

# Mobile Application Development

Higher Diploma in Science in Computer Science

---

Produced  
by

Eamonn de Leastar (edelestar@wit.ie)

Department of Computing, Maths & Physics  
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



# Introduction to UML

---

# Why develop a UML model?

---

- Provide structure for problem solving
- Experiment to explore multiple solutions
- Furnish abstractions to manage complexity
- Decrease development costs
- Manage the risk of mistakes

# The Challenge

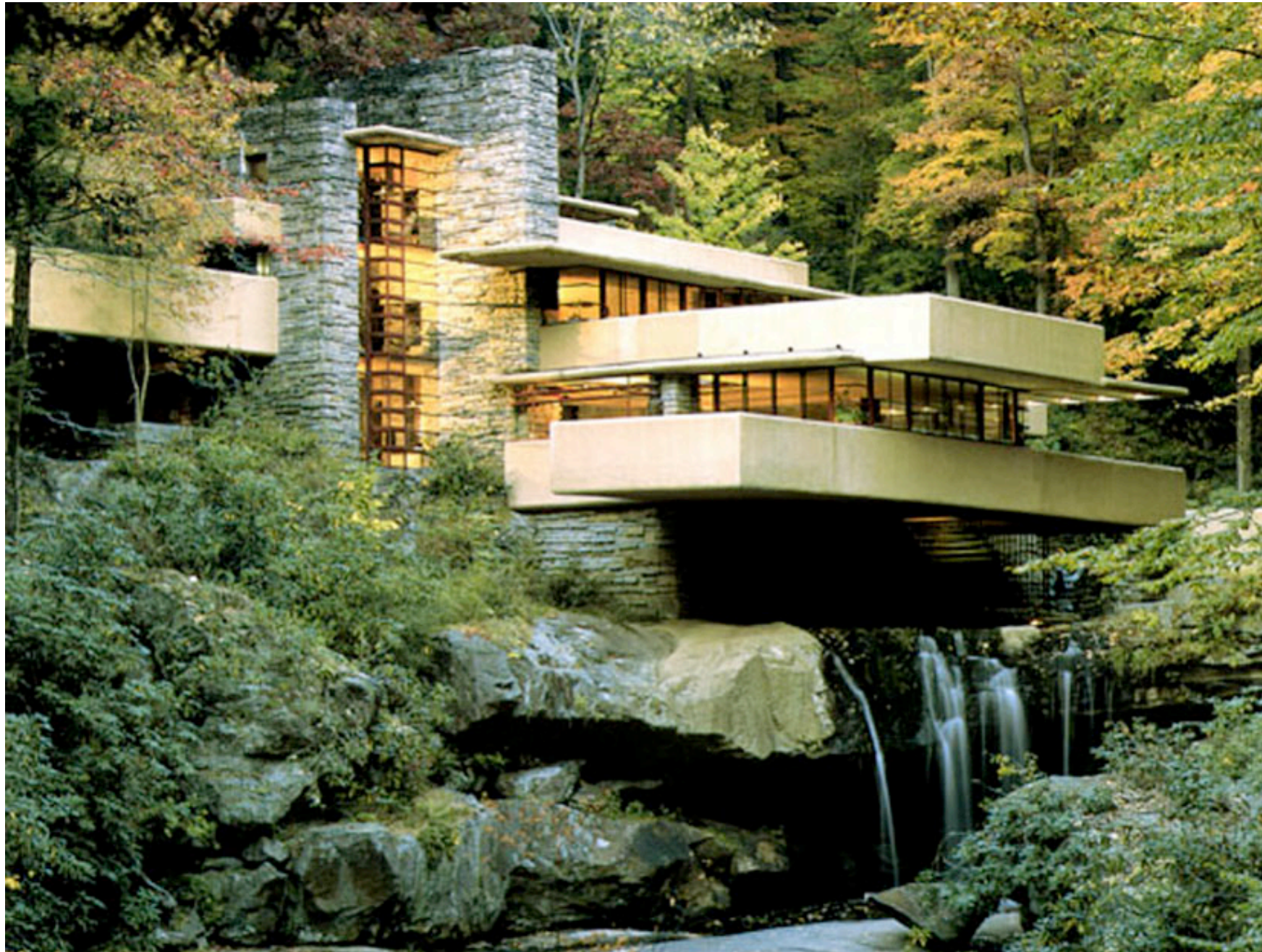
---





# The Vision

---



# Why do we model graphically?

---

- Graphics reveal data.
  - Edward Tufte  
The Visual Display of Quantitative Information, 1983
- 1 bitmap = 1 megaword.
  - Anonymous visual modeler

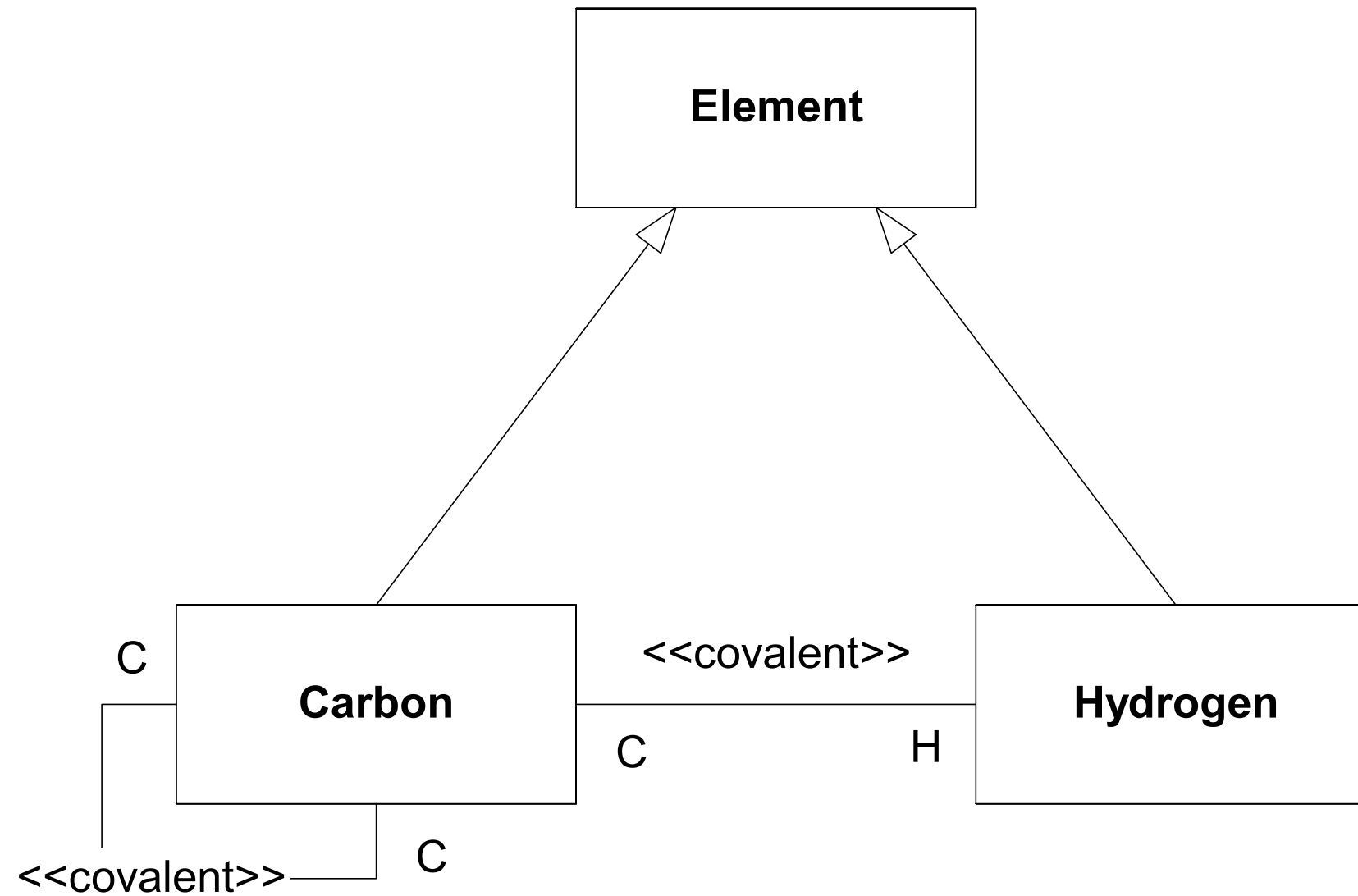
# Building Blocks of UML

---

- The basic building blocks of UML are:
  - model elements (classes, interfaces, components, use cases, etc.)
  - relationships (associations, generalization, dependencies, etc.)
  - diagrams (class diagrams, use case diagrams, interaction diagrams, etc.)
- Simple building blocks are used to create large, complex structures
  - eg elements, bonds and molecules in chemistry
  - eg components, connectors and circuit boards in hardware

# Example : Classifier View

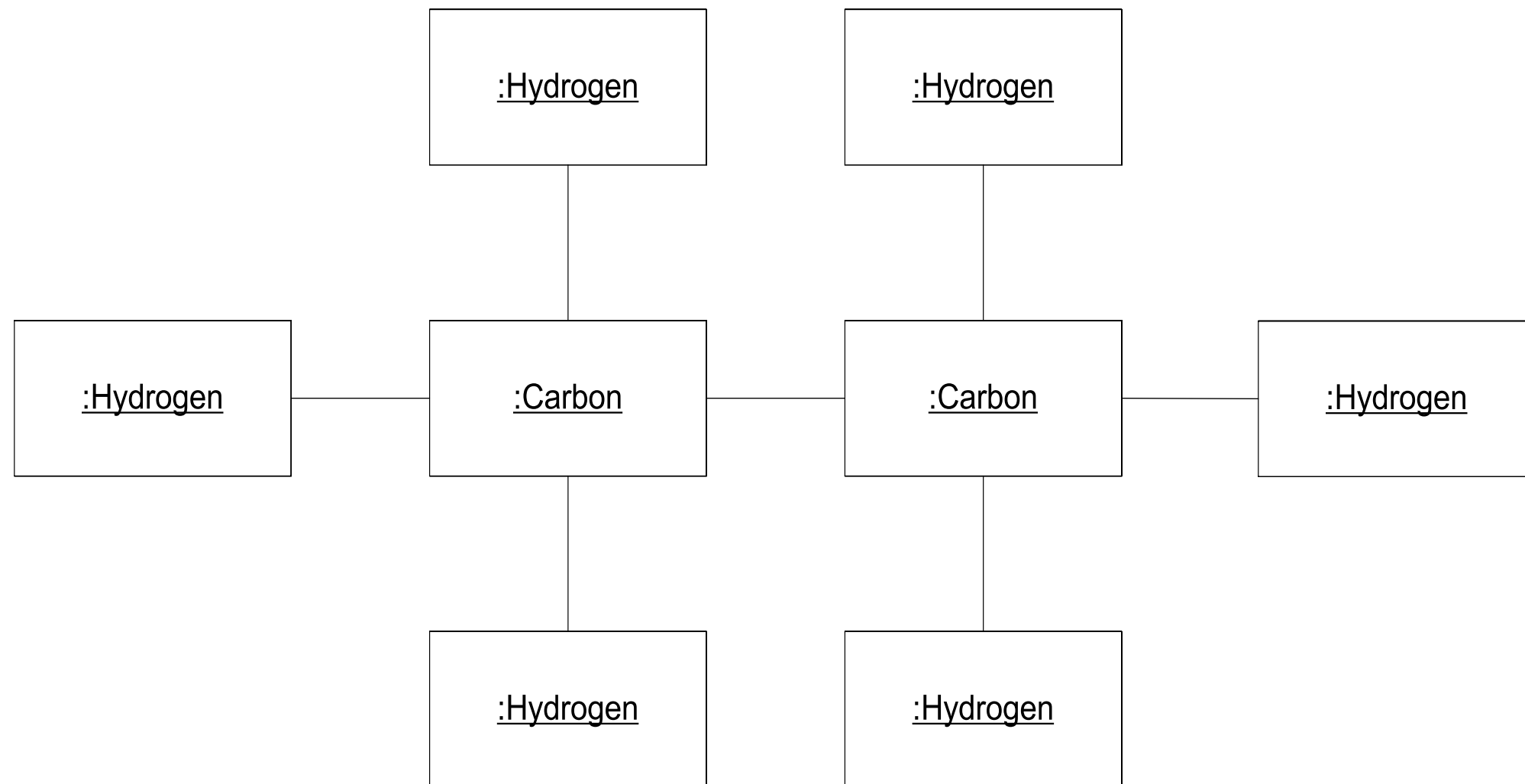
---





# Example: Instance View

---



# UML Modeling Process

---

- Use Case
- Structural
- Behavioural
- Architectural

# Use Case

## Use Case Modeling

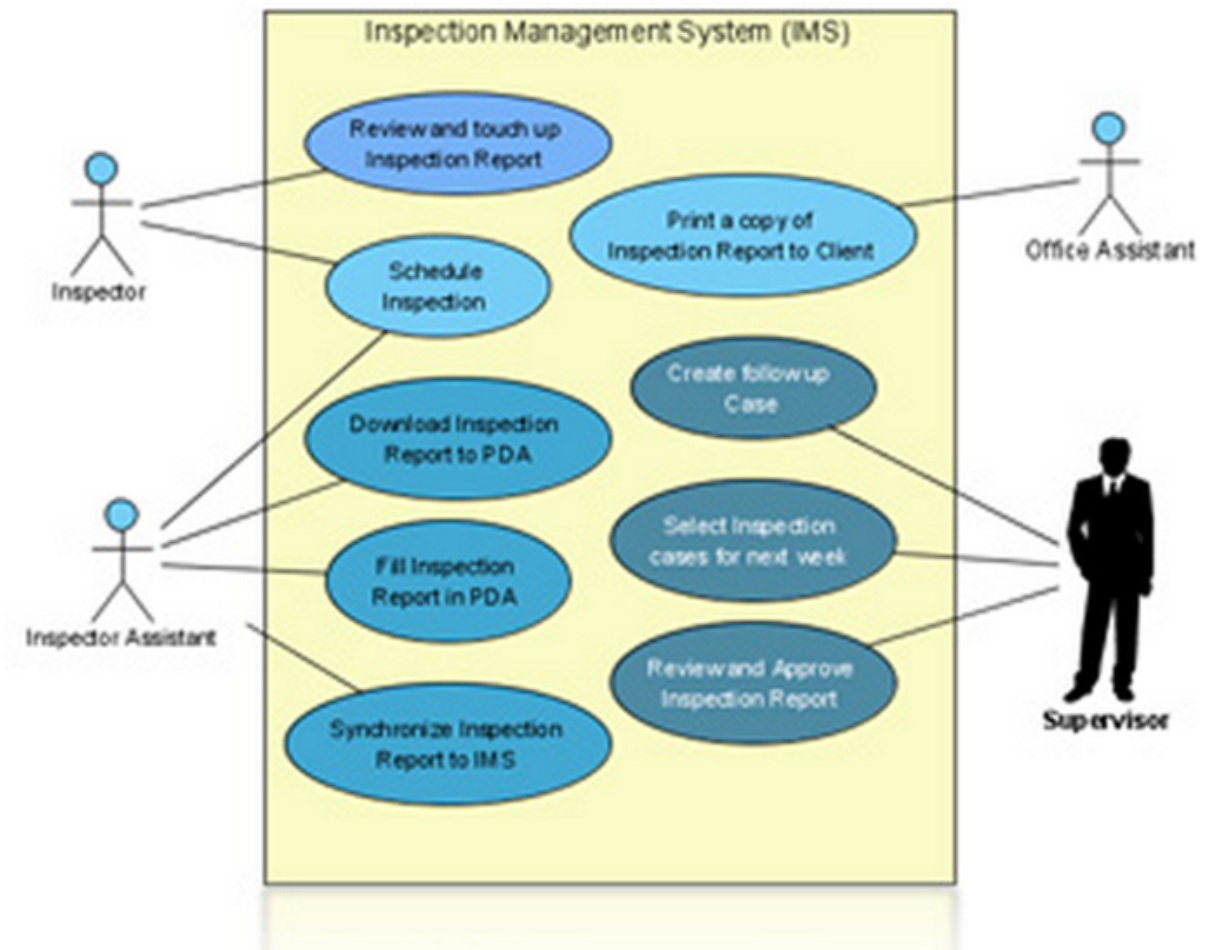
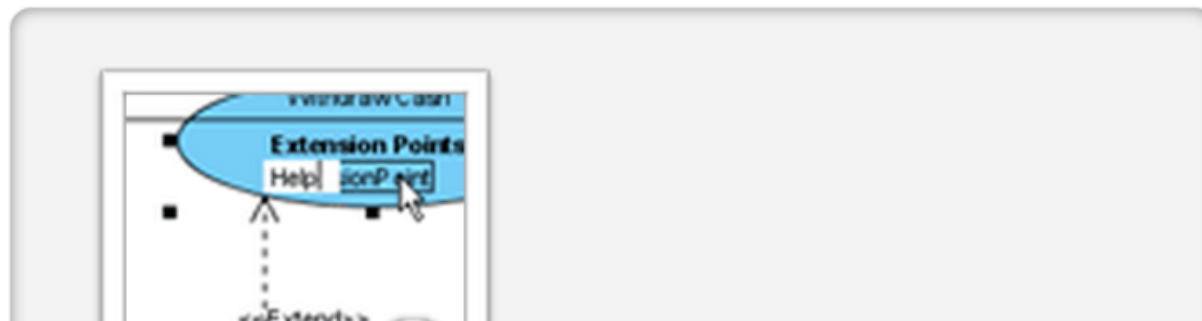
## Structural

## Behavioral

## Architectural

### Use case diagram

You can visualize high level system functions or requirements by drawing use case diagram, which contains primarily actors and use cases. Actors are entities that interact with the system, while use cases are the system functions actors involve. Note that you draw use case diagram to tell readers WHAT the system should do but not how to do.



# Structural Modeling

Use Case

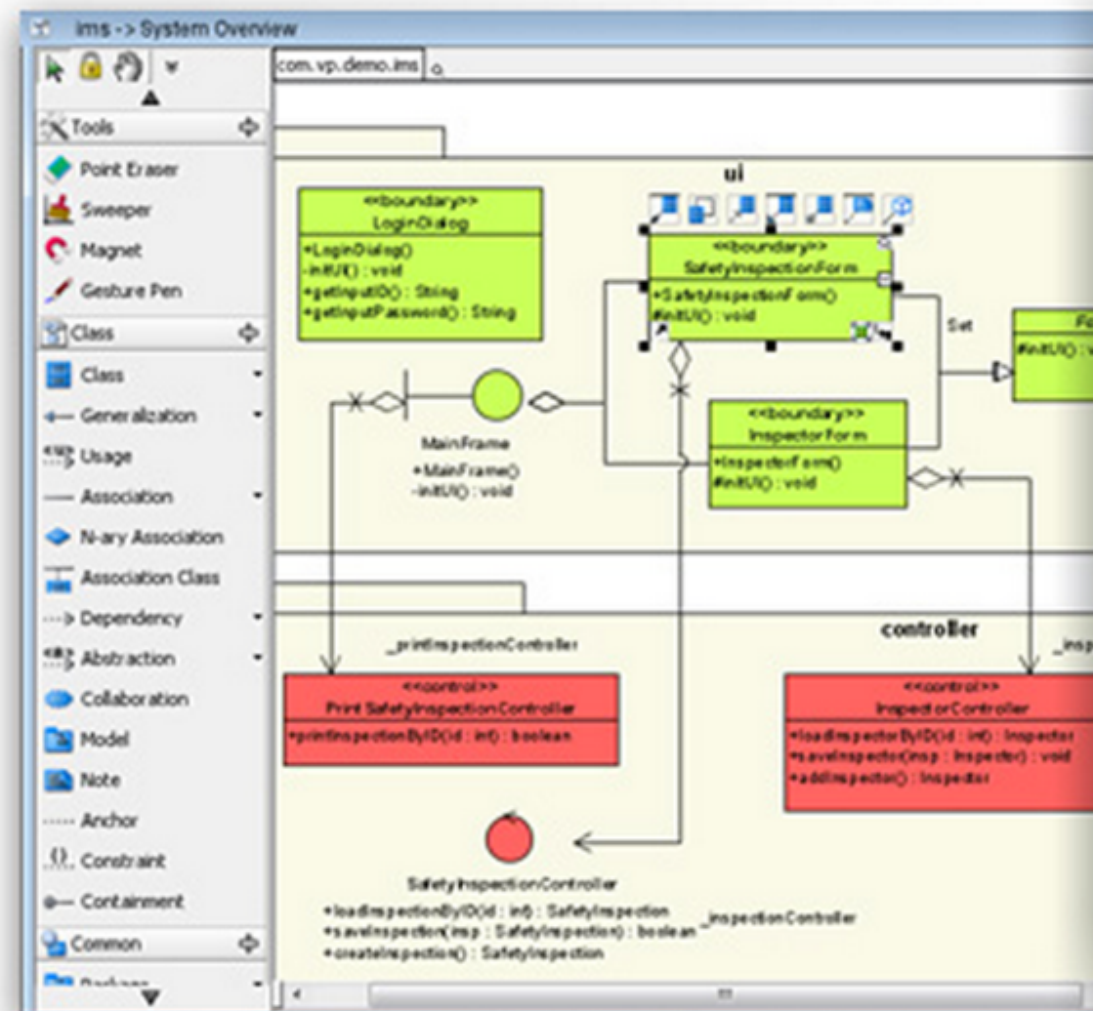
Structural Modeling

Behavioral

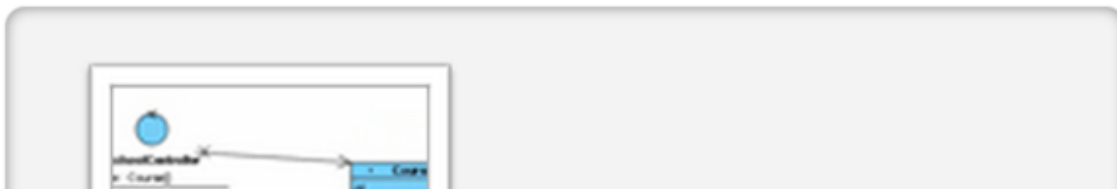
Architectural

## Class diagram

Class diagram is the most widely used diagram in modeling object-oriented system. A class diagram is an important diagram in software development. It represents the static aspect of system with classes, interfaces, associations and generalizations. Package is commonly used model element for organizing elements in class diagram. Class diagrams are not just for visualizing and documenting structure models but also for constructing executable system with **forward, reverse and round-trip engineering**. There is also a **synchronization engine** for generating and updating **entity relationship diagram** from class diagram.



*Class diagram sample shows classes, associations and generalizations*



# Behavioural Modeling

Use Case

Structural

Behavioral Modeling

Architectural

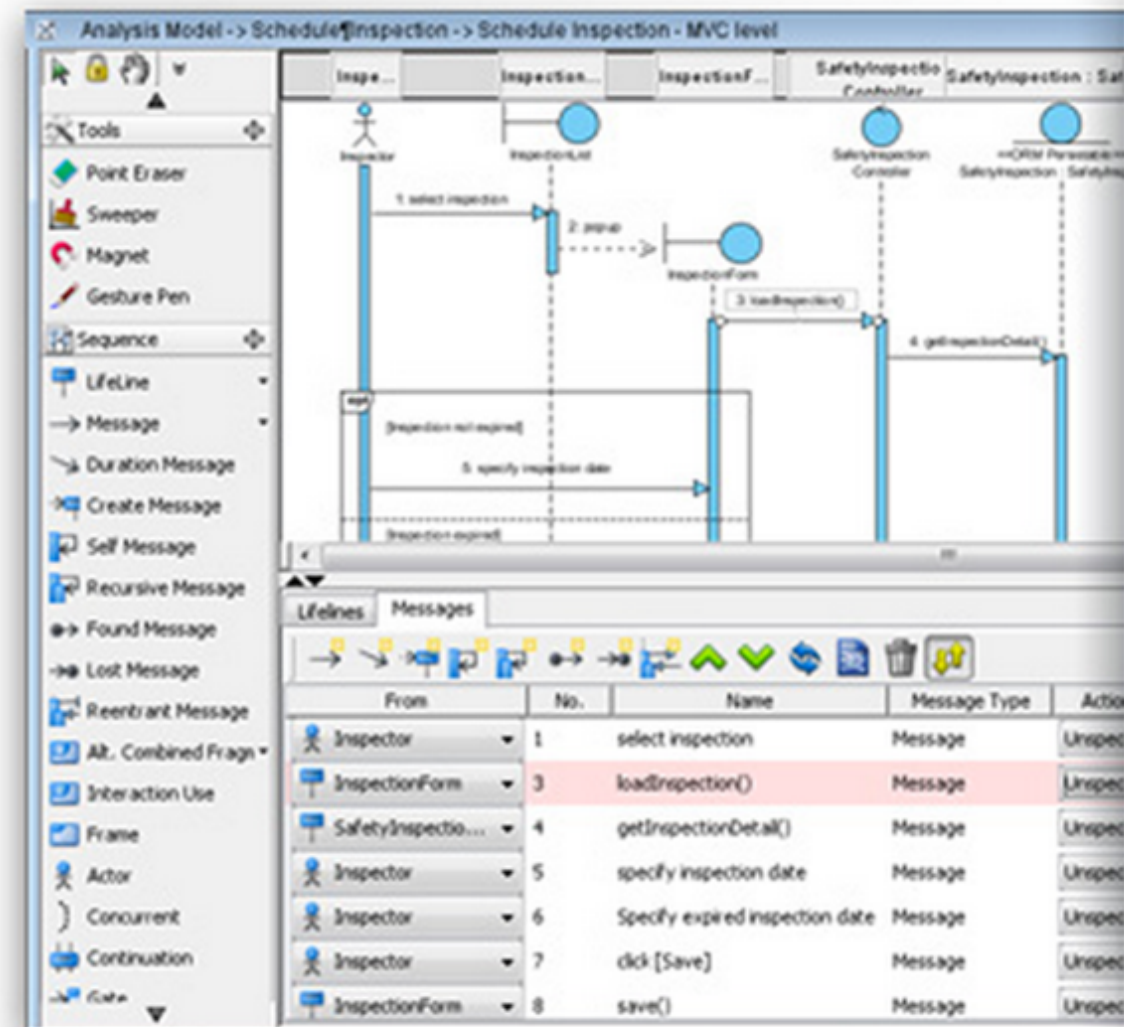
## Sequence diagram

Sequence diagram shows the interaction between users, systems and sub-systems, and emphasize the ordering of time of messages. You can draw sequence diagram solely by mouse or with keyboard shortcuts.



### Tutorial

- Draw sequence diagram with keyboard
- Constructing sequence diagram with existing classes
- Different ways of numbering sequence



Sequence diagram sample with keyboard control panel



# Architectural Modeling

Use Case

Structural

Behavioral

Architectural Modeling

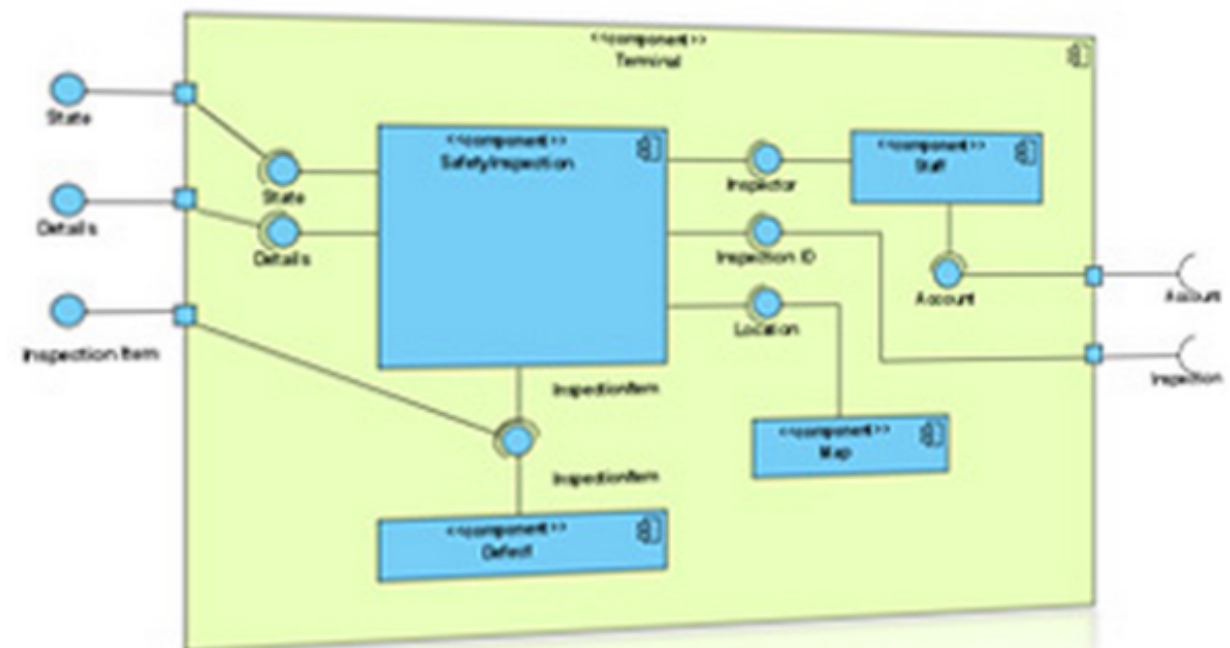
## Component diagram

Component diagram shows the physical aspect of an object-oriented software system. Component diagram illustrates the architectures of the software components and dependencies between them. That's why it is commonly used in software development.



### User's Guide

- [Drawing component diagrams](#)



*Component diagram sample shows objects and interfaces*

# Structural Modeling

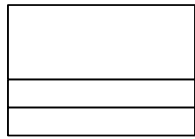


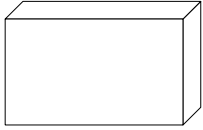
---

- Core concepts
- Diagram Types

# Structural Modeling Core Elements




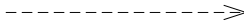
---

- a view of an system that emphasizes the structure of the objects, including their classifiers, relationships, attributes and operations.

Construct	Description	Syntax
<b>class</b>	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
<b>interface</b>	a named set of operations that characterize the behavior of an element.	
<b>component</b>	a modular, replaceable and significant part of a system that packages implementation and exposes a set of interfaces.	
<b>node</b>	a run-time physical object that represents a computational resource.	

# Structural Modeling: Core Relationships

---

Construct	Description	Syntax
<b>association</b>	a relationship between two or more classifiers that involves connections among their instances.	
<b>aggregation</b>	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
<b>generalization</b>	a taxonomic relationship between a more general and a more specific element.	
<b>dependency</b>	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

# Structural Diagram Tour

---

- Show the static structure of the model
  - the entities that exist (e.g., classes, interfaces, components, nodes)
  - internal structure
  - relationship to other entities
- Do not show
  - temporal information
- Kinds
  - static structural diagrams
    - class diagram
    - object diagram
  - implementation diagrams
    - component diagram
  - deployment diagram



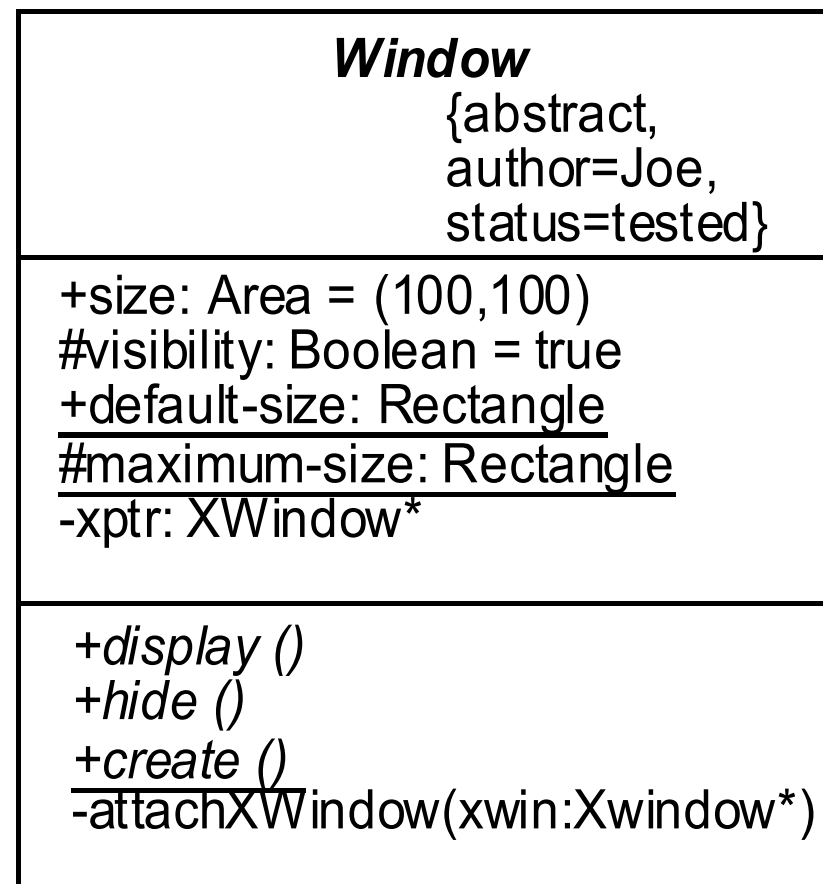
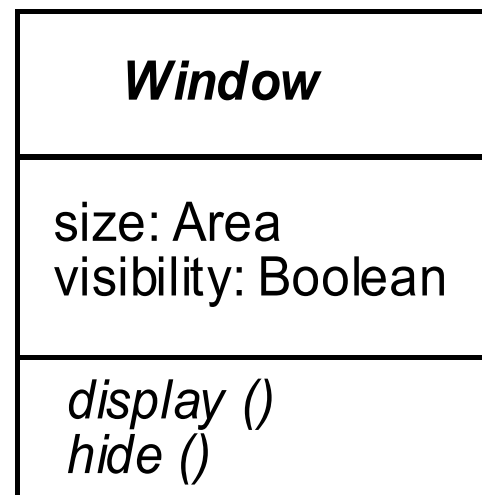
# Static Structural Diagram Examples

---

- Shows a graph of classifier elements connected by static relationships.
- kinds
  - class diagram: classifier view
  - object diagram: instance view

# Classes

---



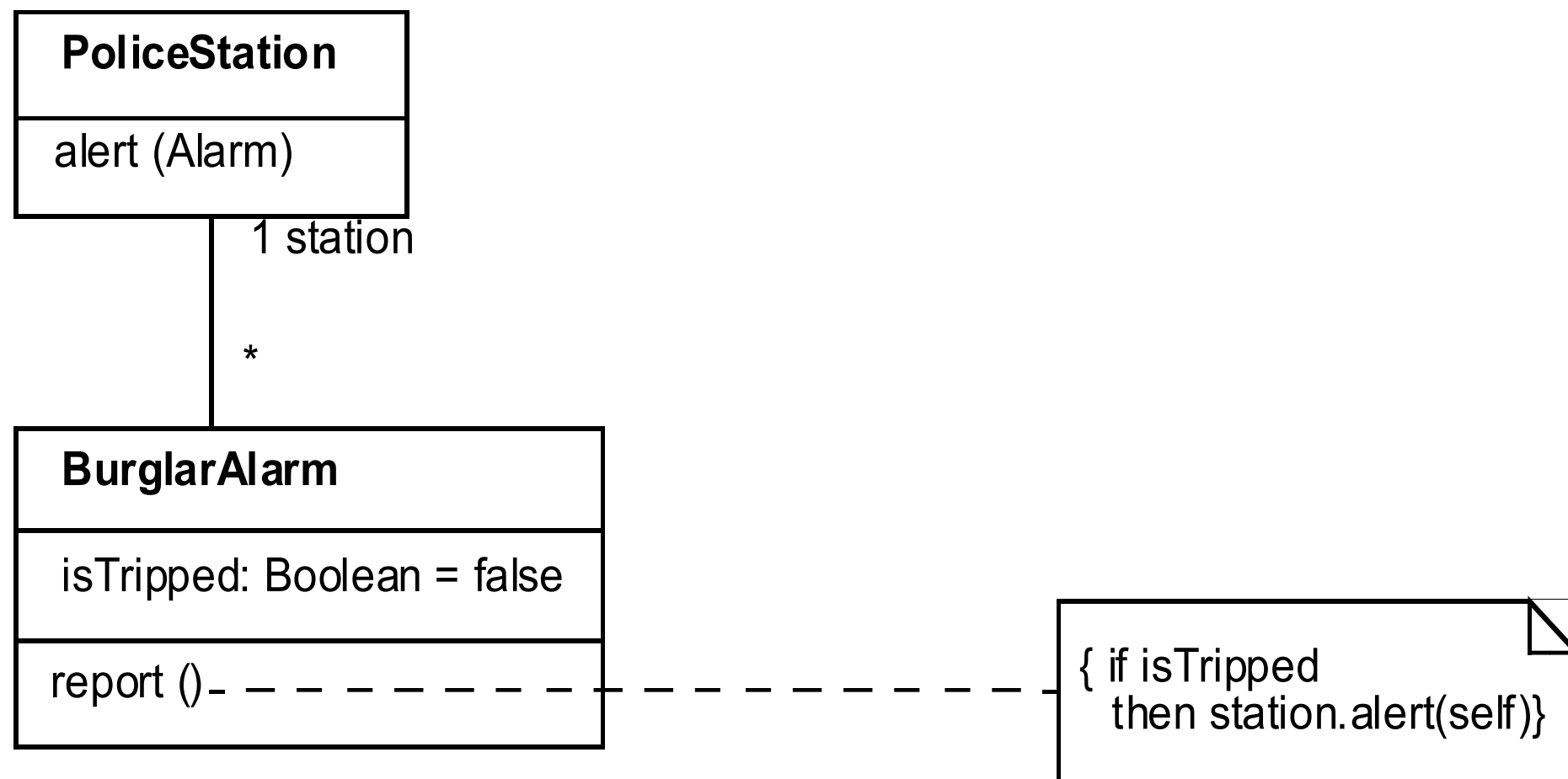
# Classes: Compartments with Names

---

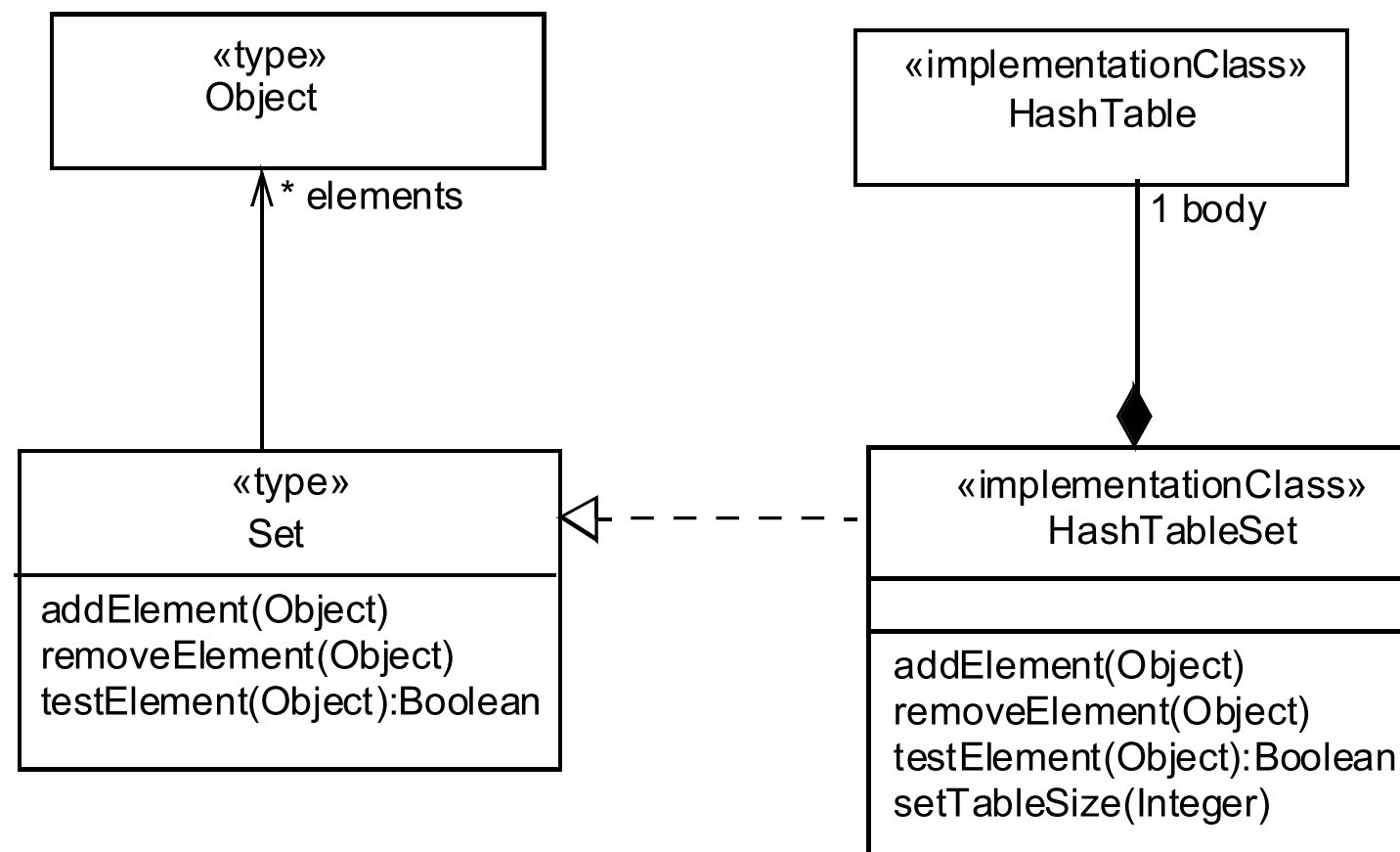
<b>Reservation</b>
<b>operations</b> guarantee() cancel () change (newDate: Date)
<b>responsibilities</b> bill no-shows match to available rooms
<b>exceptions</b> invalid credit card

# Classes: method body

---



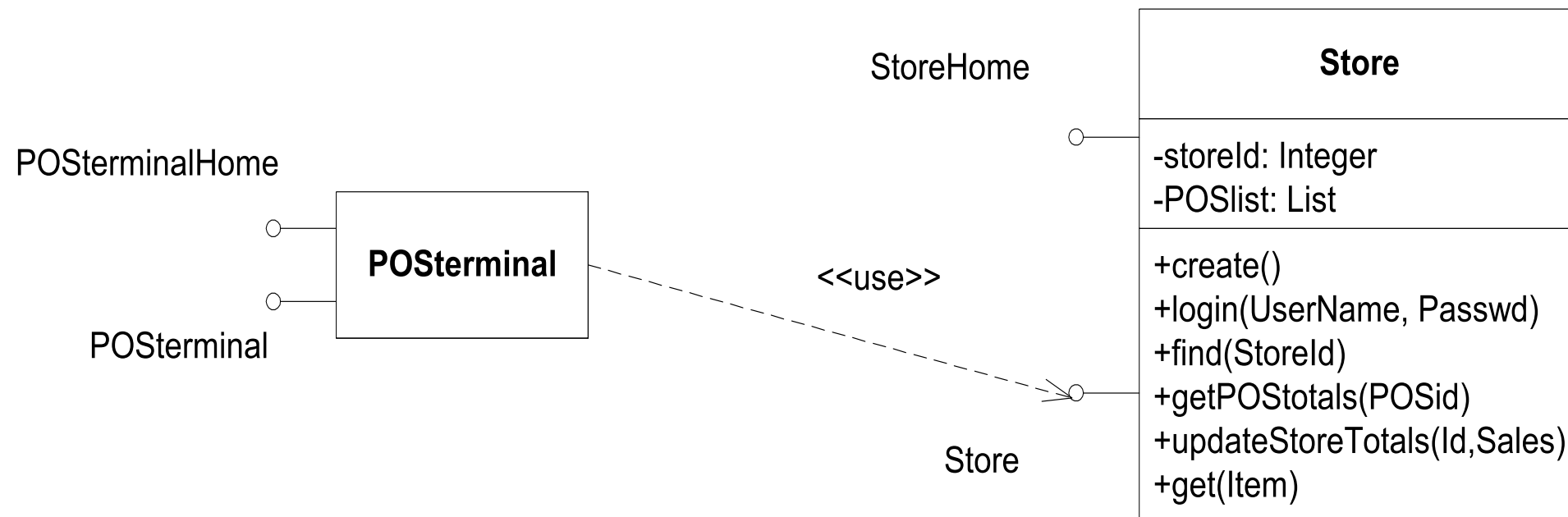
# Types & Implementation Classes





# Interfaces: Shorthand Notation

---



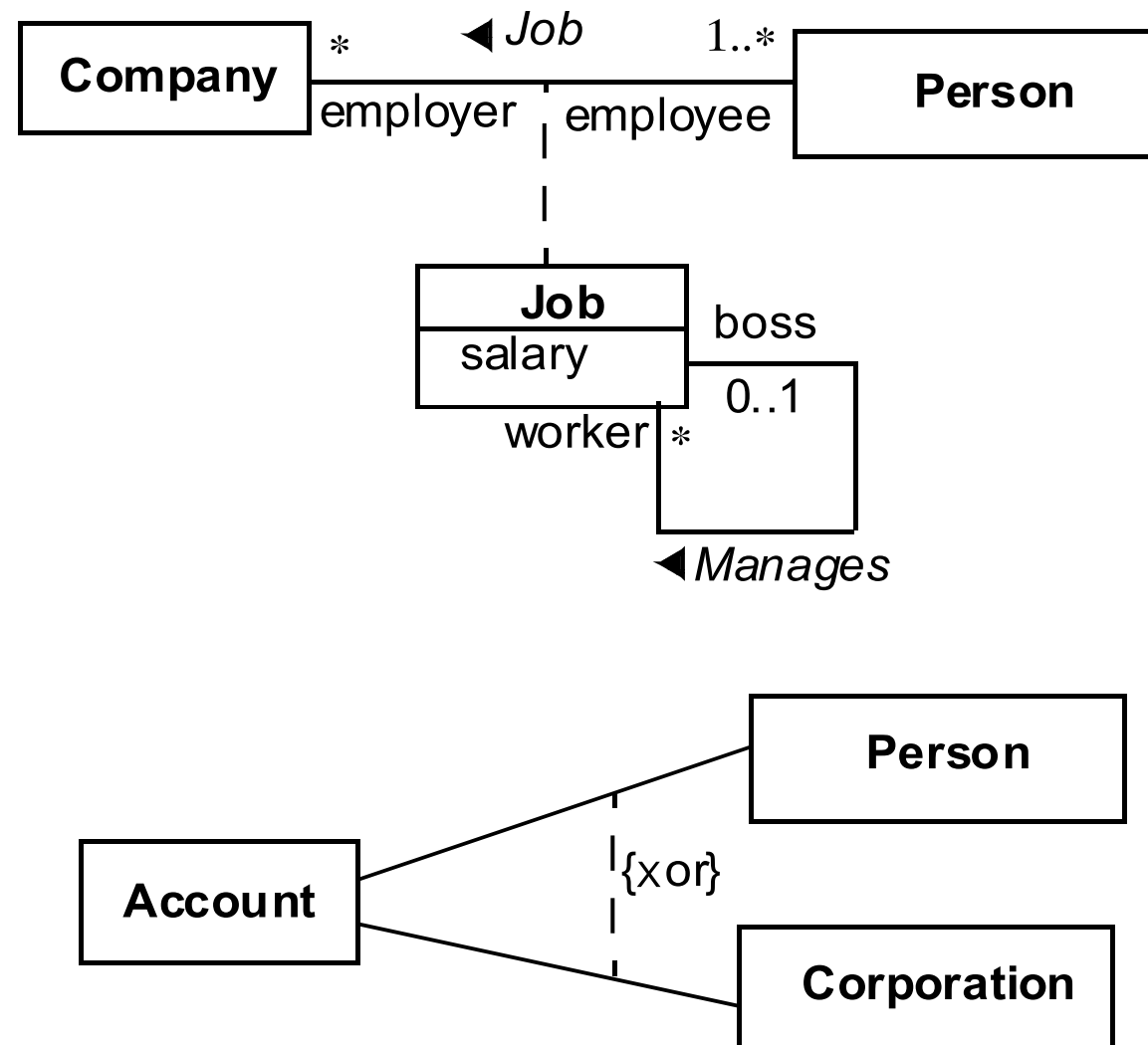
# Interfaces: Longhand Notation

---



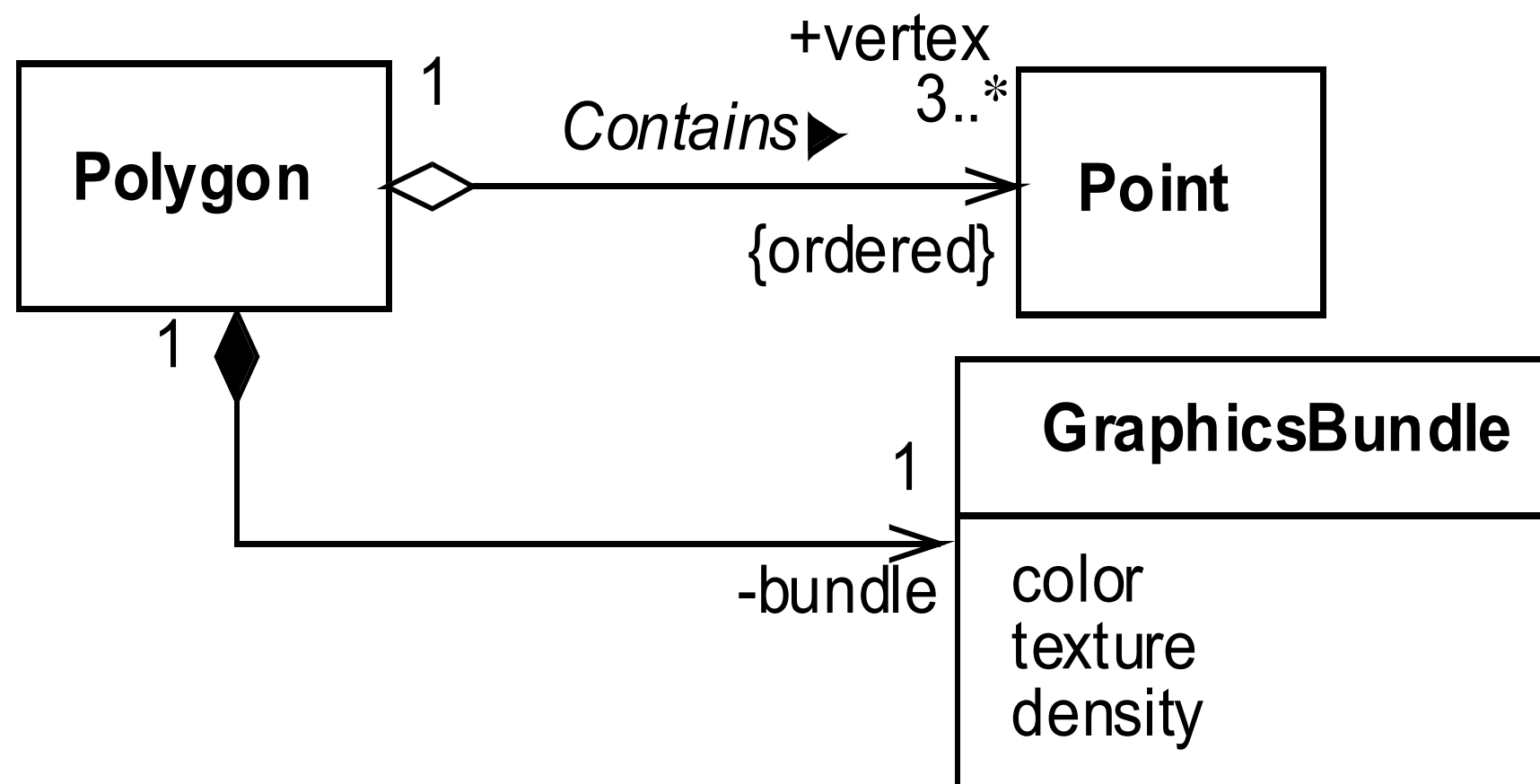
# Associations

---



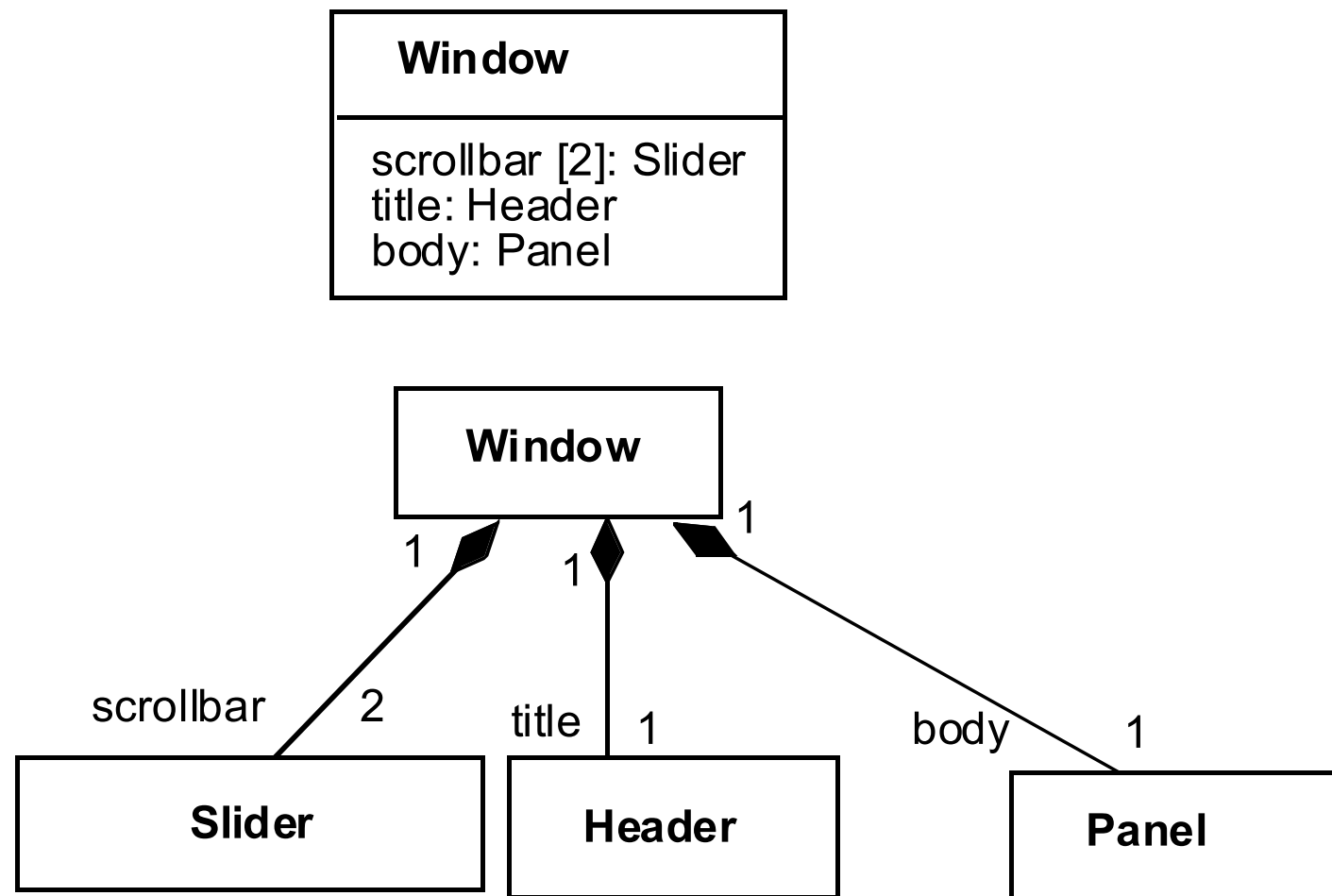
# Association Ends

---



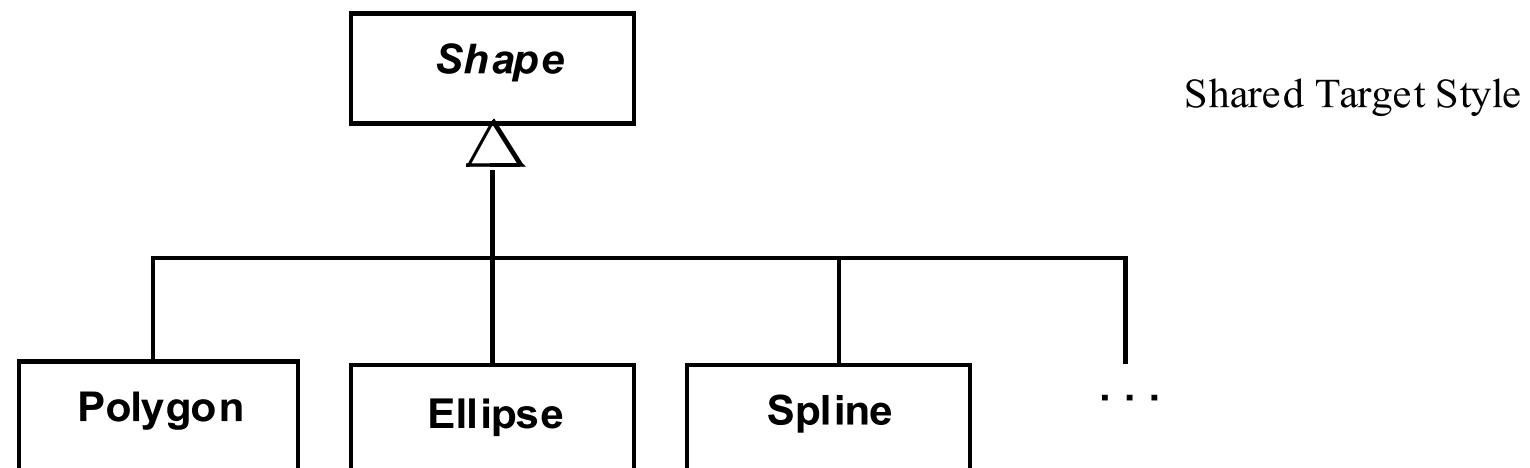
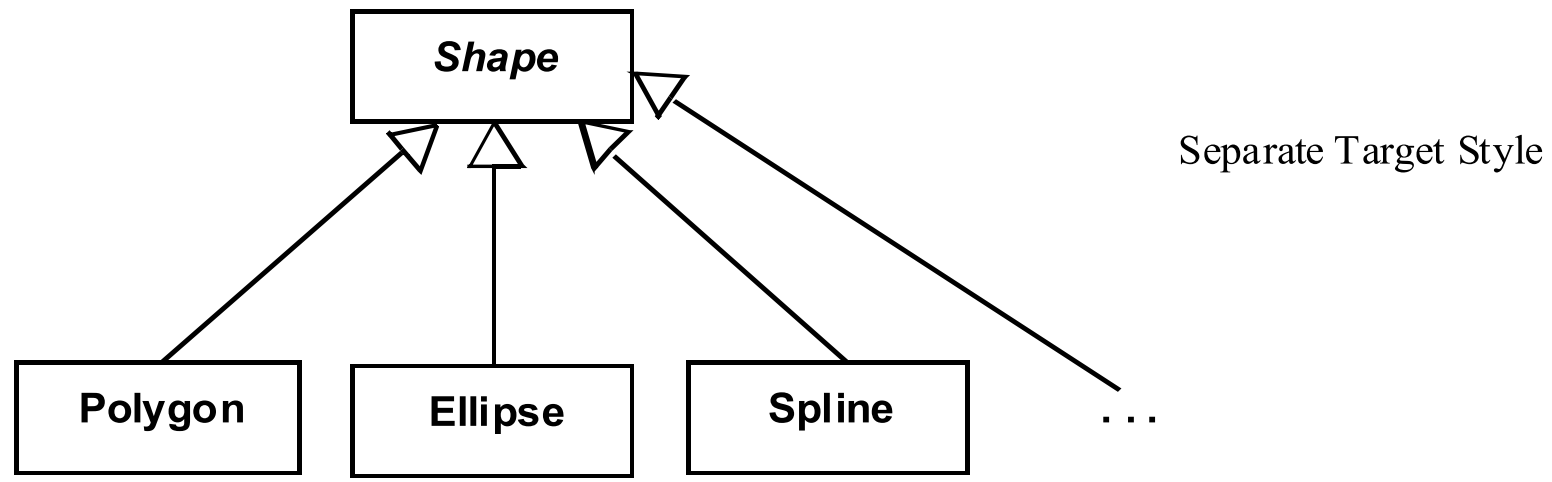
# Composition

---



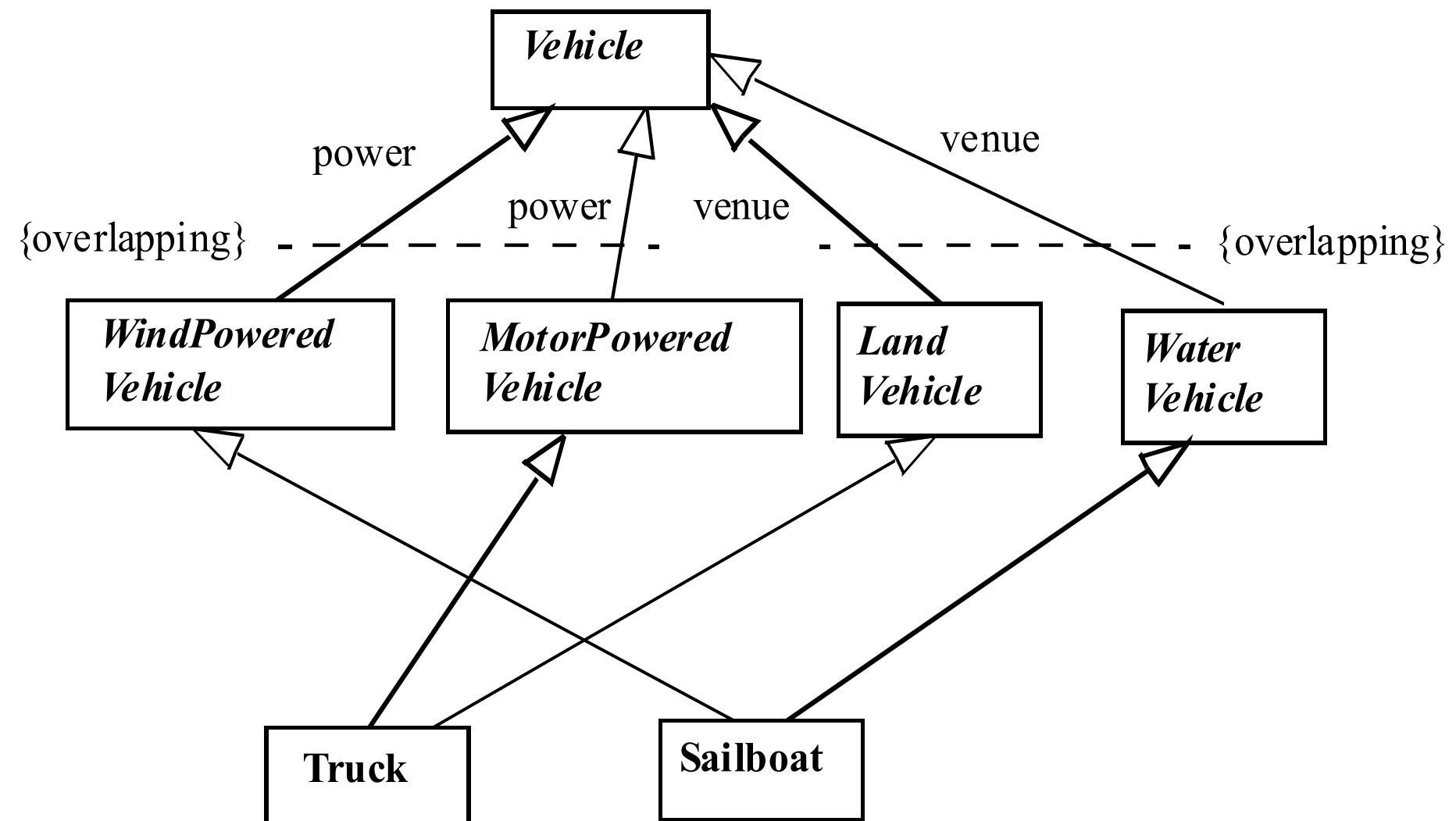
# Generalization (Inheritance)

---



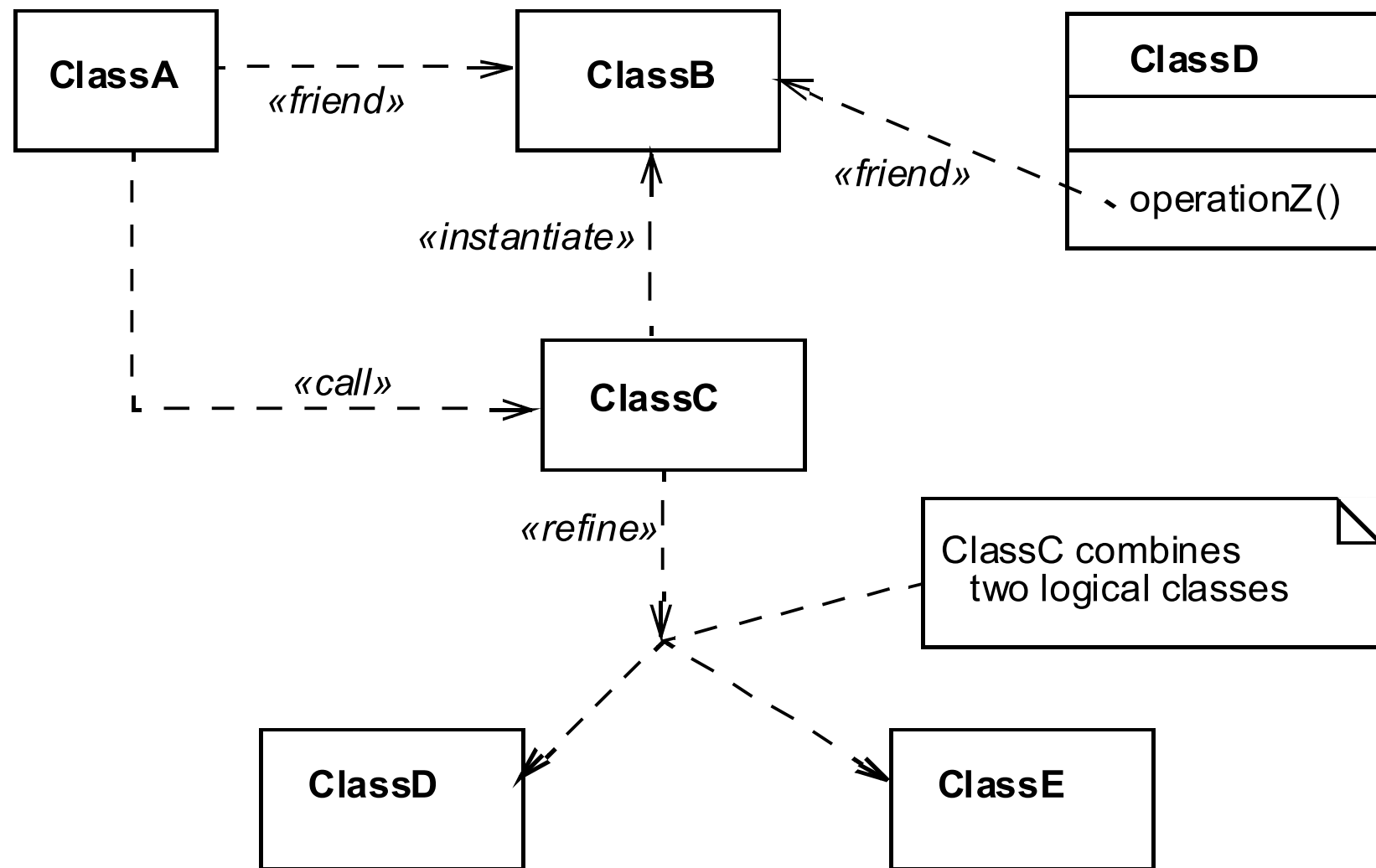
# Generalization (Inheritance)

---



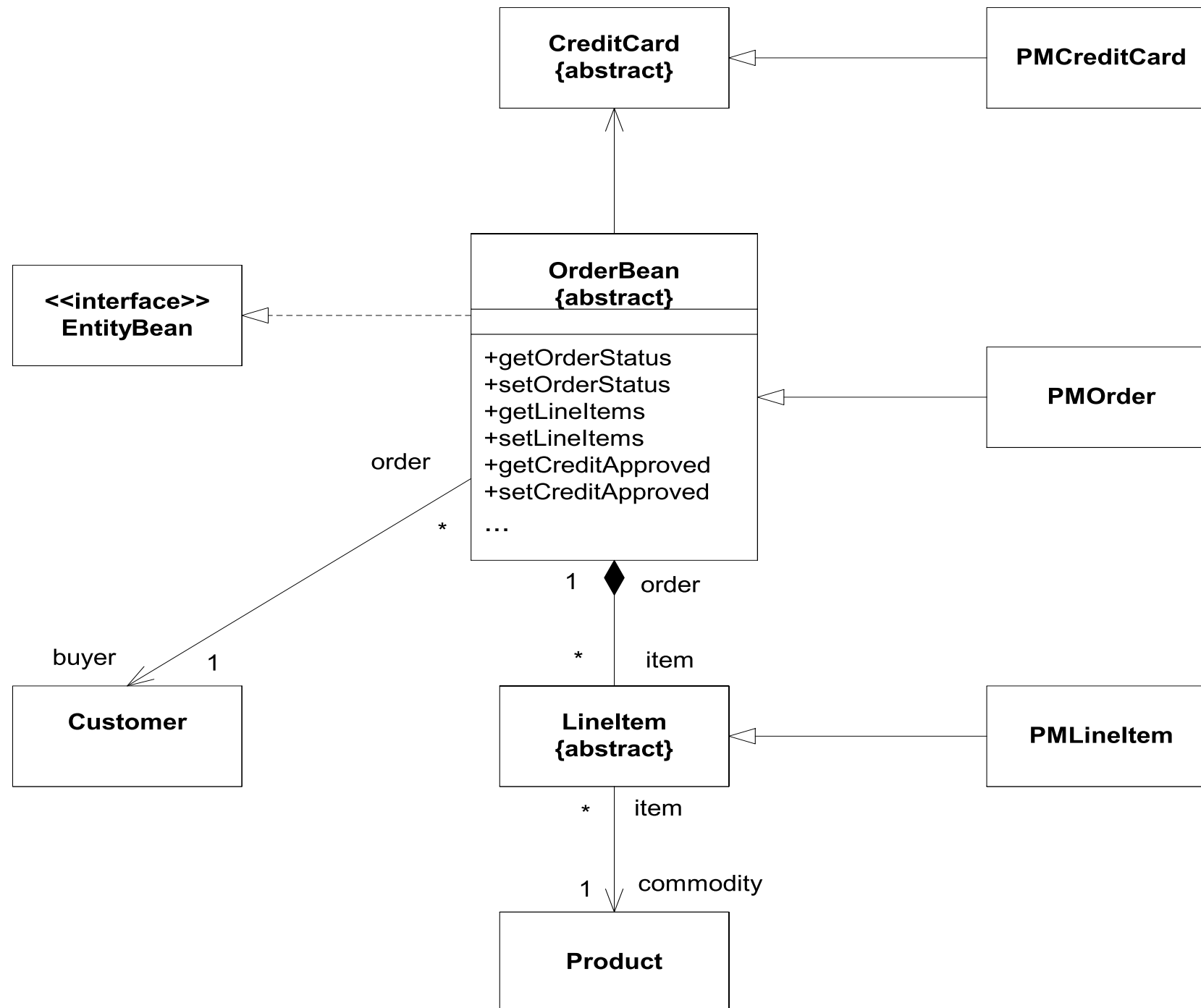
# Dependencies

---





# Class Diagram Example



Some of these slides were adapted from a presentation by Cris Kobryn  
Co-Chair UML Revision Task Force  
+ the Visual Paradigm Online Help

<http://www.visual-paradigm.com/product/vpuml/provides/umlmodeling.jsp>

© 1999-2001 OMG and Contributors: Crossmeta, EDS, IBM, Enea Data, Hewlett-Packard, IntelliCorp, Kabira Technologies, Klasse Objecten, Rational Software, Telelogic, Unisys



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

