UML

Contents

[Useful Resources: 1](#_Toc423680708)

[Overview 1](#_Toc423680709)

[Benefits: 1](#_Toc423680710)

[Shortcomings: 2](#_Toc423680711)

[UML Class Diagrams 2](#_Toc423680712)

[Class Hierarchies in UML 3](#_Toc423680713)

[UML State Diagrams 4](#_Toc423680714)

[UML Use Case Diagrams 4](#_Toc423680715)

# Useful Resources:

* <http://www.uml.org>; (official page contains official specification, tutorials…)
  + <http://www.omg.org/technology/documents/formal/uml.htm>.
* <http://edn.embarcadero.com/article/31863> (useful tutorial)

# Overview

* Specify and document architecture of large object-oriented software systems, using diagrams
* UML 2.2 contains 13 diagrams including 3 categories:
  + Structure (represent static application structure)
    - Class diagrams
    - Object diagrams, Component diagrams, composite structure diagrams, package diagrams, package diagrams, deployment diagrams
  + Behaviour
    - Use case diagrams
    - Activity diagrams, state machine diagrams
  + Interaction of components
    - Sequence diagrams, communication diagrams, timing diagrams, interaction overview diagrams
* Tool support:
  + - Microsoft Visio Professional
    - Dia
  + Some Tools can generate
    - source code from UML diagrams,
    - generate diagrams from source code
    - generate test cass and test suites from UML diagrams

# Benefits:

* Standard way to communicate with other programmers
* good for Object-Oriented Programming, very common in industry
* an open standard (not owned by any company)
* visual language, easy to glance at
* agnostic in terms of software processes or development lifecycles

# Shortcomings:

* all syntax, no meaning
* contains redundant and infrequently-used constructs
* only works for OOP
* UML tools don’t play nicely together

# UML Class Diagrams

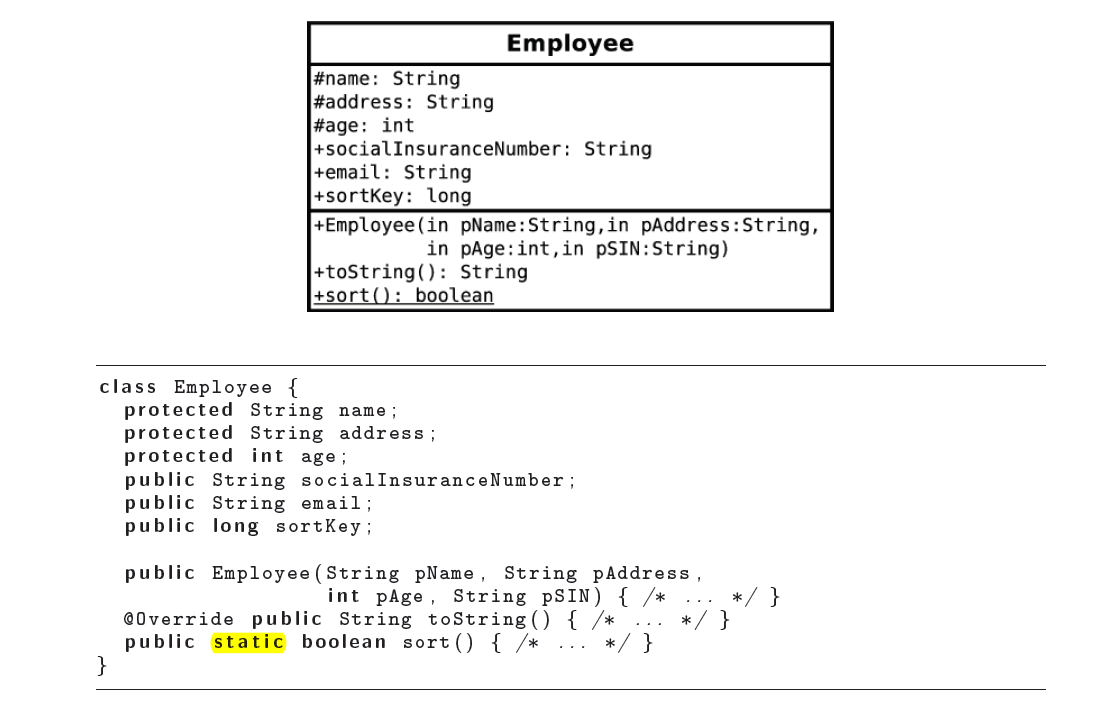
Describe the members of a class

• attributes / fields

• operations (constructors, destructors, methods, indexers and properties)

* A + (plus sign) = public visibility
* A - (minus sign) = private visibility
* A # (hash sign) = protected visibility
* A ~ indicates a destructor or package
* A .. indicates a range of values
* A : seperates a name from a type
* A , separates items in a set

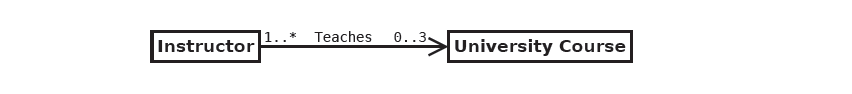
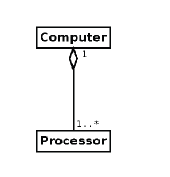
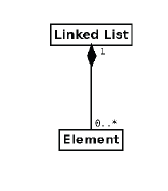
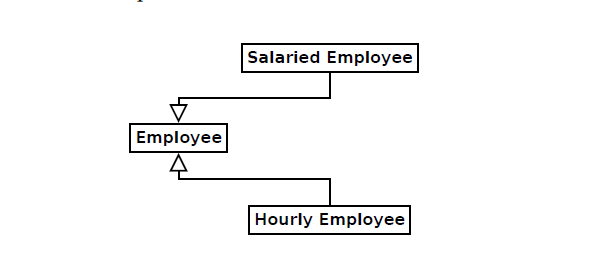
**Example:**



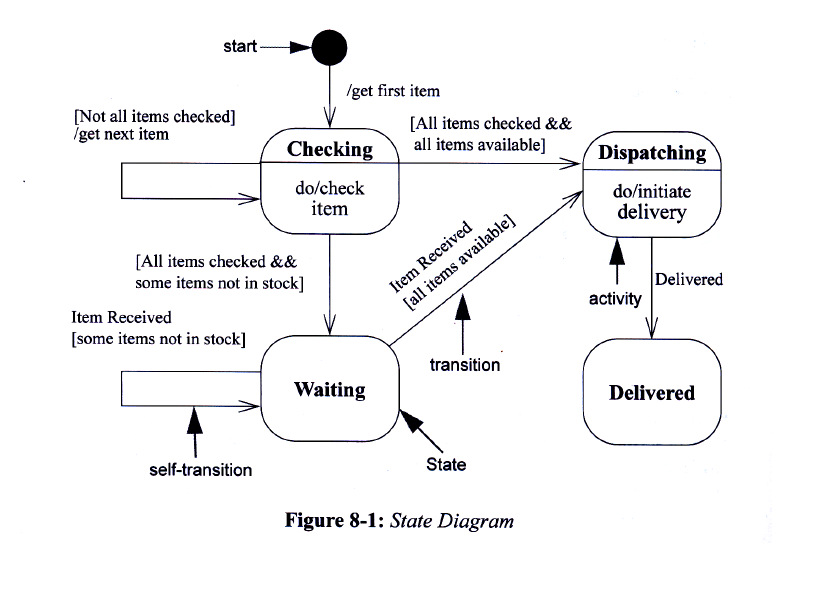
# Class Hierarchies in UML

UML can represent relationships between classes and instances

Between instances of class:

* **Association** : represent a relationship between instances of two classes
  + 
  + Instructors teach 0 -3 courses each term, each course has at least 1 - inf instructors
  + A method that the instructor might call on the course might be **giveLecture()**
* **Aggregation** : “part-whole” relationship between a collection or container instance and its contents
  + 
  + Computers contain 1 or more processors, but processors only belong to 1 Computer.
  + Aggregation means: Processors can exist independently of a Computer
* **Composition**: relationship between container and its contents
  + 
  + Composition means : Elements may not exist independently of a Linked List
  + Linked List contains 0 or more Elements, but Elements belong to only 1 Linked List;
* **Generalization**: relationship between base classes and their derived classes
  + 
  + Means: Salaried Employee and Hourly Employee extends Employee

# UML State Diagrams

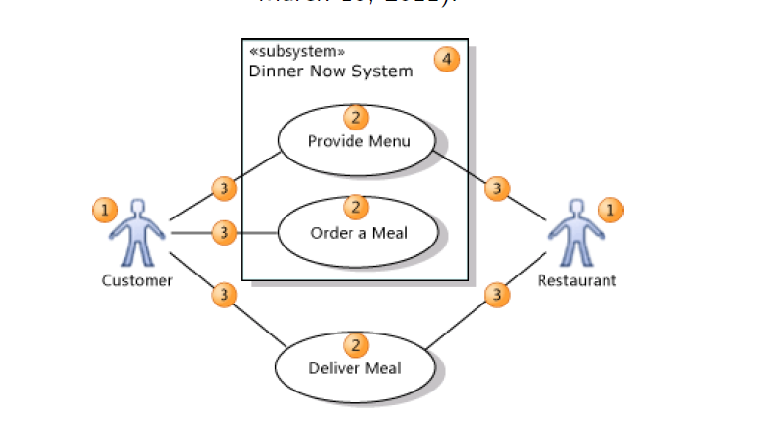


* States
* Transitions: edges between states
* Superstates: contain nested states and transitions
* Timing

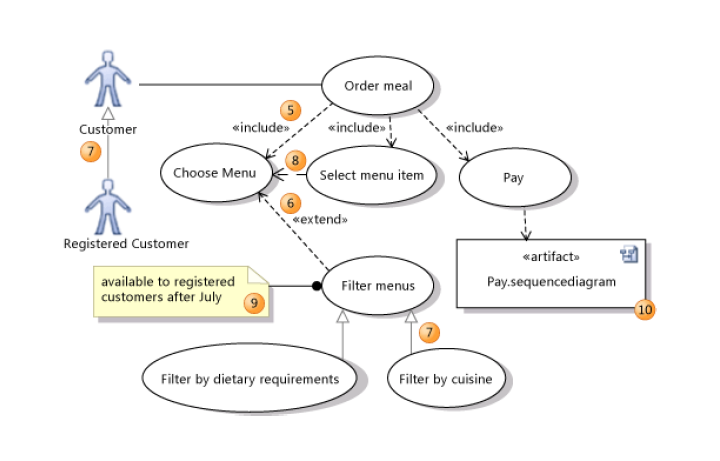
# UML Use Case Diagrams

describes a user’s interaction with a system.

**Examples:**



1: actors 2:use cases 3:communication 4:subsytems or components



5: inclusion stereotypes on dependencies

6: extension stereotypes on dependencies

7: inheritance relationships between use cases

8: plain dependencies

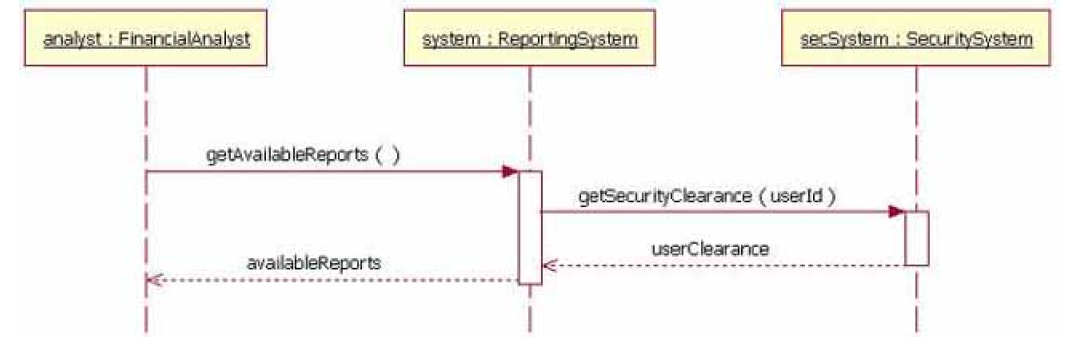
9: comments

10) references to other artifacts

**Useful for**

* Documenting essential features of a software system
* Communicating system behaviour to clients
* Generating appropriate test cases for scenarios

# UML Sequence Diagrams ( Express system behaviour as a sequence of events and activities )



* Instances of objects: boxes at the top of the diagram
* Lifelines (vertical dashed lines, extending downwards from instances of the objects): denote the passing of time
* Messages (horizontal lines): denote communication between objects
* Action boxes (boxes on top of lifelines): denote the occurrence of activities.
* Gates (filled circles on the boundary of the diagram): denote occurrence of external events.