**IDEA Progetto**

Design Thinking ; Requisiti ; Kanban ; Pitch

D1

**Implementazione**

Git / GitHub
User Stories
RESTful API
OpenAPI
Web 2.0 JavaScript
WebAPI Node.js
MongoDB
Authentication JWT + GoogleAuth
Frontend
Deplyment & CI-CD
Testing Jest

D3

**Analisi e Progettazione**

Processi di sviluppo
Agile
Linguaggi di modellazione
Use Case Diagram
Sequence + Activity Diagram
Architetture
Component Diagram
**Class Diagram**
Class Diagram -> API
Testing

D2

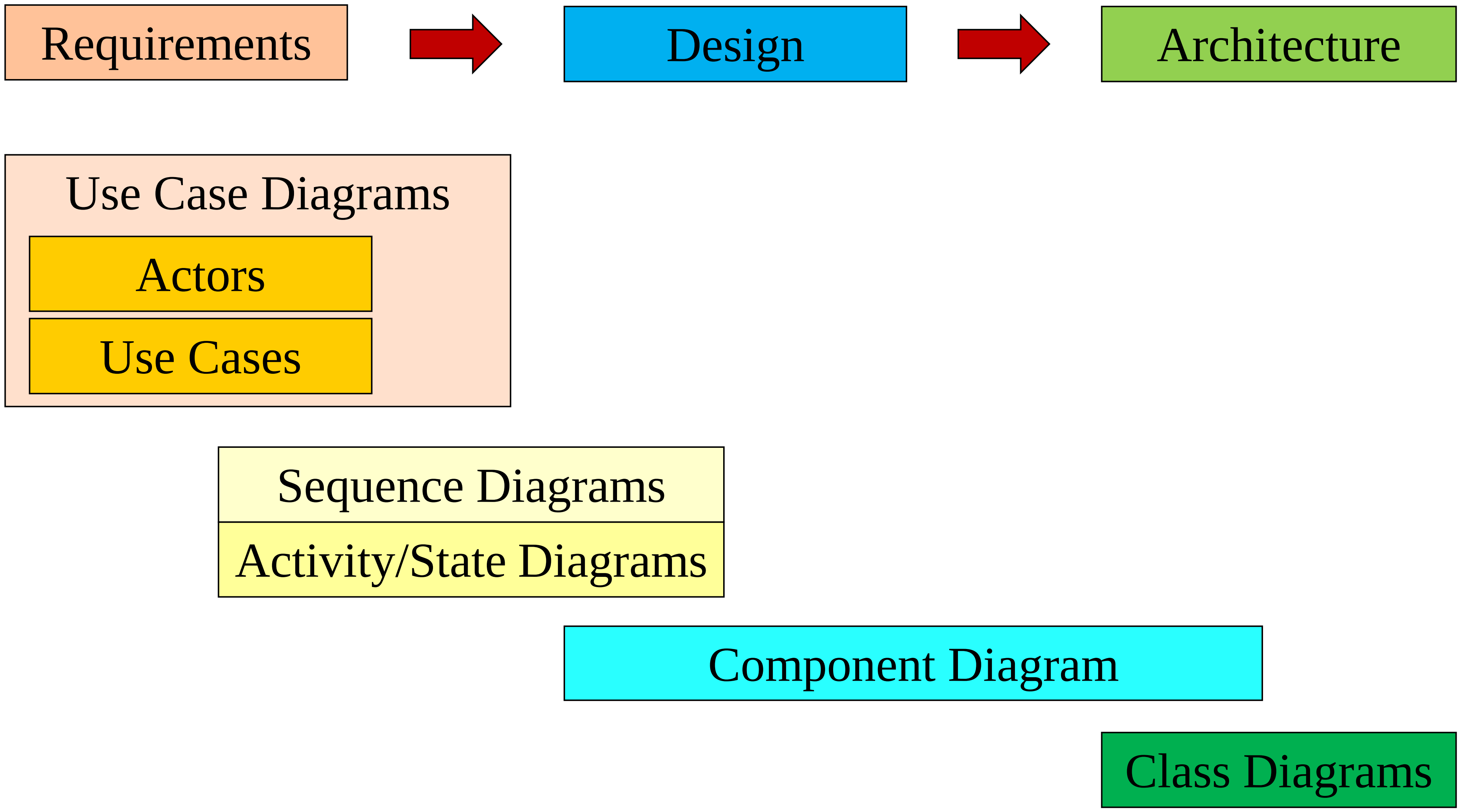
**Report finale**

D4

# Software Engineering

## Architectural Design

# Diagrams in our Process



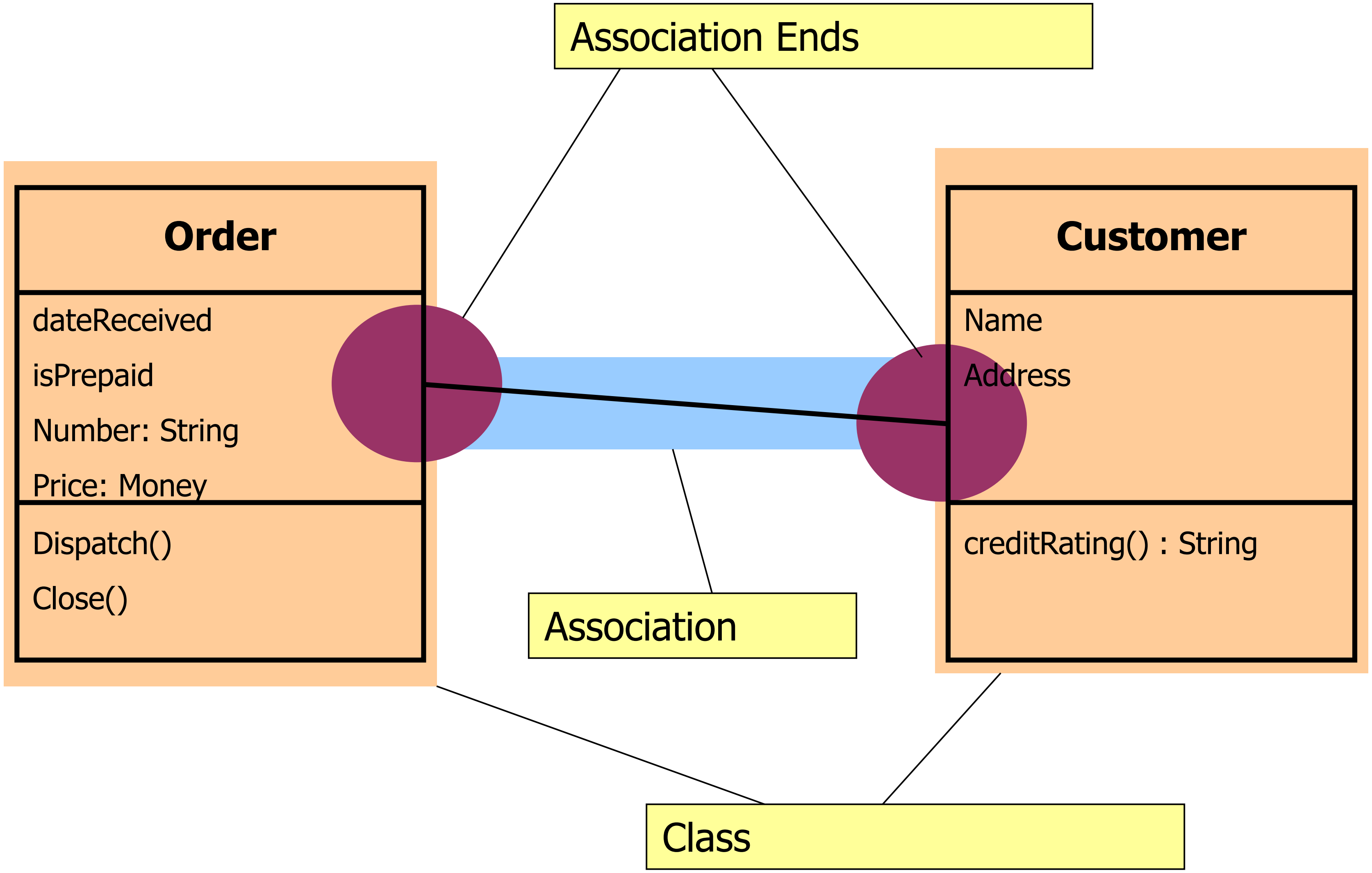
## Architectural Design

- **Goal**
  - Define the architecture (structure) of the system
- **Output**
  - Architectural Specification Document (possibly containing various **UML** diagrams)
- **“Main” Diagram for Architectural Design:**
  - Class Diagrams

# Software Engineering

## Class Diagrams

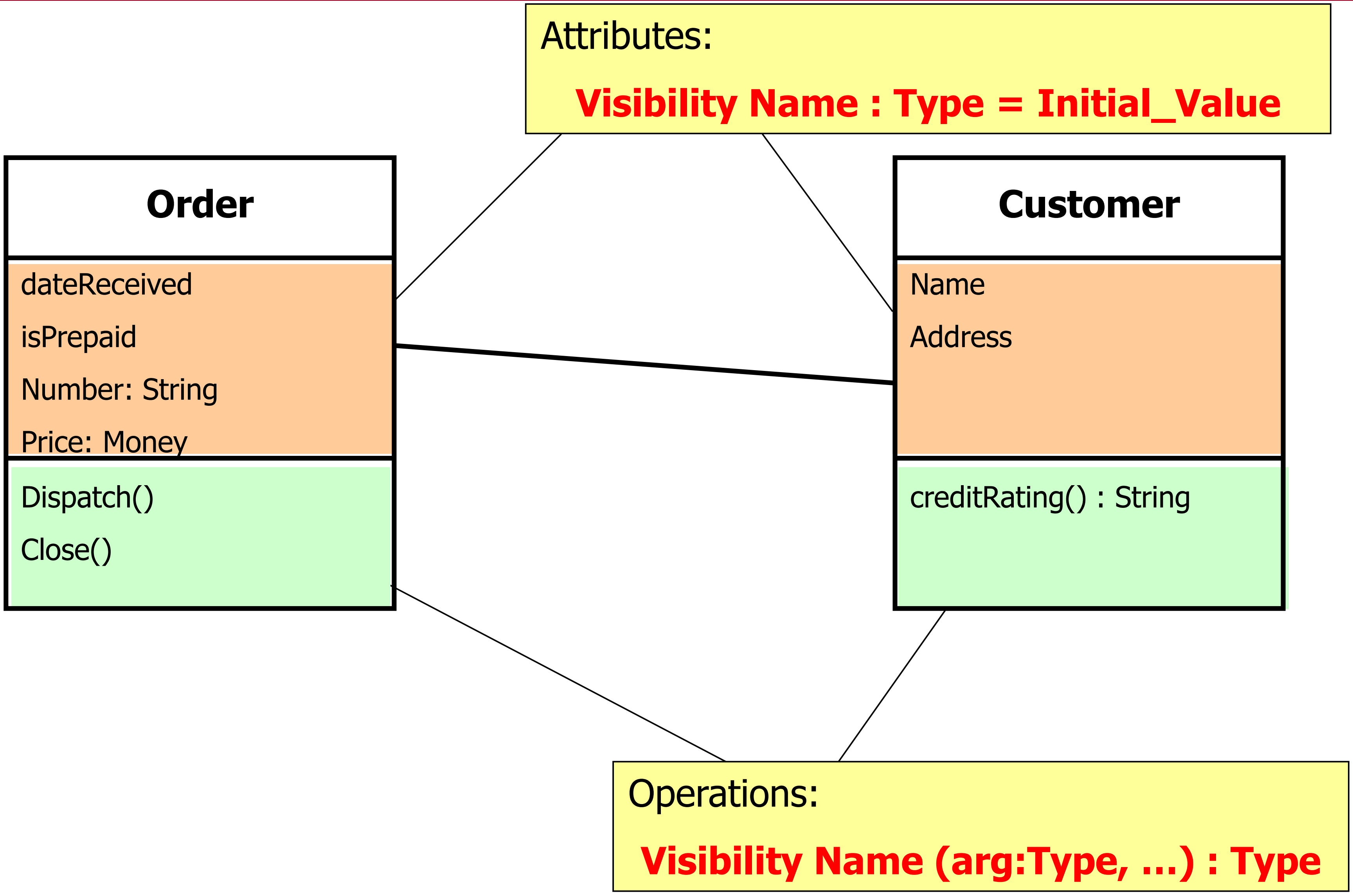
# Class Diagram: Example



## Class Diagram Elements

- Classes
  - Name: obvious
  - Attributes: identify characteristics of the class
  - Operations: list operations (methods) of the class
- Association
  - Name: association name
- Association Ends
  - Navigability: allows navigation toward the class pointed by
  - Roles: role played by the class attached to the association end
  - Multiplicity: range of allowable cardinalities a set may assume

## Classes

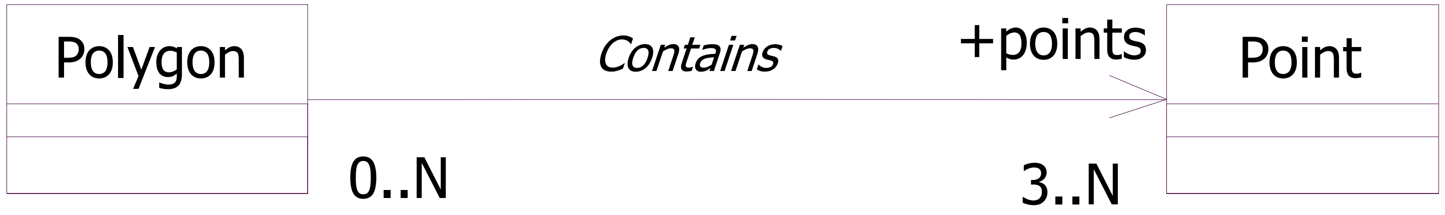

Attributes:
Visibility Name : Type = Initial_Value
Order
dateReceived
isPrepaid
Number: String
Price: Money
Dispatch()
Close()
Customer
Name
Address
creditRating() : String
Operations:
Visibility Name (arg:Type, ...) : Type

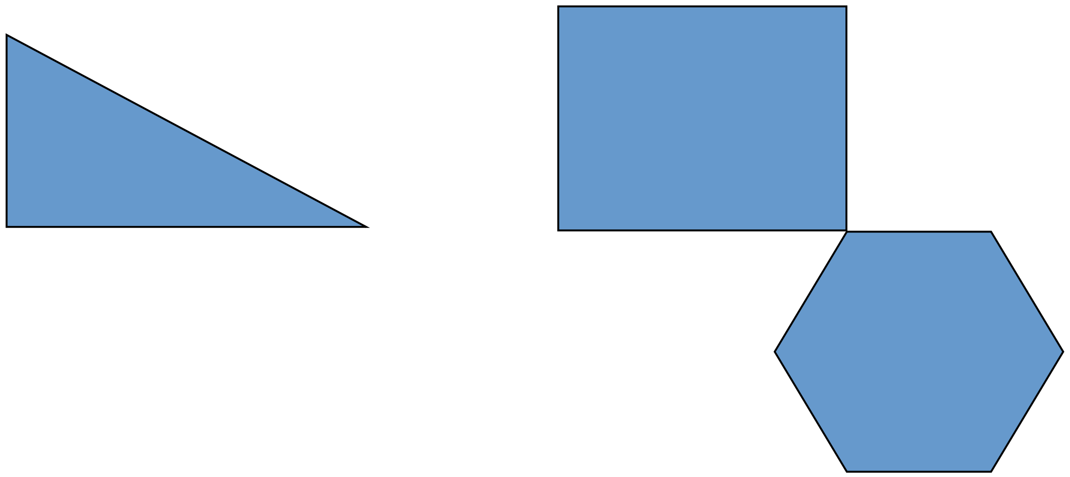
## Classes

- Attributes
  - visibility name [multiplicity] : type-expression = initial-value {property string}
- Operations
  - visibility name (arguments) : type-expression = initial-value {property string}
- Visibility

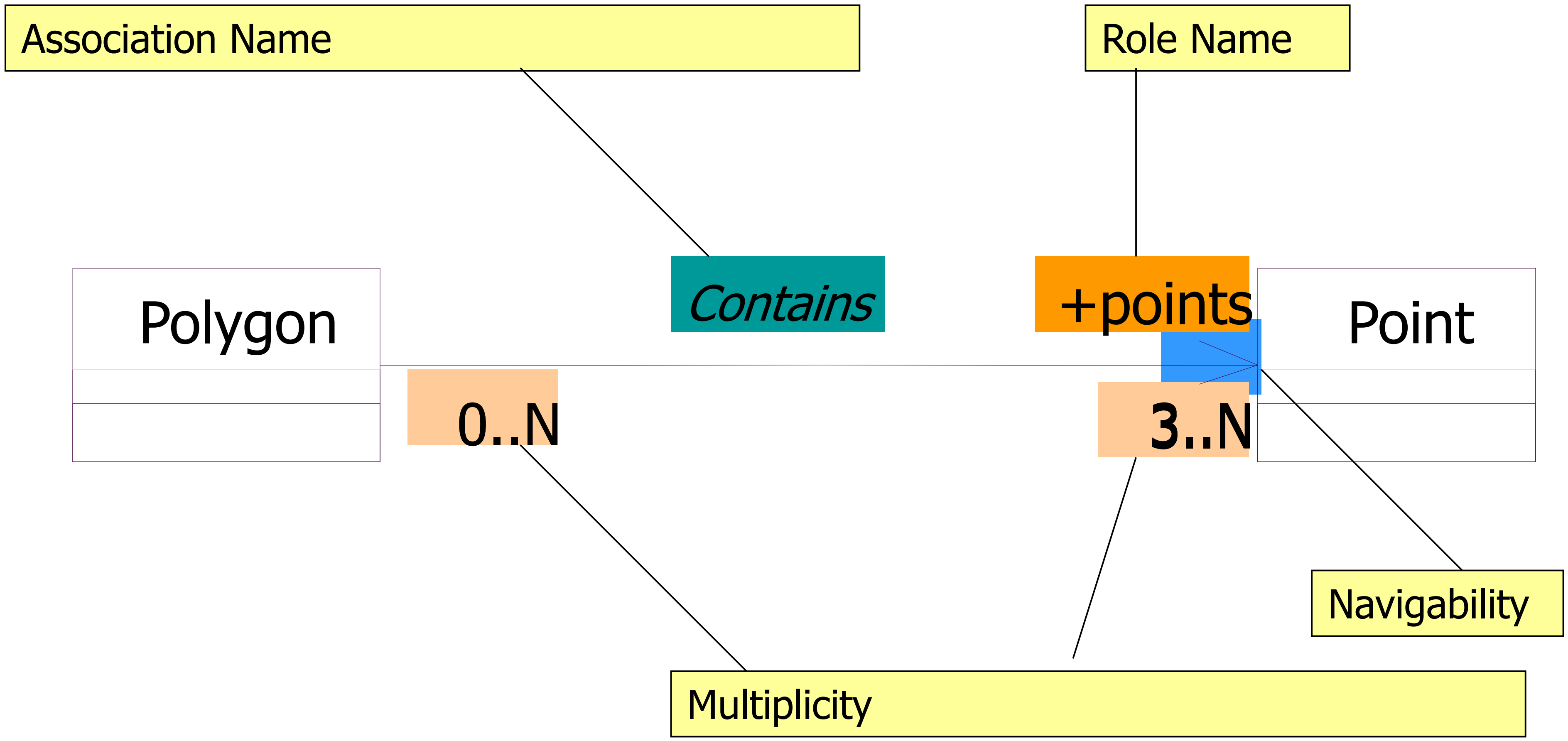
  + public # protected - private

## Association





## Association



## Association

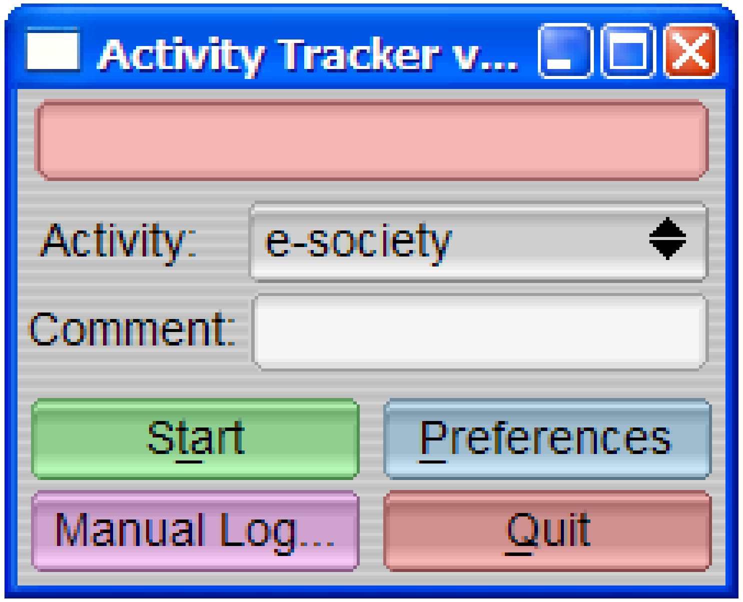
- Navigability
  - No navigability implies bi-directional **or** unspecified navigability (try it in Rose)
- Association Names vs Roles
  - Association Name: Name of the association
  - Role Name: Role a class play in the association
- Multiplicity
  - 0..1 or 1 or h..N or N or $*$ (:=0..N)
  - Multiplicity always refers to a single instance of the class attached to the other end of the association

## Software Engineering

### Class Diagram: example

# Example: Time Tracker

We have been contacted by a small firm. They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports.



The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

## Step 1: Identify Classes

Candidates for classes are objects, concepts, and specific terms that can be found in the Requirement Documents

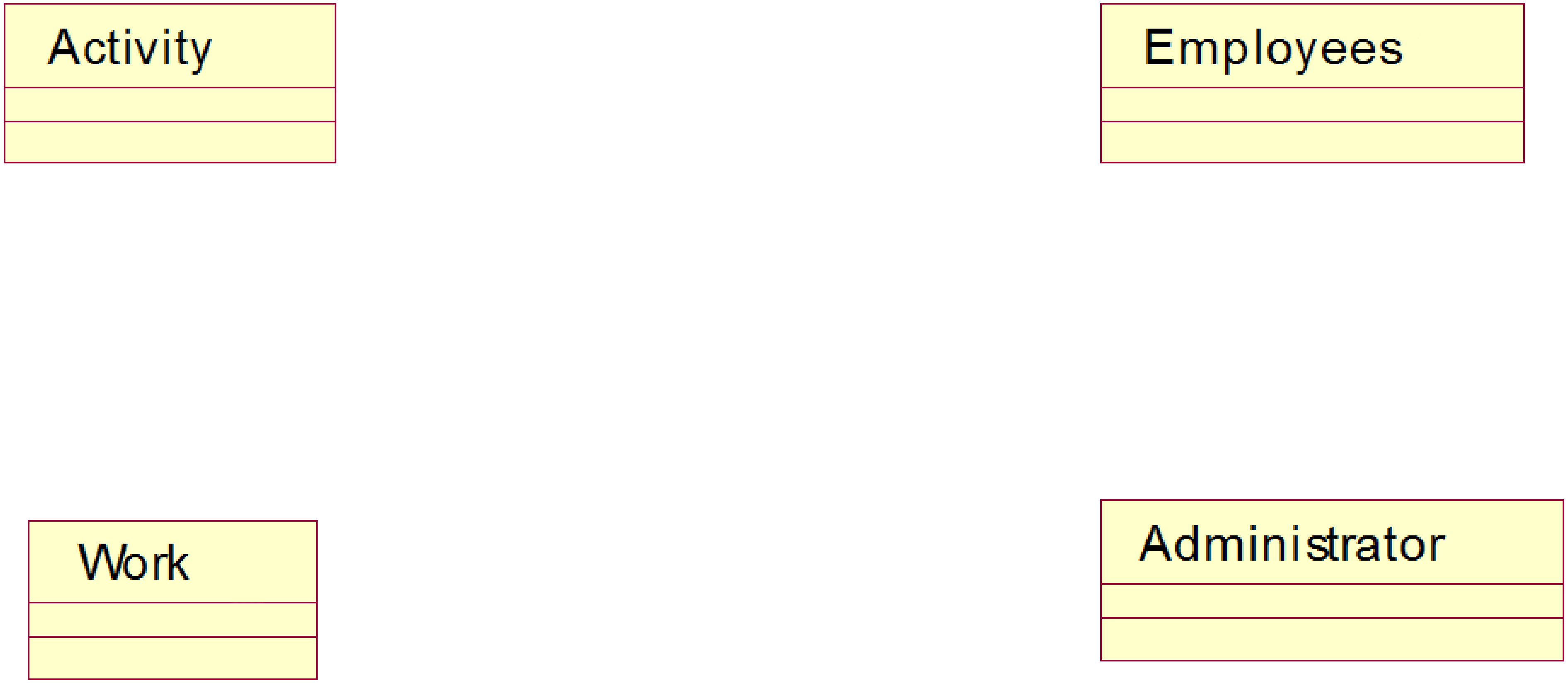
## Step 1:Time Tracker

We have been contacted by a small software firm.

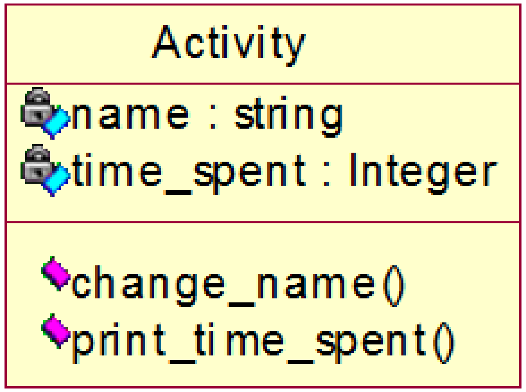
They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

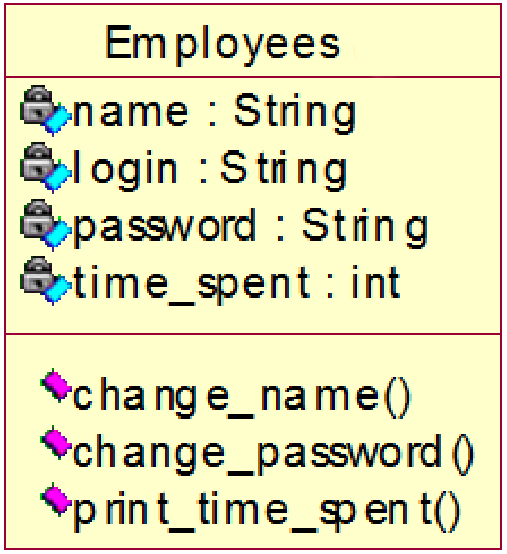
The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

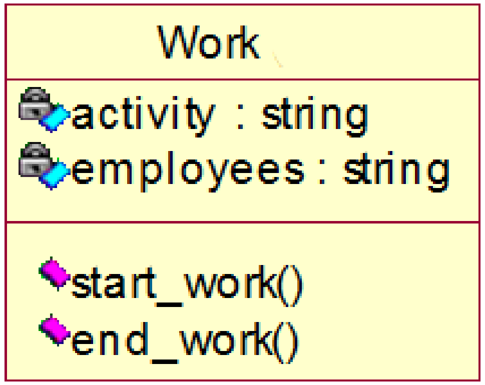
## Step 1: Identify Classes

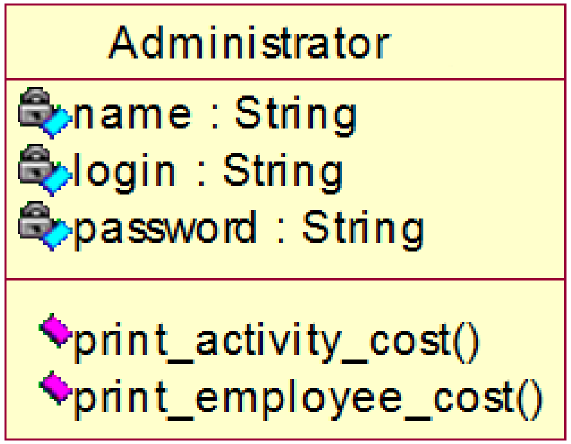


## Step 1: Identify Classes (ctd)









## Step 2: Establish Associations

**Associations are established by looking at the description of objects, concepts, and specific terms found in the Requirement Documents**

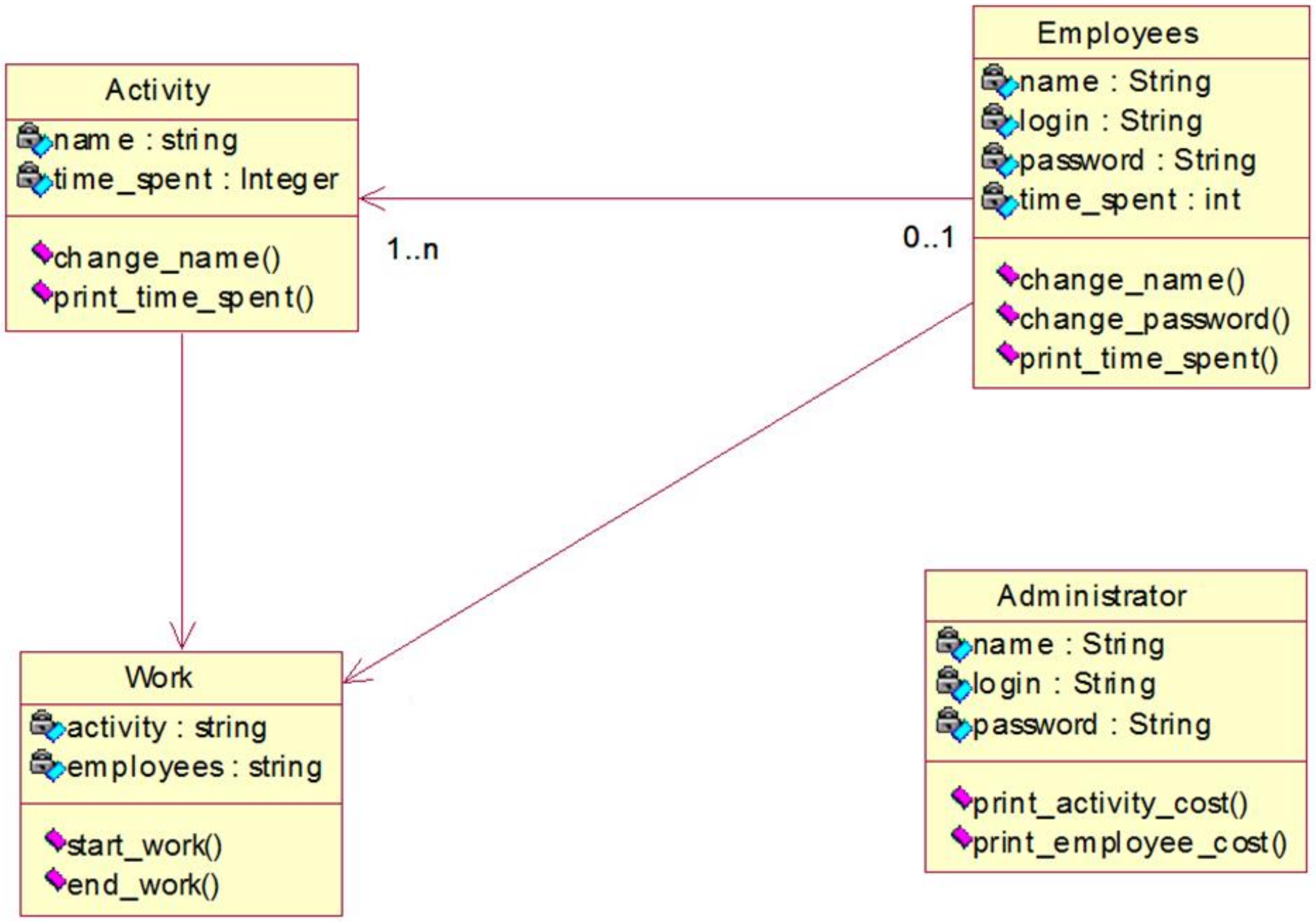
## Step 2: TimeTracker

We have been contacted by a small software firm.

They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

## Step 2: Establish Associations



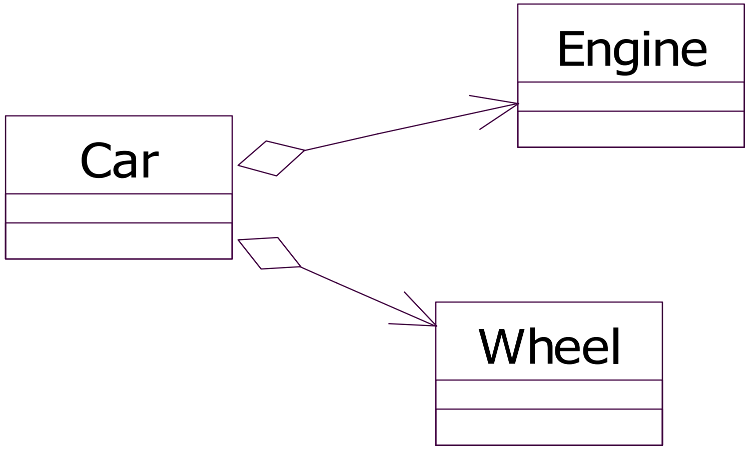
# Software Engineering

## Class Diagrams: Advanced Concepts

# Aggregation & Composition

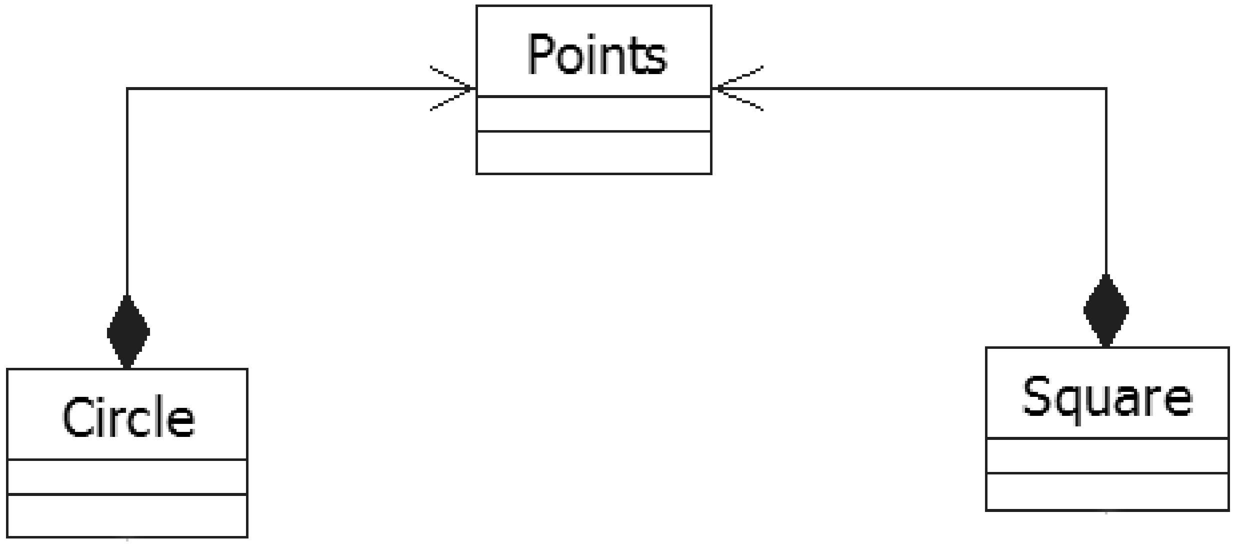
**Aggregation:** A type of association used to represent "Part Of" relationship

**Example:** "Engine" is a "part of" a "Car"



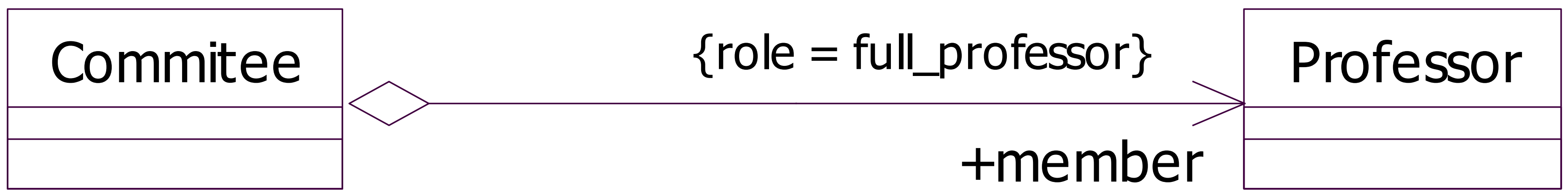
## Aggregation & Composition

**Composition:** a particular (stronger) type of aggregation: the “contained” objects exists and live only with the “container” class.



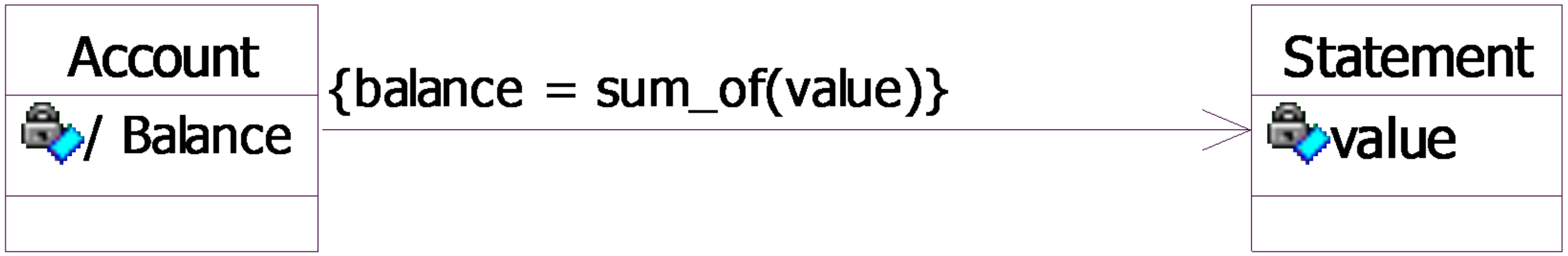
## Constraint

**Constraint:** expression of some semantic condition that must be preserved while the system is in a steady state



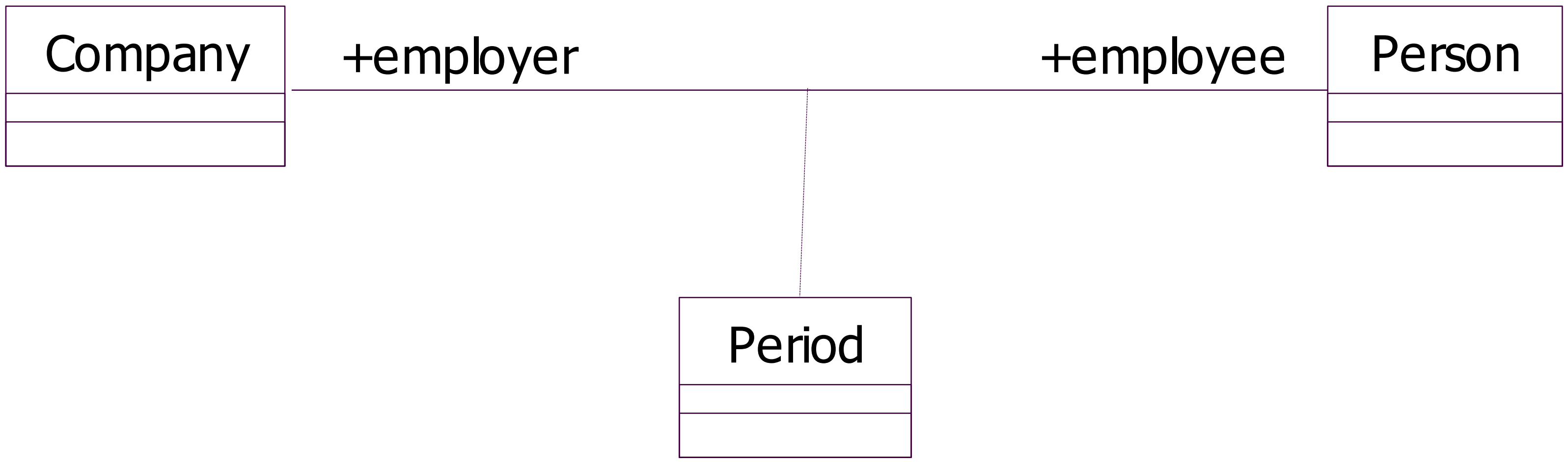
# Derived Attributes

**Derived Attributes:** help highlight information that can be derived from elements of the model



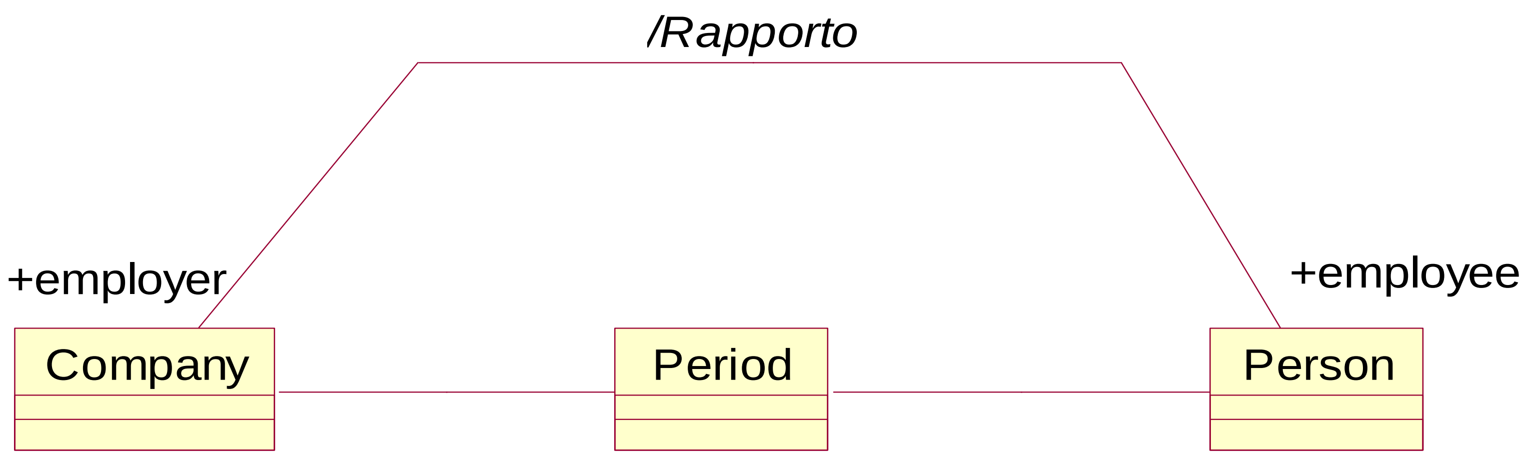
## Association Class

**Association Class**: enriches and further qualifies associations between classes



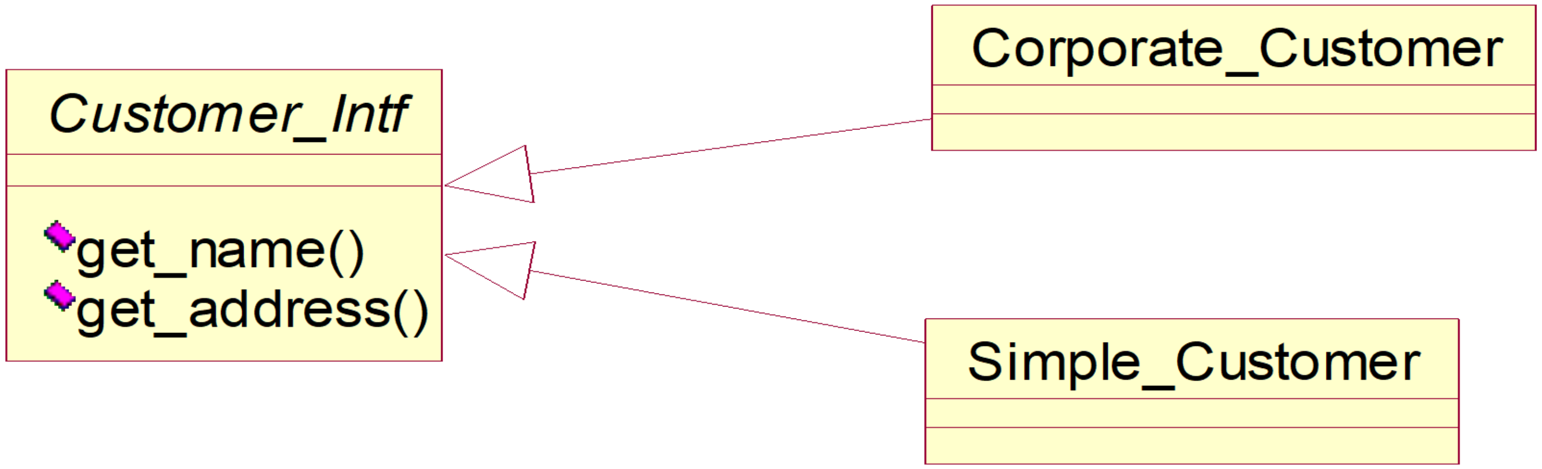
## Association Class (ctd)

Association Classes are not always necessary, but they may help simplify diagrams



## Abstract Class

**Abstract Class:** it provides a specification for an interface



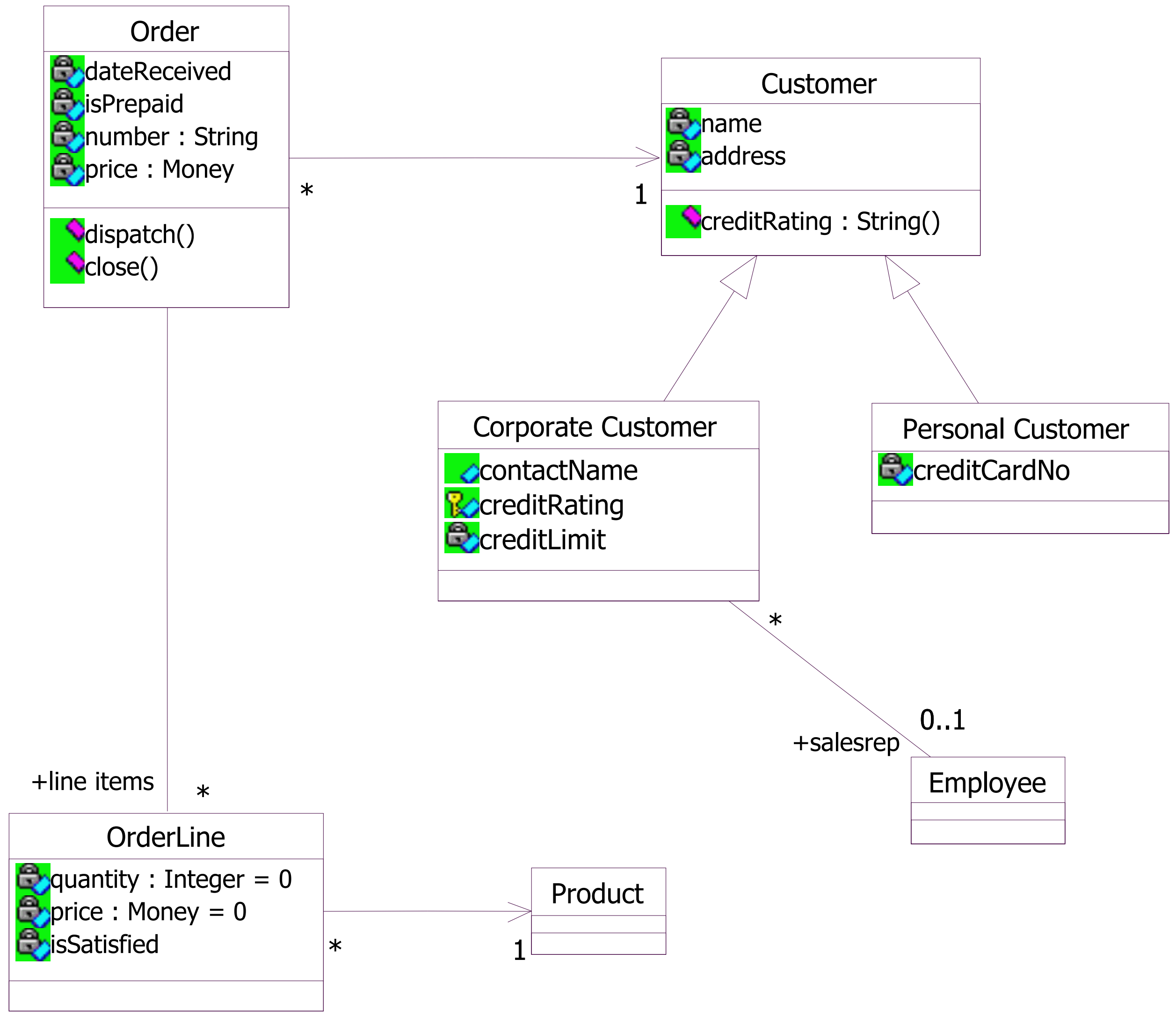
## Caveat!

Most of the times (if not always) you will be using the standard features of class diagrams.

Always question whether the features you are using in a class diagram are really useful:

- Do they help me simplify the diagram?
- Do they help me explain the diagram to other people?

# Class Diagram: one more example



# Restaurant Management System

**Scrivere un Class Diagram per il seguente progetto software:**

We have been asked to build a system to automate the ordering and billing activities of a restaurant. The system is distributed: waiters and waitresses are provided with handheld devices to take orders. The handheld devices communicate orders to the kitchen and to the cashier. The handheld devices receive real-time information about availability of the different items in the menu. Once placed, orders can be changed by the customers, within a time frame from the order (5 minutes) or after the time-out, if the corresponding order has not yet been processed by the Cook.
The system computes bills and is also used to manage reservations of tables. Reservations can either happen by phone or via the internet.