the target end of the association, together with its multiplicity
- The target end of the association links to the class that is the type of the property

Properties

Associations



When to Use Class Diagrams

Class diagrams are the backbone of the UML, so you will find yourself using them all the time

The biggest danger with class diagrams is that you can focus exclusively on structure and ignore behavior

When to Use Class Diagrams

Therefore, when drawing class diagrams to understand software, always do them in conjunction with some form of behavioral technique

If you're going well, you'll find yourself swapping between the techniques frequently

Multiplicity

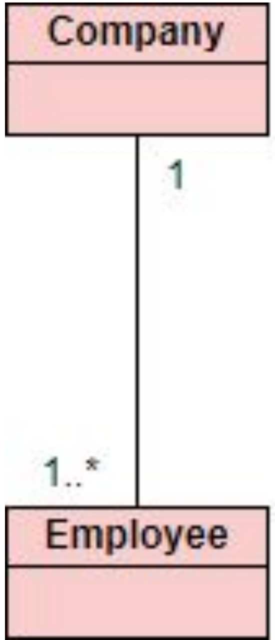
The multiplicity of a property is an indication of how many objects may fill the property

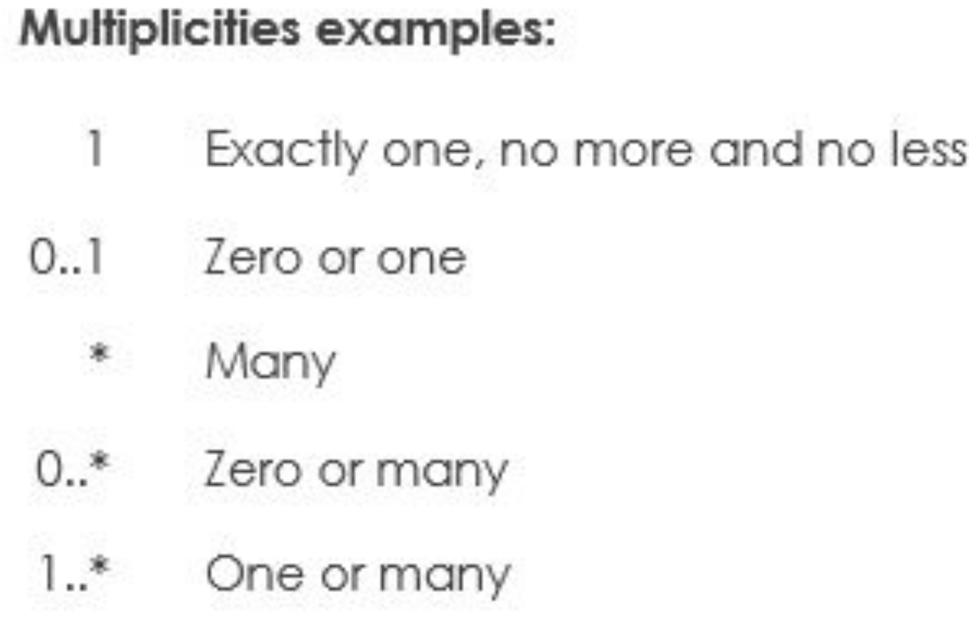
The most common multiplicities you will see are

- 1 (An order must have exactly one customer)
- 0..1 (A corporate customer may or may not have a single sales rep)
- * (A customer need not place an Order and there is no upper limit to the number of Orders a Customer may place—zero or more orders)

Multiplicity

Multiplicity



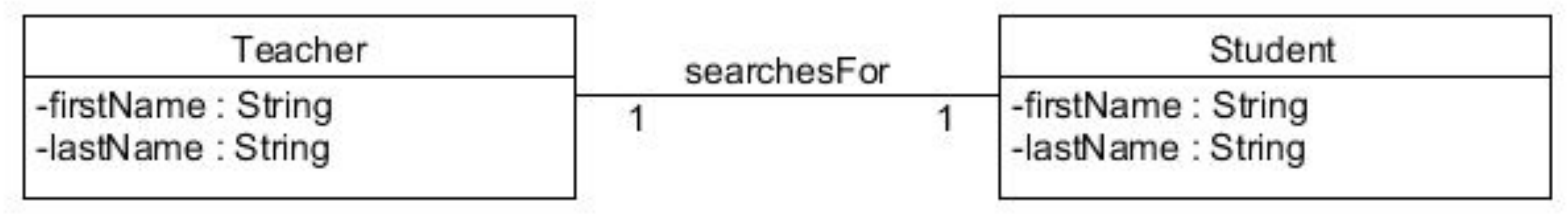


Multiplicities examples:

1	Exactly one, no more and no less
0..1	Zero or one
*	Many
0..*	Zero or many
1..*	One or many

Bidirectional Associations

A bidirectional association



Operations

Operations are the actions that a class knows to carry out

Operations most obviously correspond to the methods on a class

Normally, you don't show those operations that simply manipulate properties, because they can usually be inferred

Generalization

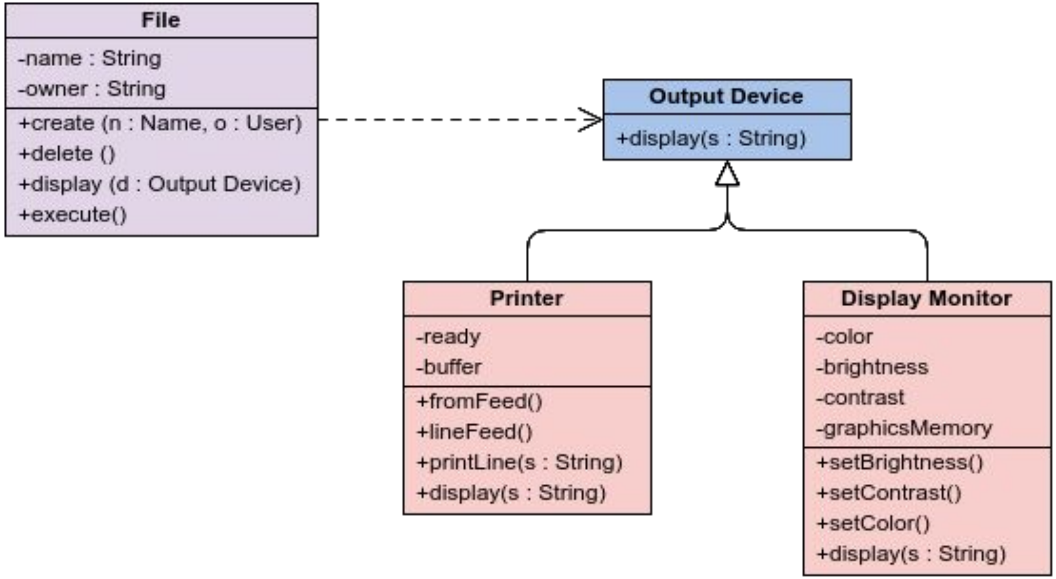
A typical example of generalization involves the personal and corporate customers of a business

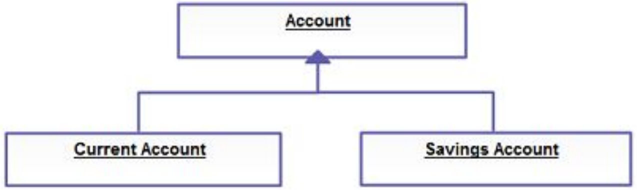
They have differences but also many similarities

The similarities can be placed in a general Customer class (the supertype), with Personal Customer and Corporate Customer as subtypes

Generalization

An important principle of using inheritance effectively is substitutability

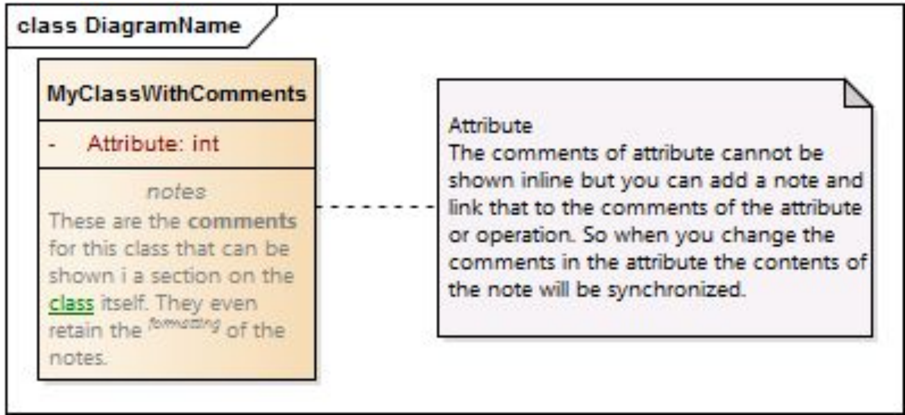




Notes and Comments

Notes are comments in the diagrams

Notes can stand on their own, or they can be linked with a dashed line to the elements they are commenting


class DiagramName
MyClassWithComments
- Attribute: int
notes
These are the comments for this class that can be shown i a section on the class itself. They even retain the formatting of the notes.
Attribute
The comments of attribute cannot be shown inline but you can add a note and link that to the comments of the attribute or operation. So when you change the comments in the attribute the contents of the note will be synchronized.

Dependency

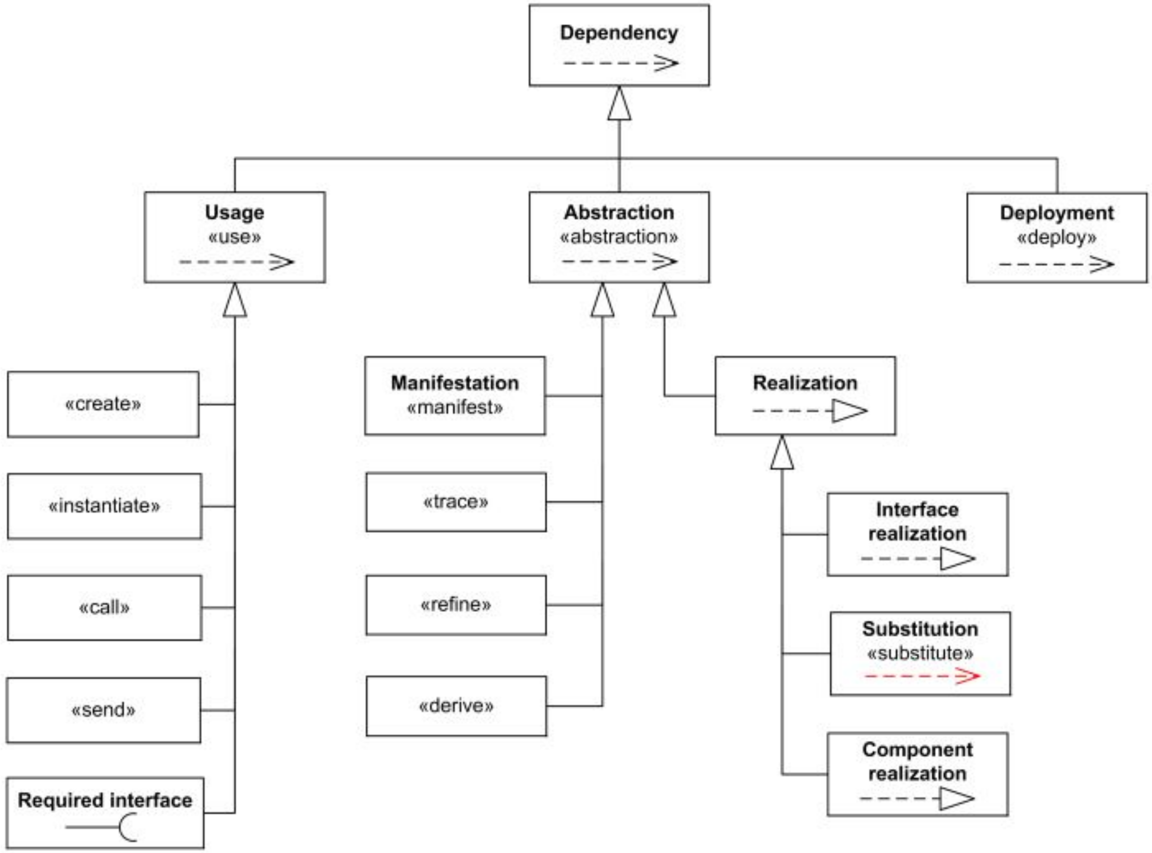
A dependency exists between two elements if changes to the definition of one element (the supplier) may cause changes to the other (the client)

With classes, dependencies exist for various reasons: One class sends a message to another; one class has another as part of its data; one class mentions another as a parameter to an operation

If a class changes its interface, any message sent to that class may no longer be valid

Class Diagrams: The Essentials

Dependency



Constraint Rules

The basic constructs of association, attribute, and generalization do much to specify important constraints, but they cannot indicate every constraint

These constraints still need to be captured; the class diagram is a good place to do that

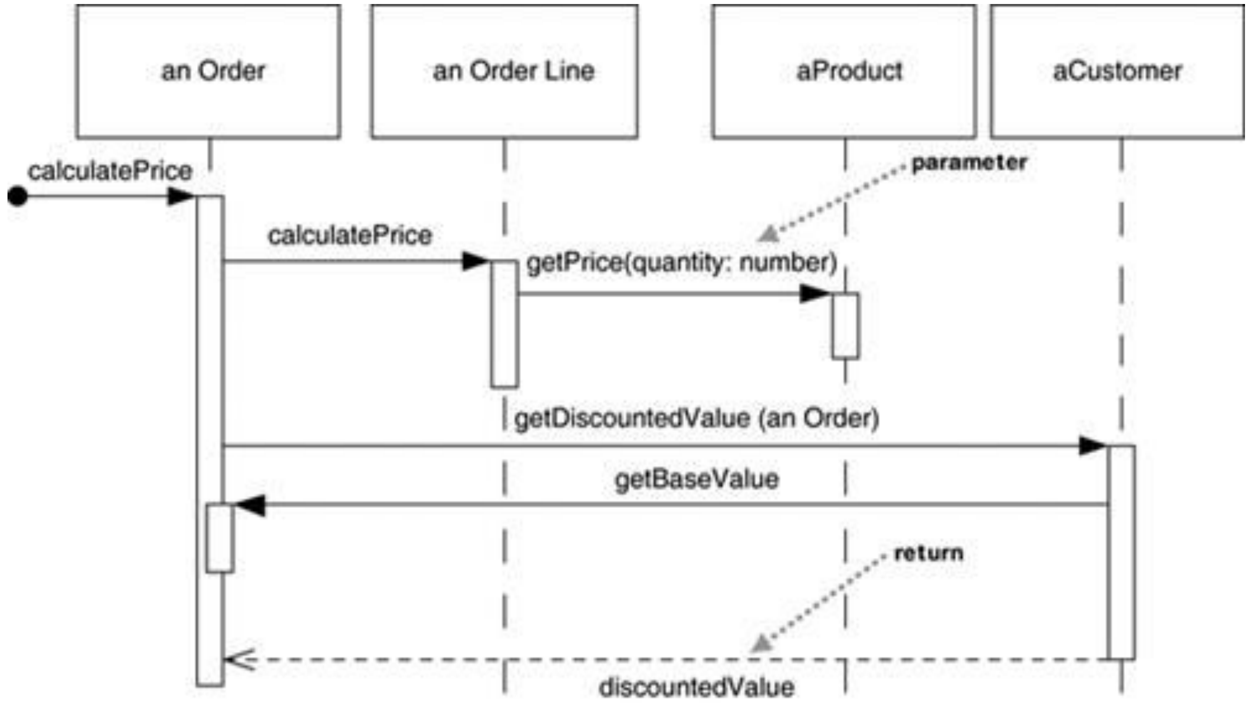
Agenda: Lesson #03 - Software Engineering - Practice

Sequence Diagrams

Interaction diagrams describe how groups of objects collaborate in some behavior

The UML defines several forms of interaction diagram, of which the most common is the sequence diagram

Sequence Diagrams



Sequence Diagrams

Typically, a sequence diagram captures the behavior of a single scenario

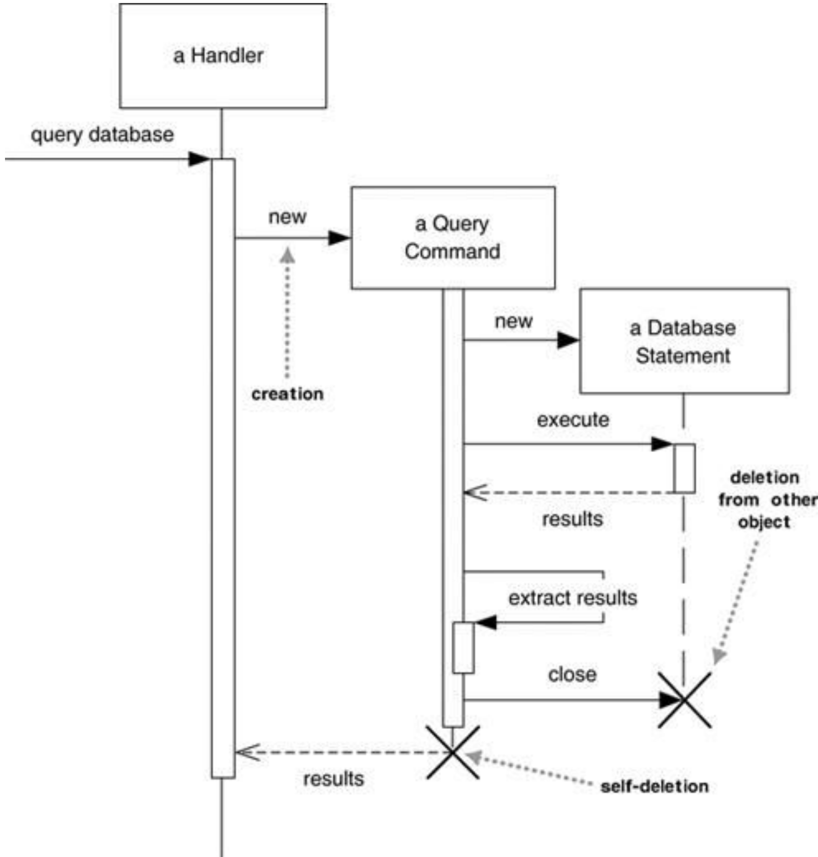
The diagram shows a number of example objects and the messages that are passed between these objects within the use case

Sequence Diagrams

Sequence Diagrams

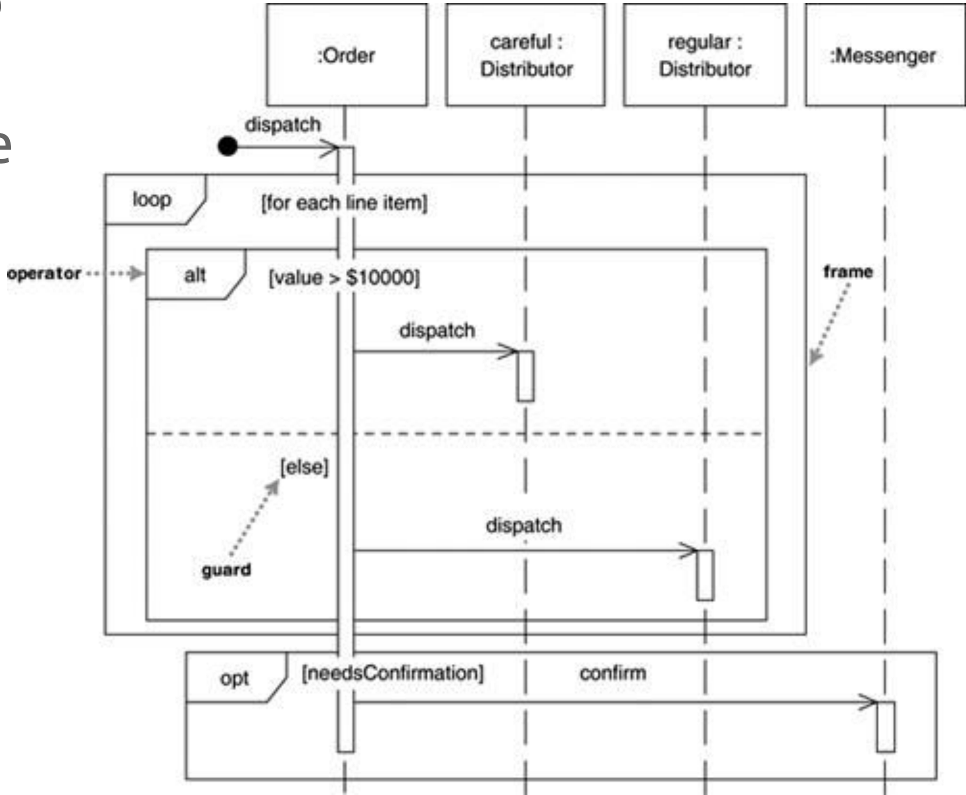
Creating and Deleting Participants

- Sequence diagrams show some extra notation for creating and deleting participants



Sequence Diagrams

Loops, Conditionals, and the Like



Agenda: Lesson #03 - Software Engineering - Practice

Class work

Software Engineering, 10. Global Edition

At the end of class, please, send your work to z.aldamuratov@kbtu.kz, indicating Software Engineering Practice #03 & your surname and name

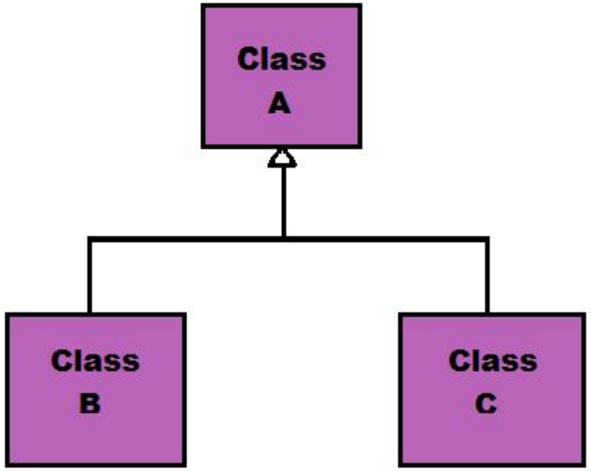
Page: 164 Exercise: 5.6 Tool: StarUML

Look carefully at how messages and mailboxes are represented in the email system that you use. Model the object classes that might be used in the system implementation to represent a mailbox and an email message.

COMING SOON

System modeling

- Class Diagrams: Advanced Concepts



Agenda: Lesson #03 - Software Engineering - Practice

Q & A