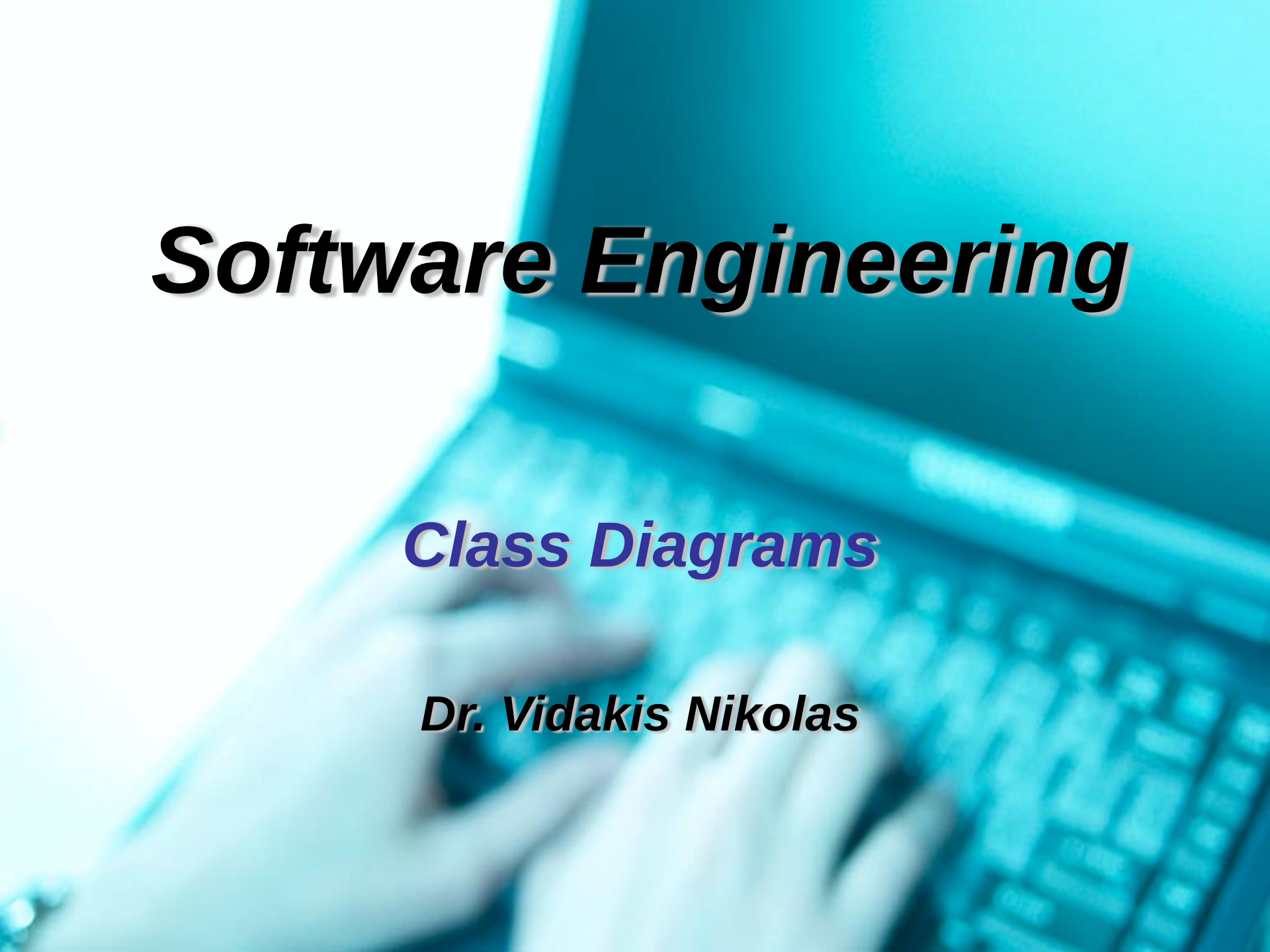


Software Engineering

Class Diagrams

Dr. Vidakis Nikolas

Διαγράμματα Κλάσεων (Class Diagrams)
Εισαγωγή

Class Diagrams

- Τα διαγράμματα κλάσεων χρησιμοποιούνται για να περιγράψουν τα αντικείμενα και τις σχέσεις τους σε ένα σύστημα.
- Τα διαγράμματα κλάσεων μοντελοποιούν τις δομές των κλάσεων και το περιεχόμενο τους χρησιμοποιώντας σχεδιαστικά στοιχεία όπως κλάσεις, πακέτα και αντικείμενα.
- Τα διαγράμματα κλάσεων περιγράφουν τρεις διαφορετικές διαστάσεις κατά τον σχεδιασμό ενός συστήματος: την εννοιολογική διάσταση (conceptual), την διάσταση τεχνικών προδιαγραφών(specification) και την διάσταση της υλοποίησης (implementation).
- Τα διαγράμματα κλάσεων εμπεριέχουν: τις κλάσεις του συστήματος, τις σχέσεις μεταξύ των κλάσεων και τις ιδιότητες & λειτουργίες των κλάσεων.

Διαγράμματα Κλάσεων (Class Diagrams)
When to Use

- Τα διαγράμματα κλάσεων χρησιμοποιούνται για να περιγράψουμε τις κλάσεις ενός συστήματος και τις σχέσεις μεταξύ τους.
- Τα διαγράμματα κλάσεων αποτελούν την σπονδυλική στήλη της UML.
- Μην προσπαθείτε να χρησιμοποιήσετε όλους τους συμβολισμούς που έχετε στην διάθεση σας. Ξεκινήστε με τους απλούς συμβολισμούς (κλάσεις, συσχετίσεις, χαρακτηριστικά, γενικεύσεις και περιορισμούς)
- Μη σχεδιάζετε μοντέλα για τα πάντα, επικεντρωθείτε στις κρίσιμες περιοχές.
- Ο μεγαλύτερος κίνδυνος με τα διαγράμματα κλάσεων είναι να εστιάσετε την προσοχή σας αποκλειστικά στην δομή και να αγνοήσετε τη συμπεριφορά.
- Τα διαγράμματα κλάσεων συνήθως χρησιμοποιούνται για:
  - Να ερευνήσουν έννοιες/θέματα τομέα με την μορφή μοντέλων τομέα
  - Να αναλύσουν απαιτήσεις με την μορφή εννοιολογικών μοντέλων
  - να παρουσιάσουν αναλυτικό σχεδιασμό αντικειμενοστρεφούς λογισμικού.

Διαγράμματα Κλάσεων (Class Diagrams)

How to Use

- Πριν ξεκινήσετε να σχεδιάζετε διαγράμματα κλάσεων λάβετε υπ' όψιν τις τρεις διαφορετικές διαστάσεις του συστήματος που θα περιγράψει το διάγραμμα

  την εννοιολογική διάσταση (conceptual),

  την διάσταση τεχνικών προδιαγραφών(specification)

  και την διάσταση της υλοποίησης (implementation)

  - Προσπαθήστε να μην εστιαστείτε σε μια διάσταση.
- Κατά την σχεδίαση κλάσεων εξετάστε:

  ποιες ιδιότητες και λειτουργίες έχει η κάθε κλάση.

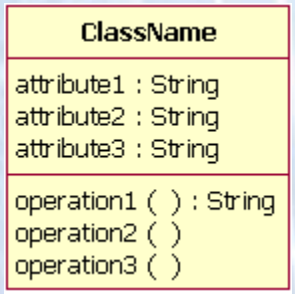
  πως αλληλεπιδρούν οι instances των κλάσεων μεταξύ τους

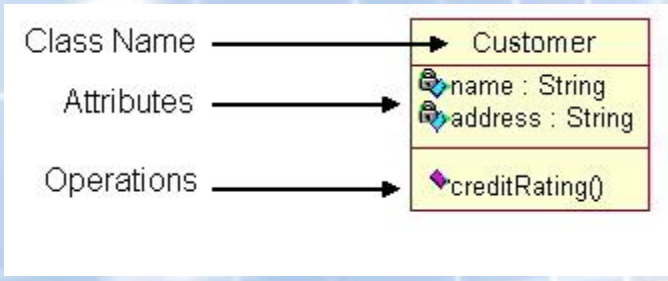
Οι παραπάνω είναι οι βασικές τεχνικές για την σχεδίαση διαγραμμάτων κλάσεων.

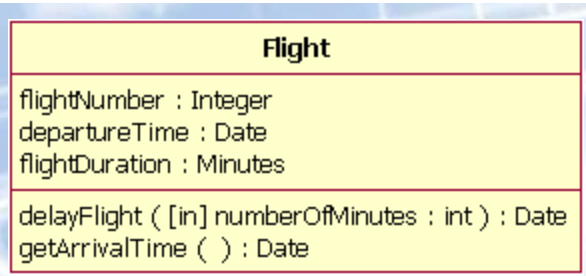
Class Diagrams

Class

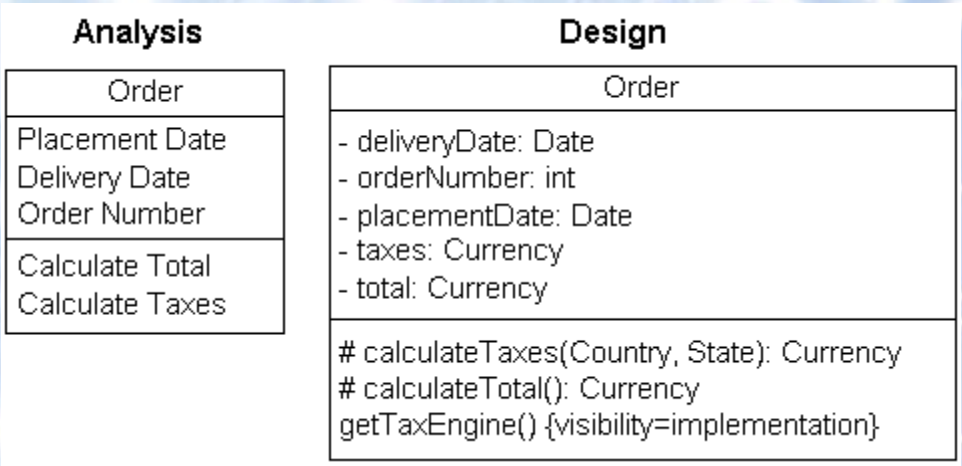
- a name,
- attributes,
- and operations.







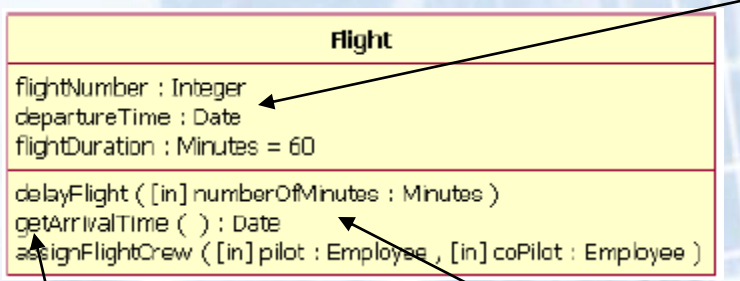
Analysis and design versions of a class



Class Diagrams
Class (cont.)

name : attribute type = default value

Attribute Name	Attribute Type
flightNumber	Integer
departureTime	Date
flightDuration	Minutes



Operation Name	Parameters		Return Value Type
delayFlight	Name	Type	N/A
	numberOfMinutes	Minutes	
getArrivalTime	N/A		Date

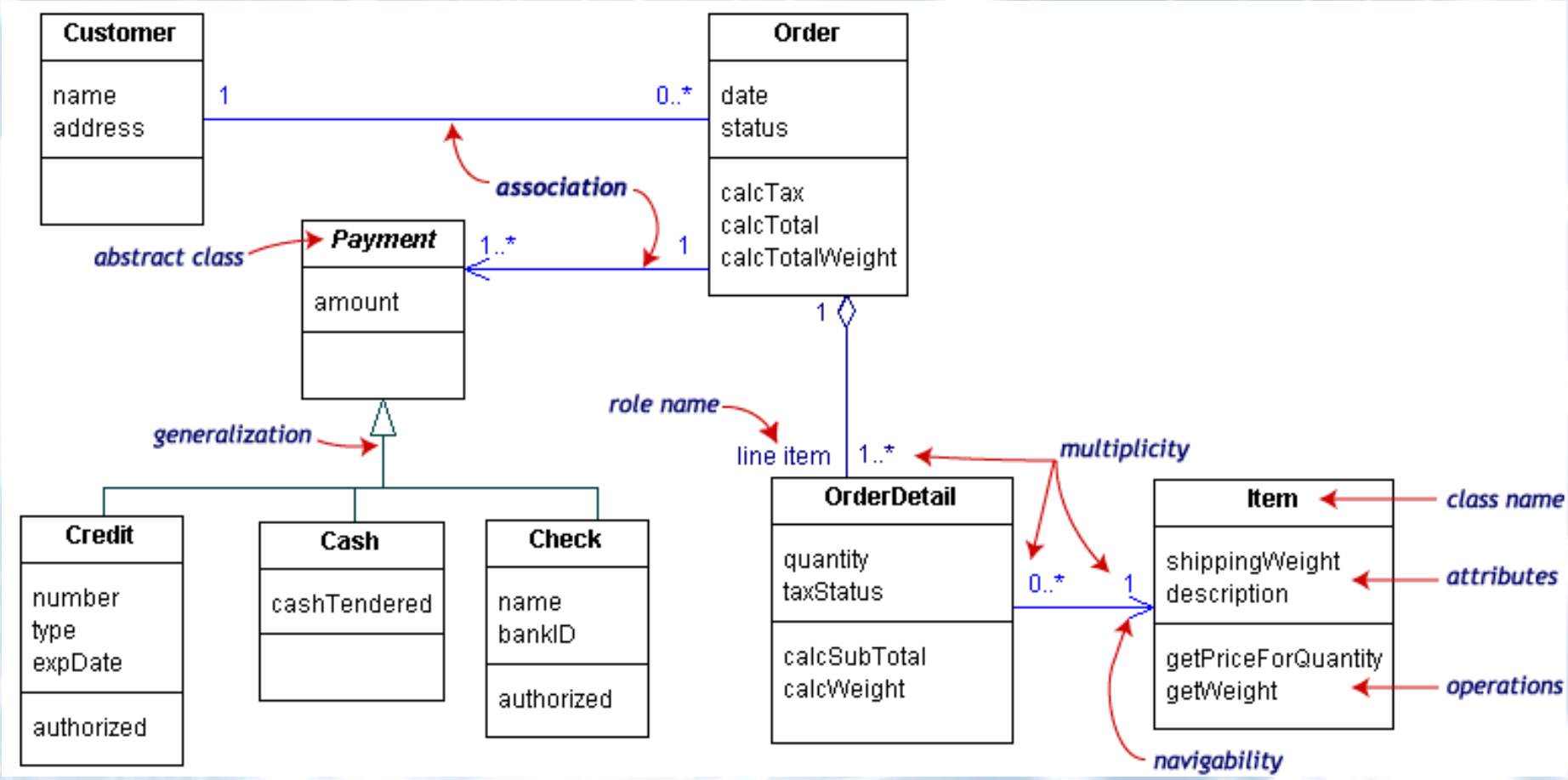
Operations

delayFlight: 1 input parameter, no return value

getArrivalTime: 0 parameter, return value

assignFlightCrew: 2 input parameters, no return value

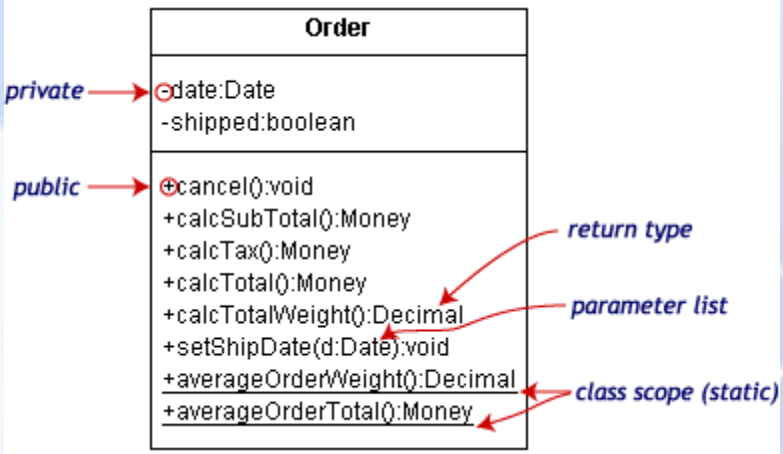
Class Diagrams
Class example (cont.)



Class Diagrams

Class (cont.)

Class information: visibility and scope



Symbol	Access
+	Public
-	Private
#	Protected
~	Package

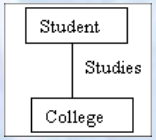
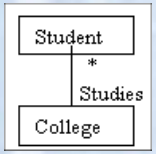
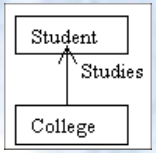
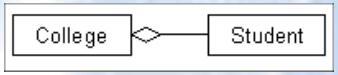
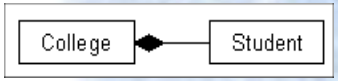
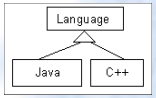
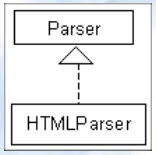
- Static members are underlined. Instance members are not.
- The parameter list shows each parameter type preceded by a colon.
- Access specifiers appear in front of each member.

Class Diagrams
Relationships

- Inheritance - Generalization
- Bi-directional association
- Uni-directional association
- Association class
- Basic aggregation
- Composition Aggregation
- Reflexive association
- Dependency

Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

A/A	Relation	Symbol	Description
1	Association		When two classes are connected to each other in any way, an association relation is established. For example: A "student studies in a college" association can be shown as:
1 a.	Multiplicity		An example of this kind of association is many students belonging to the same college. Hence, the relation shows a star sign near the student class (one to many, many to many, and so forth kind of relations).
1 b.	Directed Association		Association between classes is bi-directional by default. You can define the flow of the association by using a directed association. The arrowhead identifies the container-contained relationship.
1 c.	Reflexive Association	No separate symbol. However, the relation will point back at the same class.	An example of this kind of relation is when a class has a variety of responsibilities. For example, an employee of a college can be a professor, a housekeeper, or an administrative assistant.
2	Aggregation		When a class is formed as a collection of other classes, it is called an aggregation relationship between these classes. It is also called a "**has a**" relationship.
2 a.	Composition		Composition is a variation of the aggregation relationship. Composition connotes that a strong life cycle is associated between the classes.
3	Inheritance/Generalization		Also called an "**is a**" relationship, because the child class **is a** type of the parent class. Generalization is the basic type of relationship used to define reusable elements in the class diagram. Literally, the child classes "inherit" the common functionality defined in the parent class.
4	Realization		In a realization relationship, one entity (normally an interface) defines a set of functionalities as a contract and the other entity (normally a class) "realizes" the contract by implementing the functionality defined in the contract.

Διαγράμματα Κλάσεων (Class Diagrams)

Multiplicity

Potential Multiplicity Values	
Indicator	**Meaning**
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
N	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)

Διαγράμματα Κλάσεων (Class Diagrams)

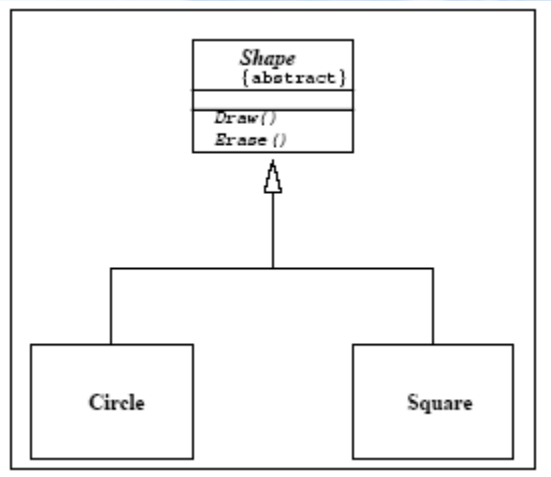
Relationships

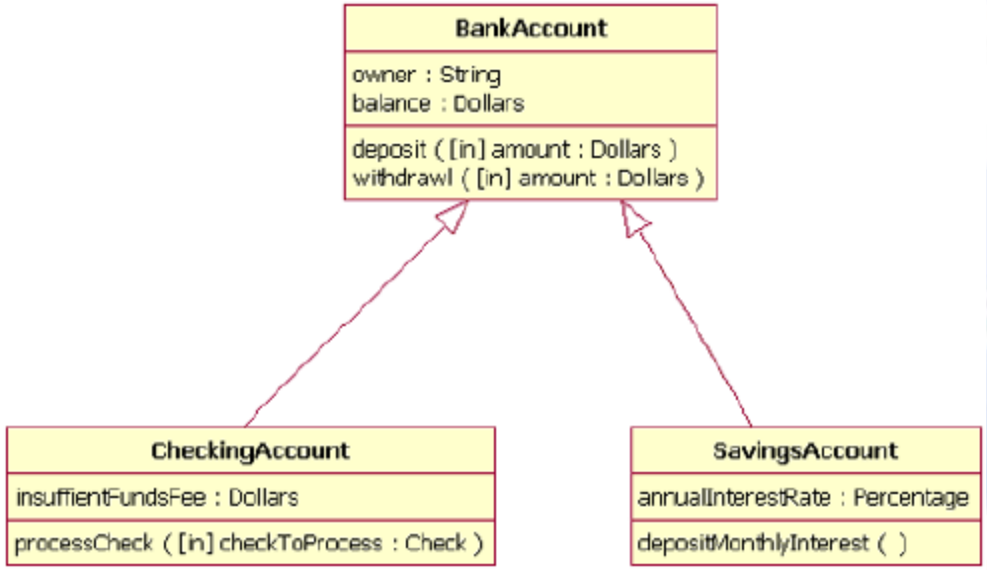
Inheritance / Generalization

Generalization: Is an **inheritance link** that defines a class as a «super class» of another class.

The notation used is the arrow pointing towards the superclass.

*At the example **Shape** is a superclass of **Circle** and **Square**.*



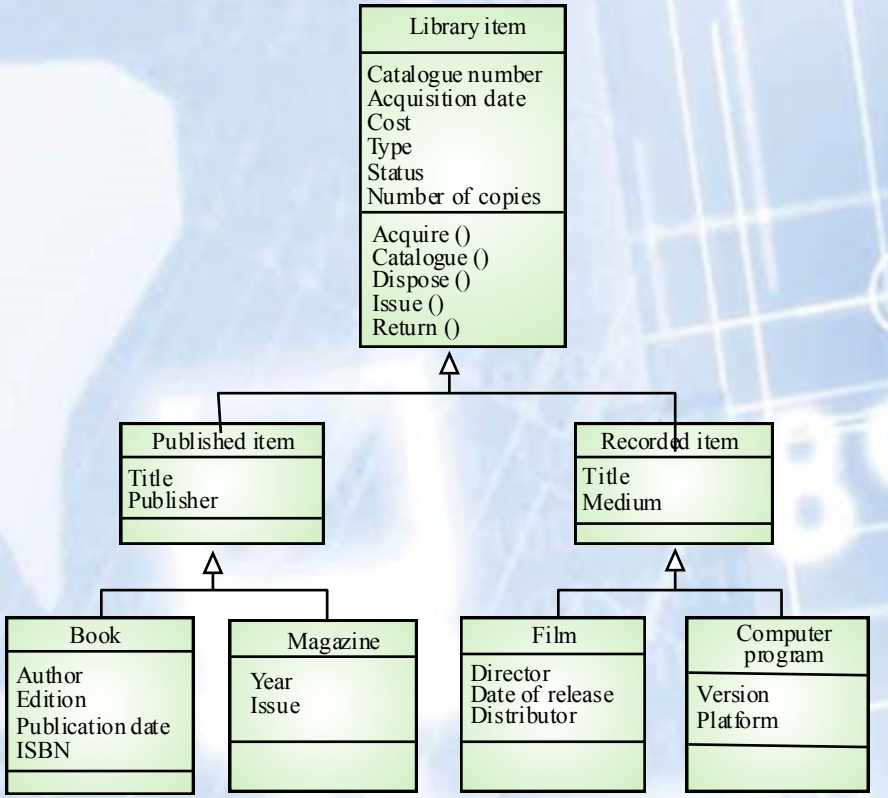


Διαγράμματα Κλάσεων (Class Diagrams)

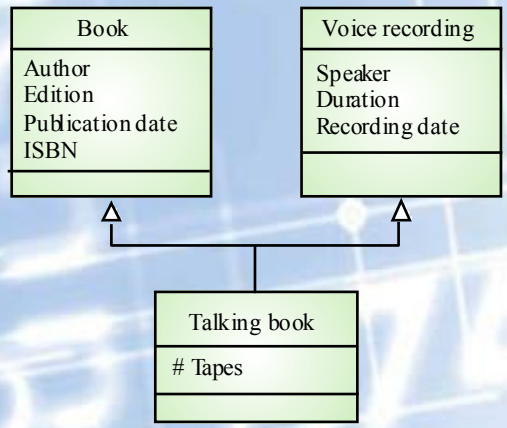
Relationships

Inheritance / Generalization

Single Inheritance



Multiple Inheritance

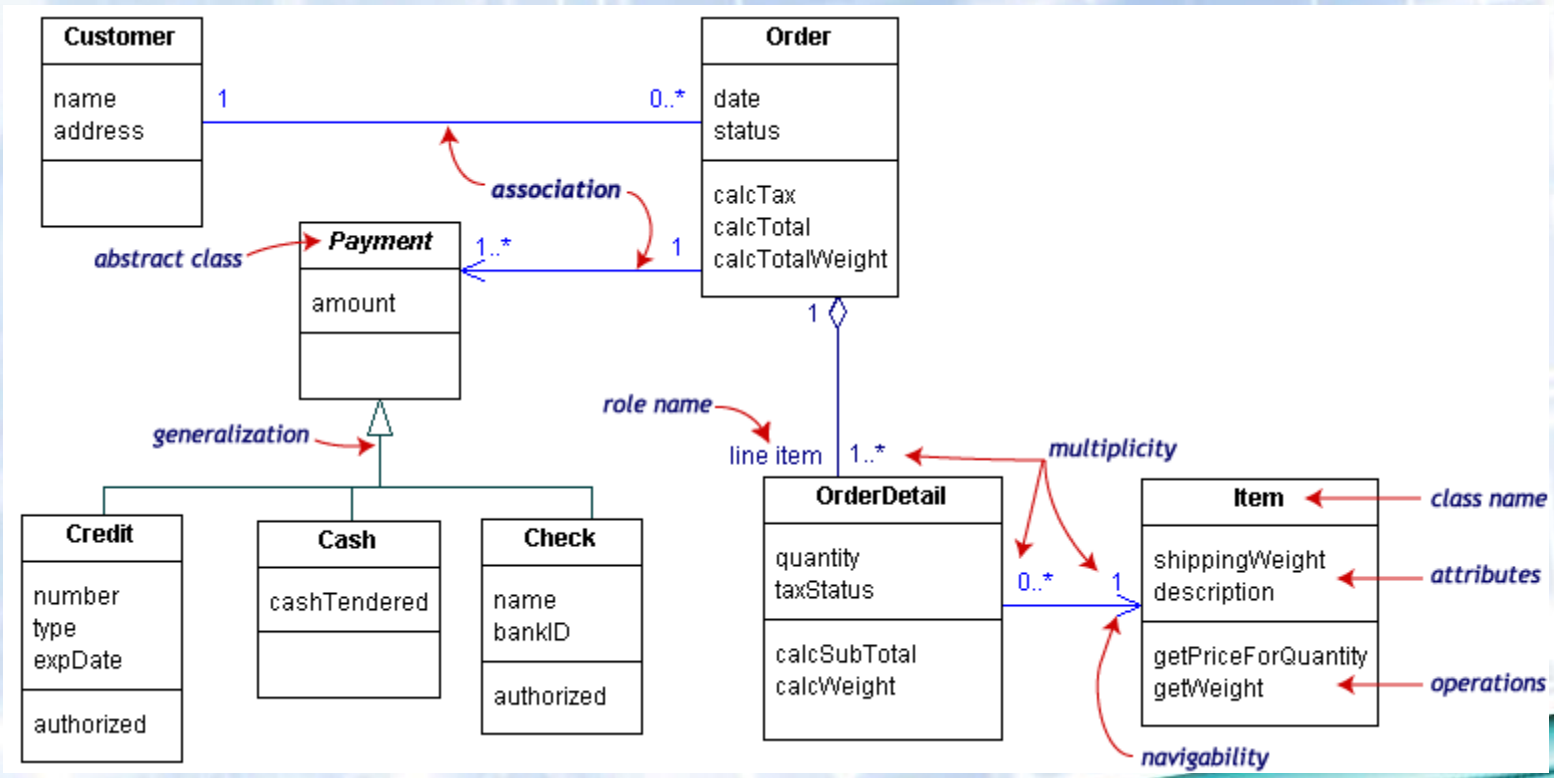


Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

Bi & Uni -Directional Association

Association: is a relation between the instances of two classes. When the instance of a class needs information about another class in order to complete its function then there is an association between the instances of the two classes.

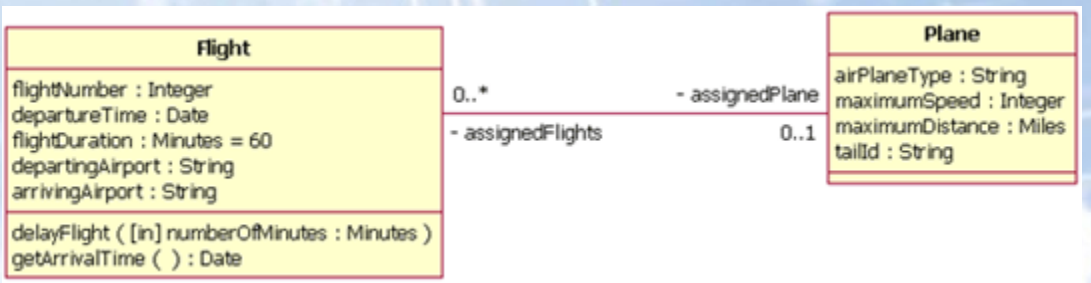


Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

Bi & Uni -Directional Association

Bi-Dierctional

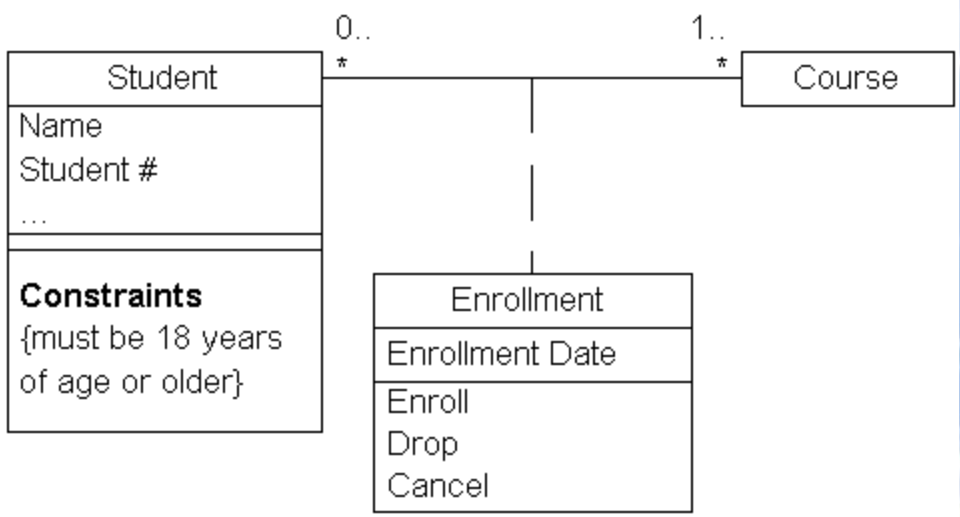


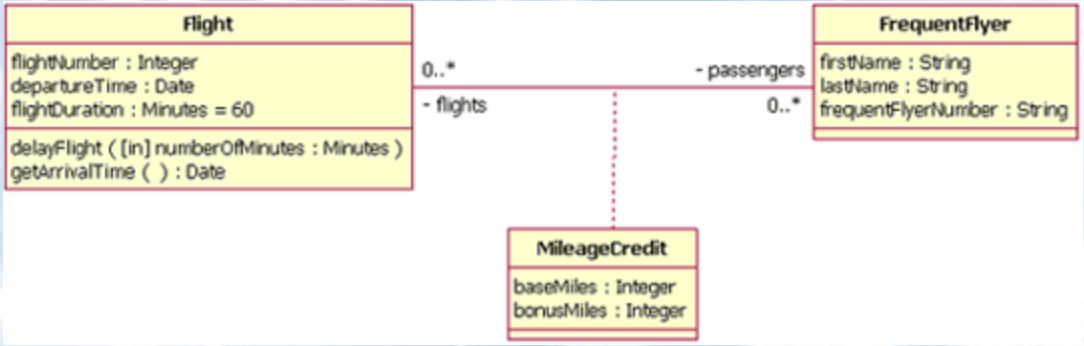
Uni-Dierctional

Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

Association Class





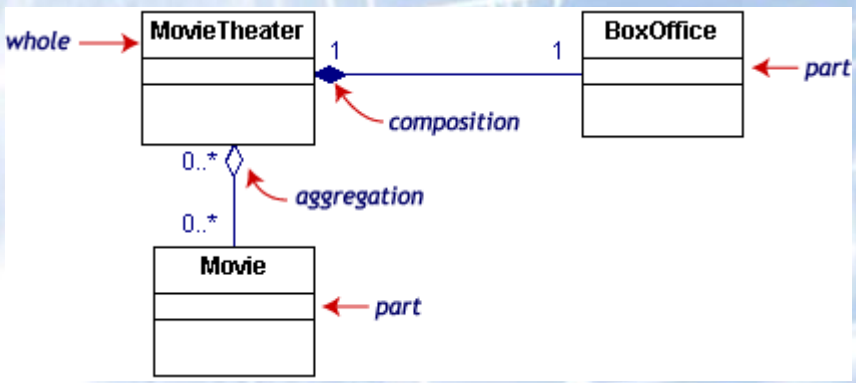
Διαγράμματα Κλάσεων (Class Diagrams)

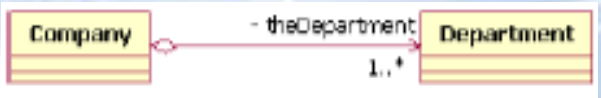
Relationships

Basic Aggregation

Aggregation: a class belongs to a whole or a collection

At the example, ***Company*** *has a collection of* ***Departments****,*
Movie Theater *has a collection of* ***Movies***





Διαγράμματα Κλάσεων (Class Diagrams)

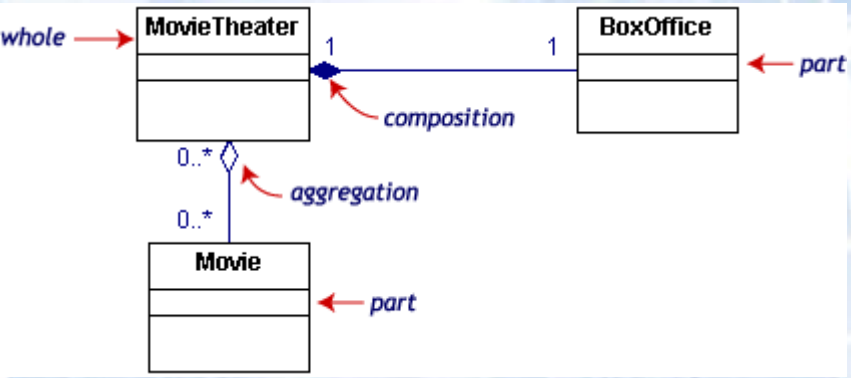
Relationships

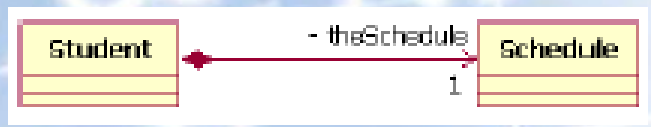
Composition Aggregation

Composition: is a (strong association) in which the (part) can belong only to one (whole). The part can not exist without the whole.

*At the example **BoxOffice** belongs to only one **MovieTheater**. If **MovieTheater** is deleted then the **BoxOffice is deleted as well.***

*On the other hand **Movies** are not so close connected to the **MovieTheater**.*



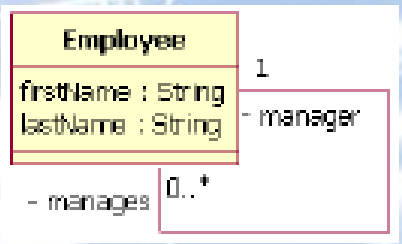


Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

Reflexive Assosiations

Reflexive: When an instance of a class associates with another instance of the same class.



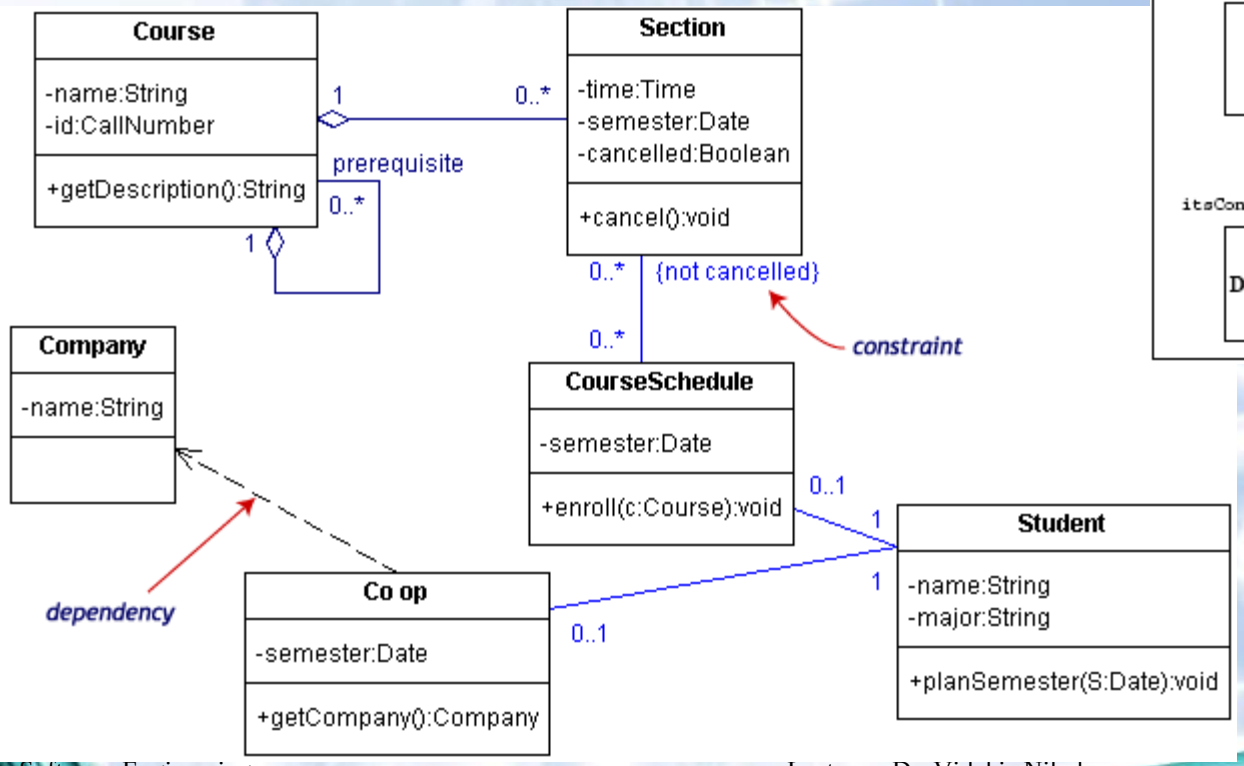
Διαγράμματα Κλάσεων (Class Diagrams)

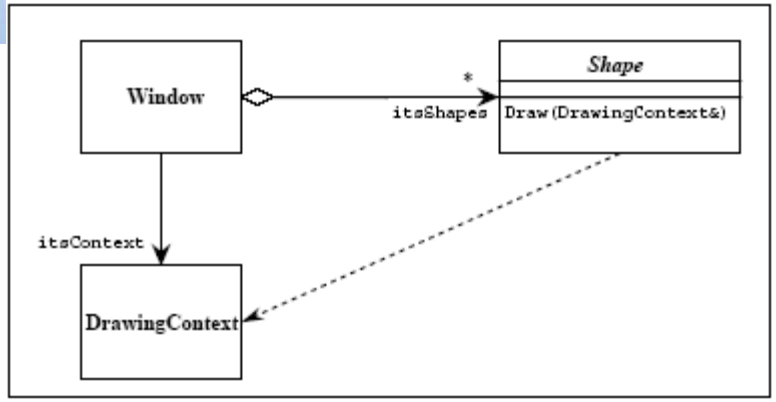
Relationships

Dependency

Dependency: when changes to one class denote changes to another class.

At the example, ***Co_op*** *depends on* ***Company****. If* ***Company*** *changes, then* ***Co_op*** *might change.*





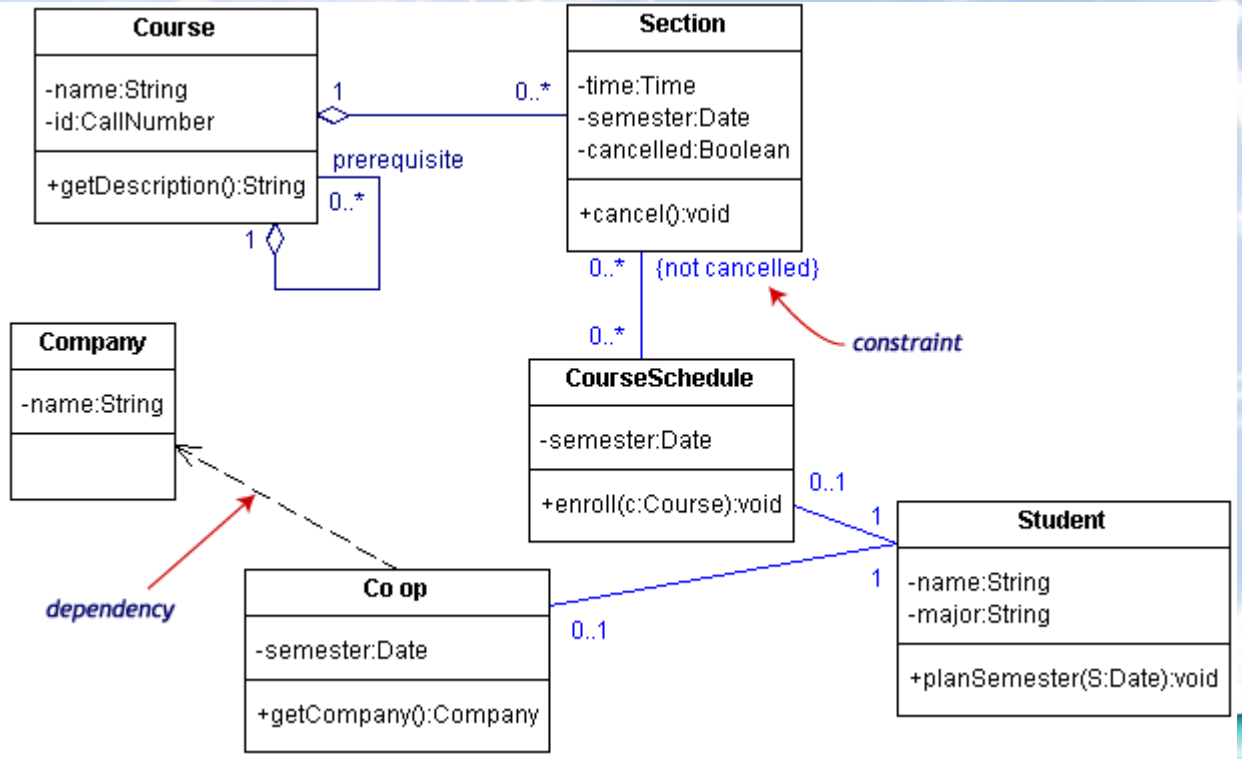
Διαγράμματα Κλάσεων (Class Diagrams)

Relationships

Constraint

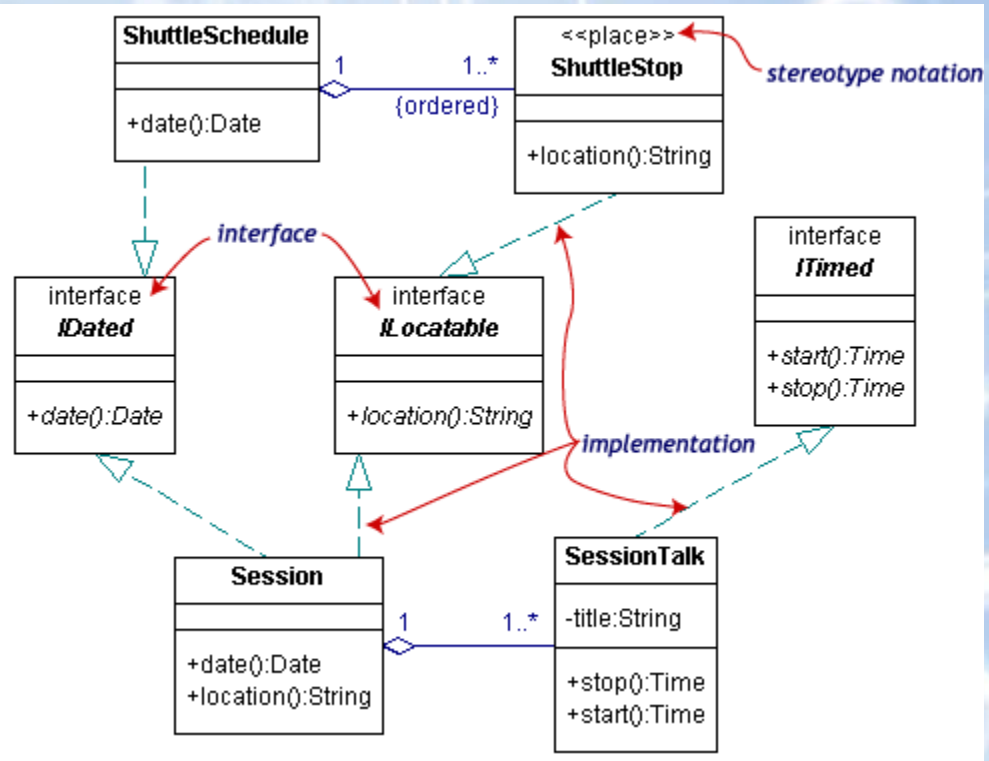
Constraint: (condition) that has to be satisfied.

*At the example (constraint) denotes that a **Section** can be part of **CourseSchedule** only if it is not canceled.*



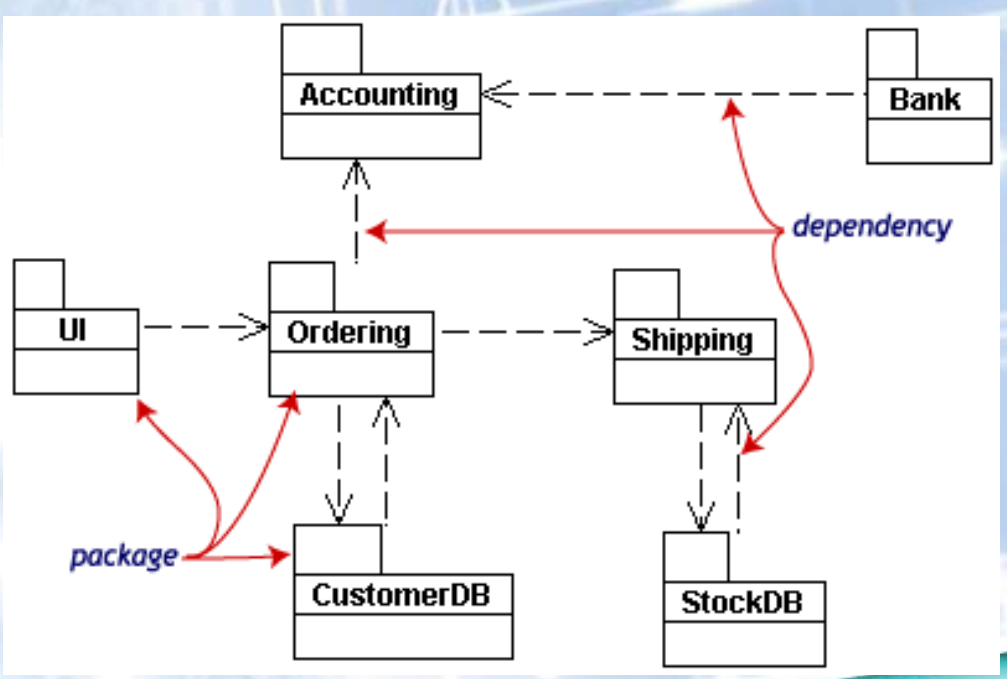
Διαγράμματα Κλάσεων (Class Diagrams)
Interfaces & Stereotypes

Interface: is a set of operation signatures. In C++, interfaces are developed with abstract classes with virtual parts. In Java, there is a straight development.



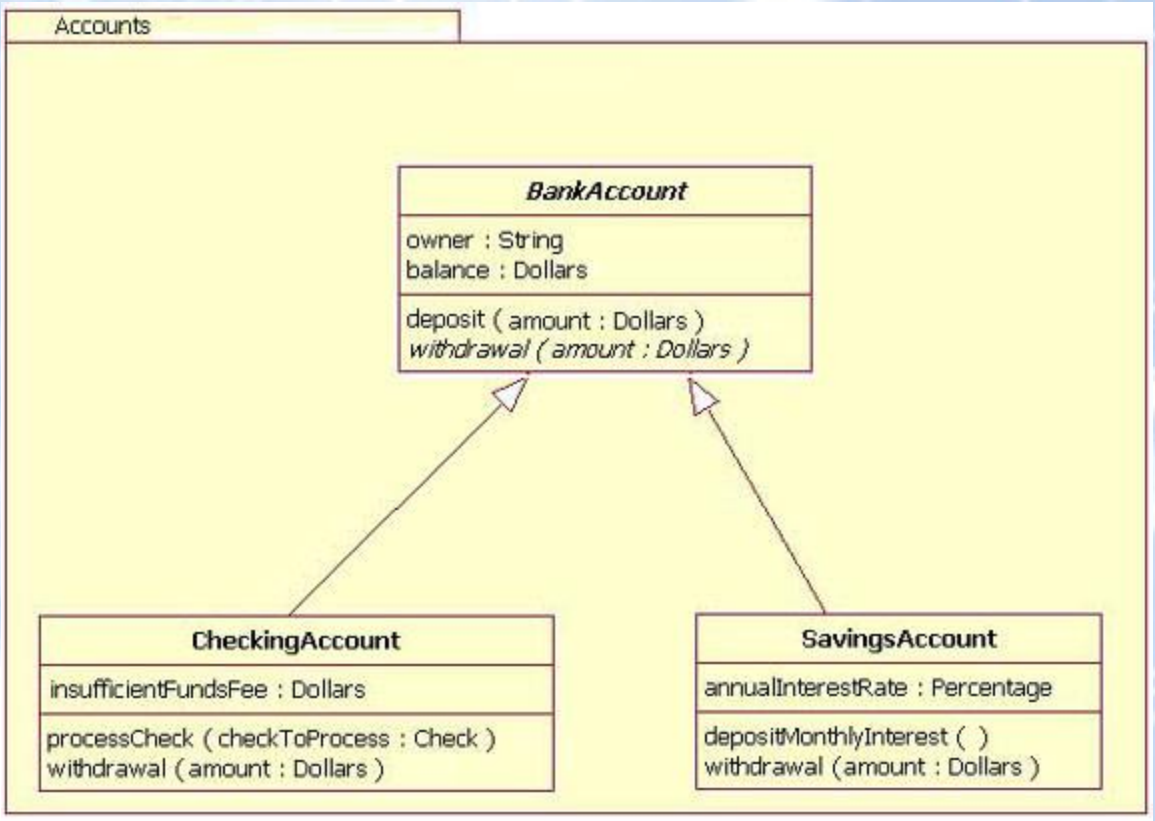
Διαγράμματα Κλάσεων (Class Diagrams)
Packages

- Packages are object diagrams.
- They are used to simplify complex class diagrams by packaging them into **packages**.
- The object in a package are logically connected
- The doted lines denote **dependencies**.



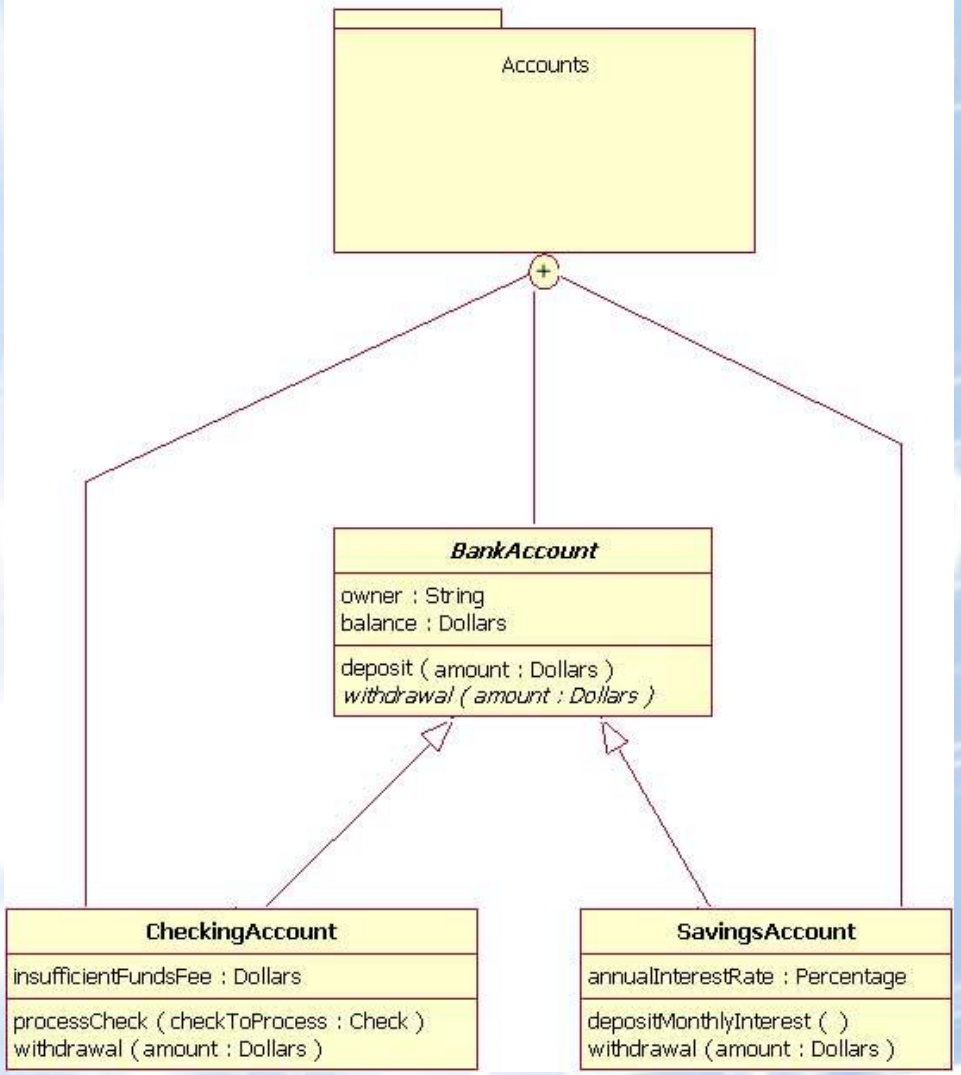
Διαγράμματα Κλάσεων (Class Diagrams)

Packages



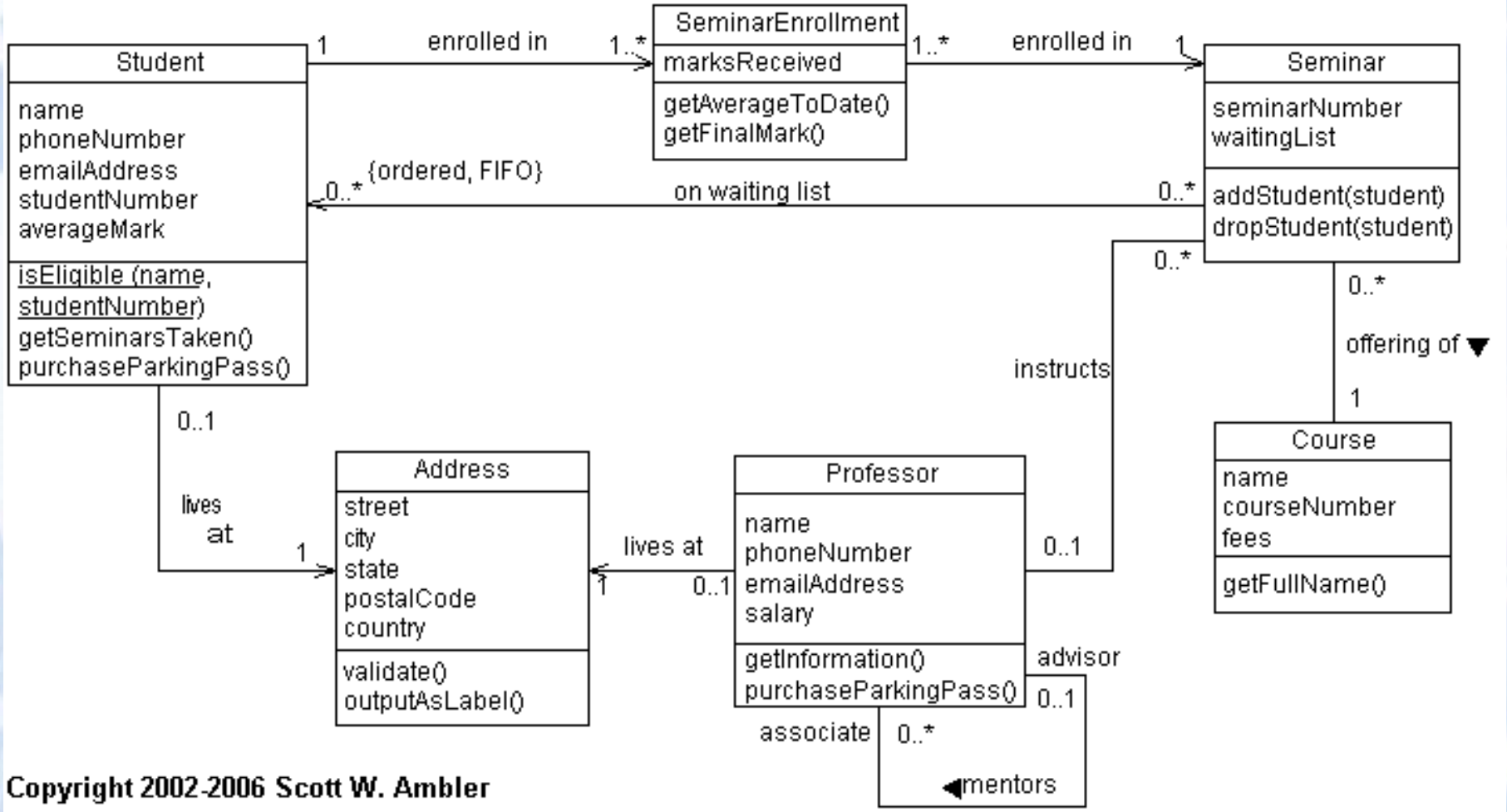
Διαγράμματα Κλάσεων (Class Diagrams)

Packages



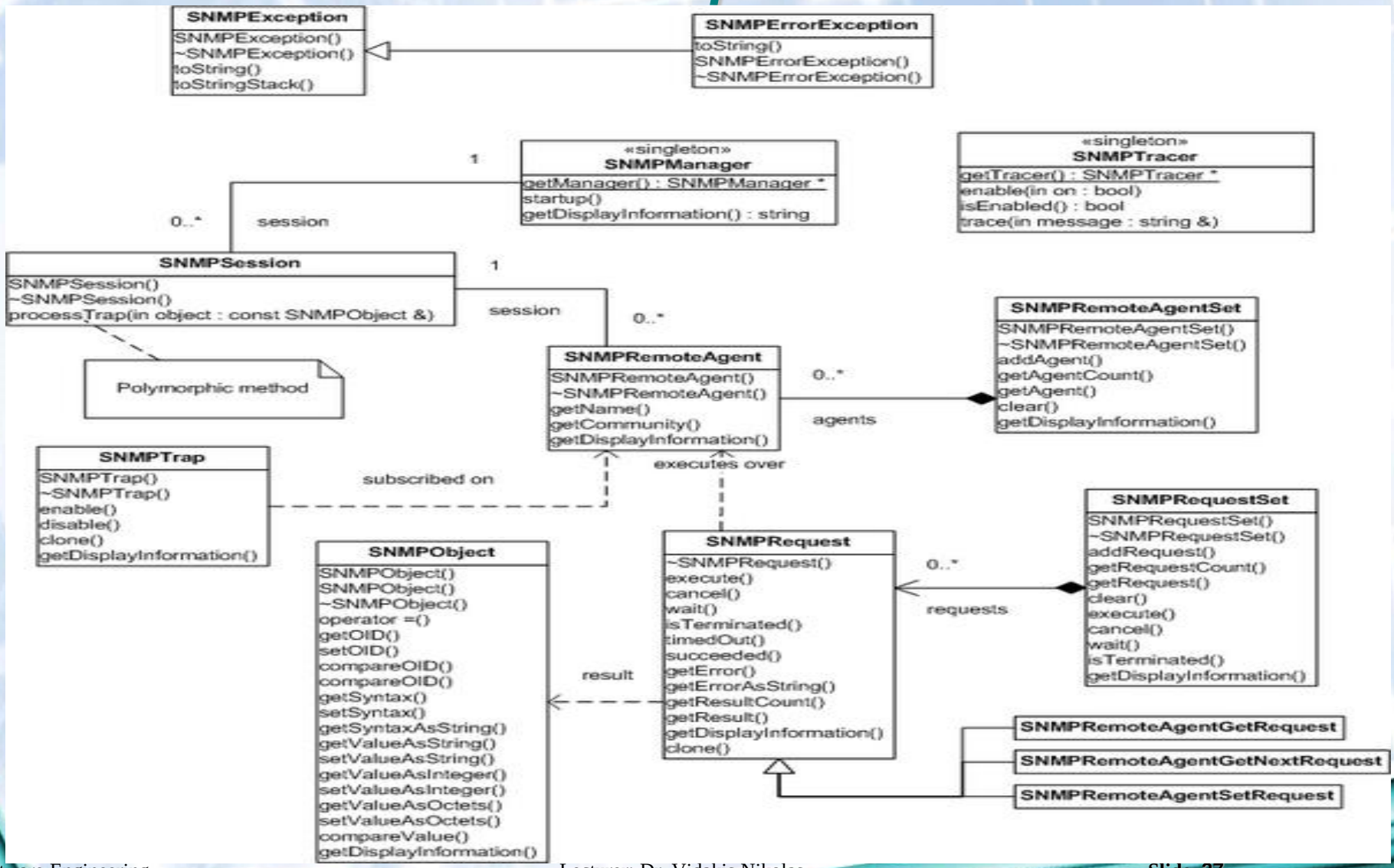
Διαγράμματα Κλάσεων (Class Diagrams)

Examples

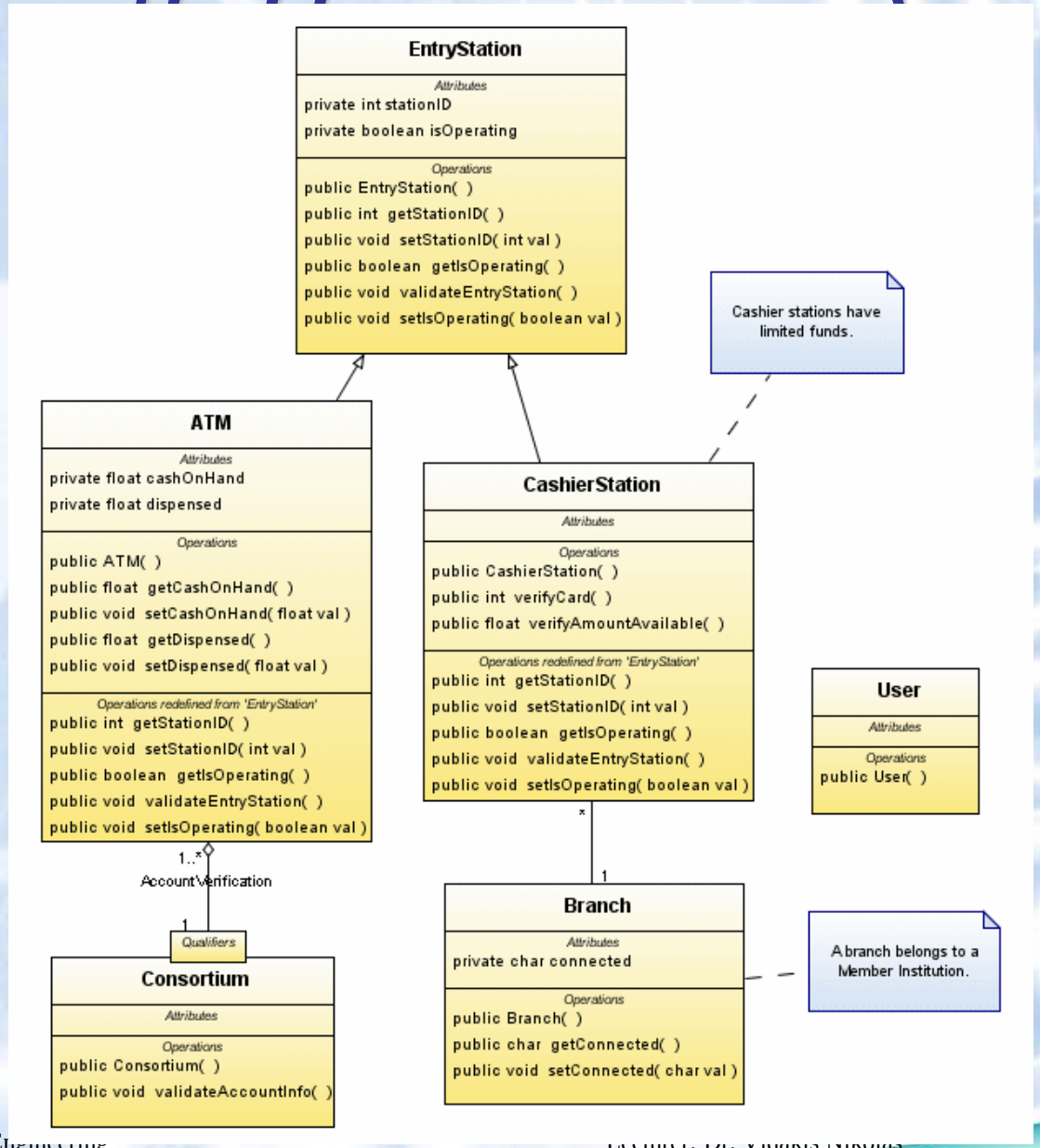


Διαγράμματα Κλάσεων (Class Diagrams)

Examples



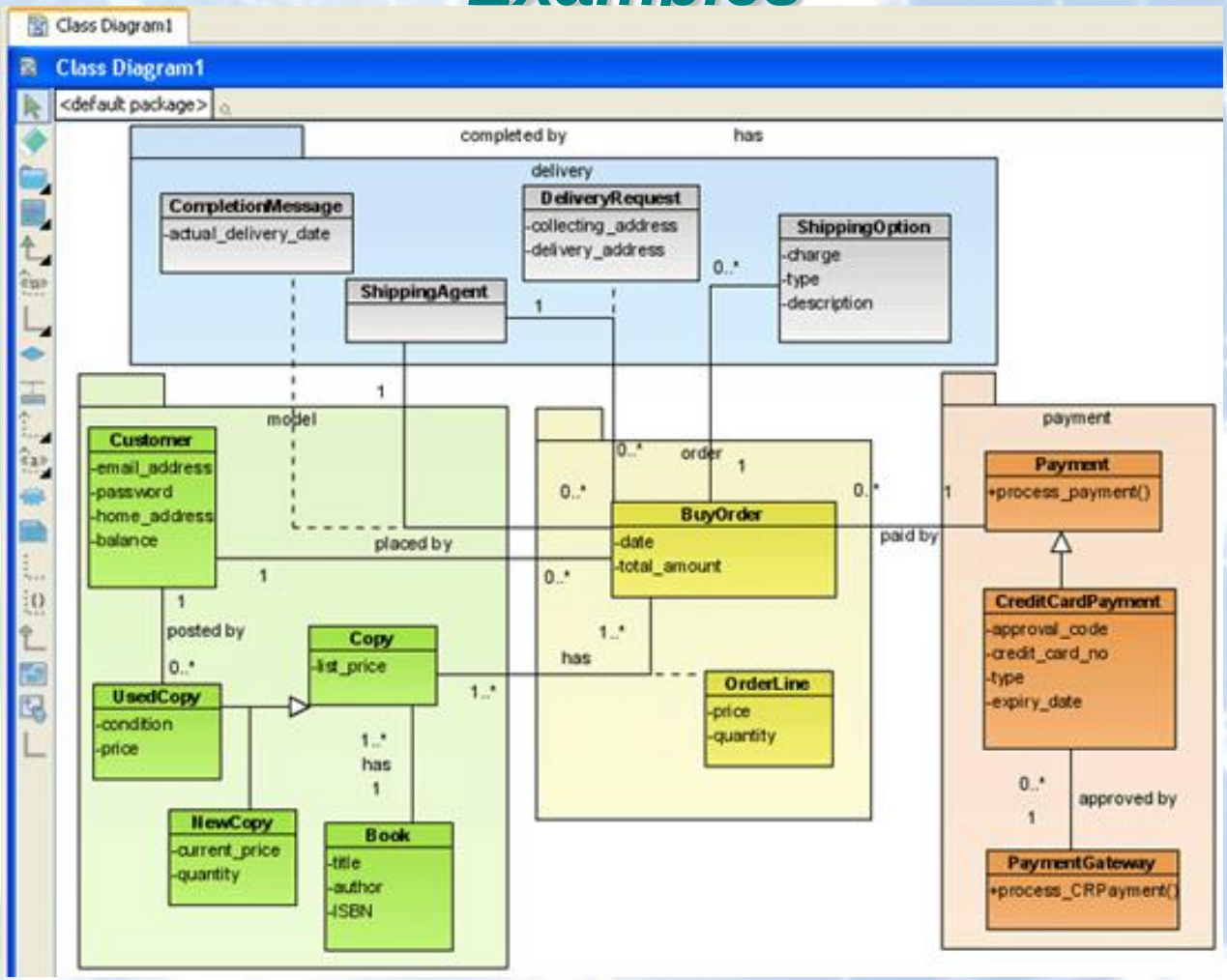
Διαγράμματα Κλάσεων (Class Diagrams)



www.netbeans.org/kb/60/uml/class-diagram.html

Διαγράμματα Κλάσεων (Class Diagrams)

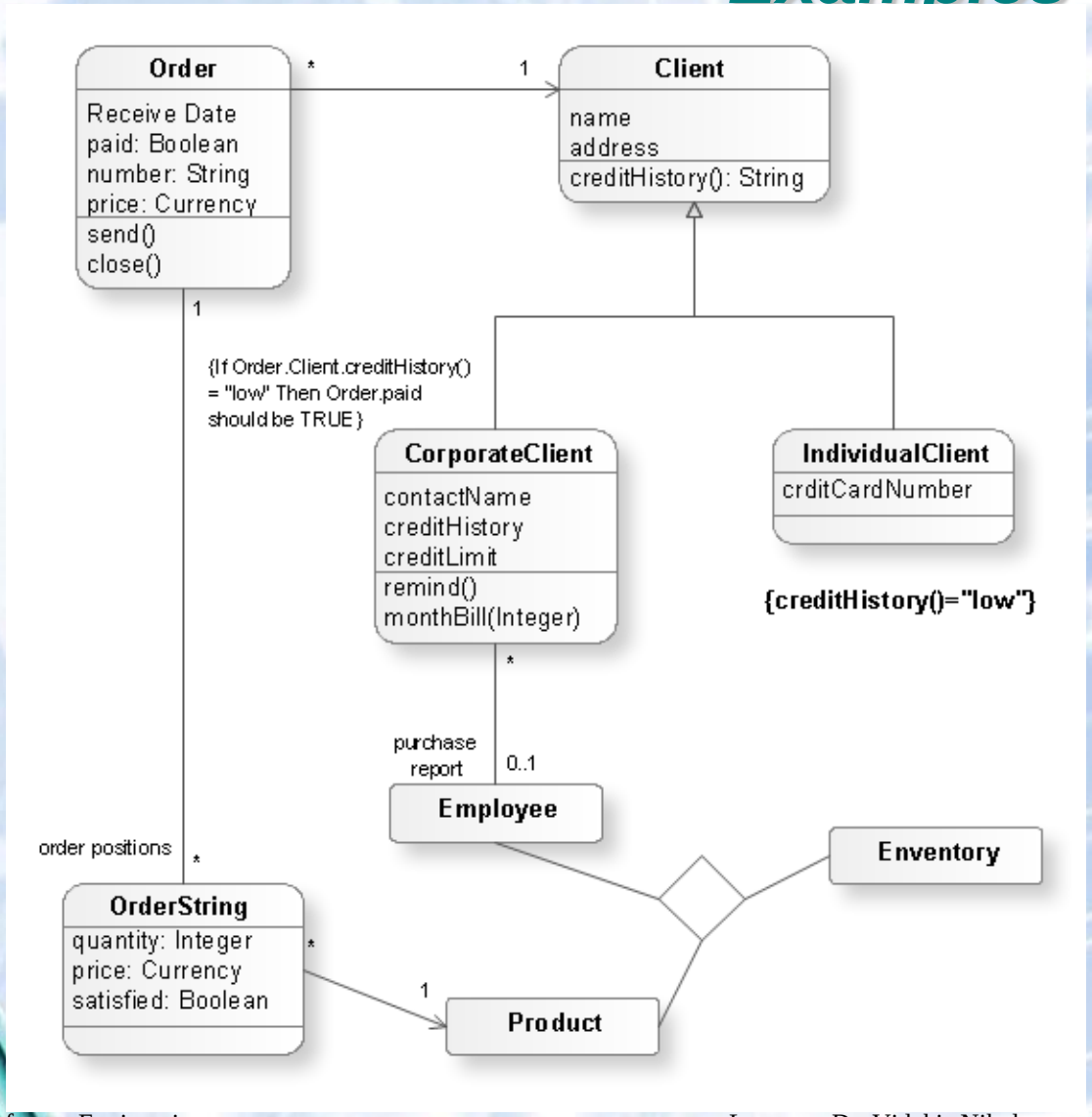
Examples



www.visual-paradigm.com/.../diagrams/Class.html

Διαγράμματα Κλάσεων (Class Diagrams)

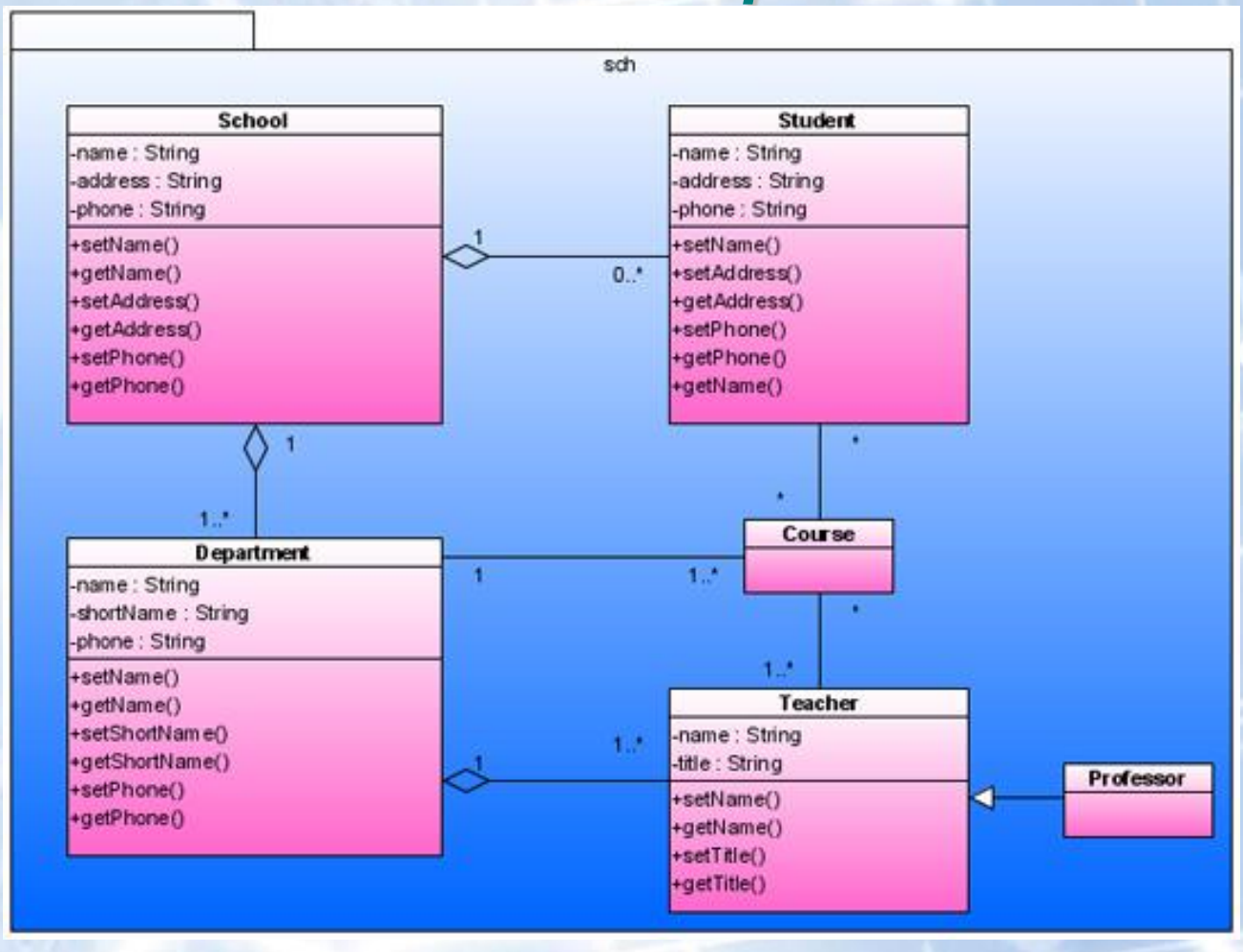
Examples



www.conceptdraw.com/en/products/cd5/ap_uml.php

Διαγράμματα Κλάσεων (Class Diagrams)

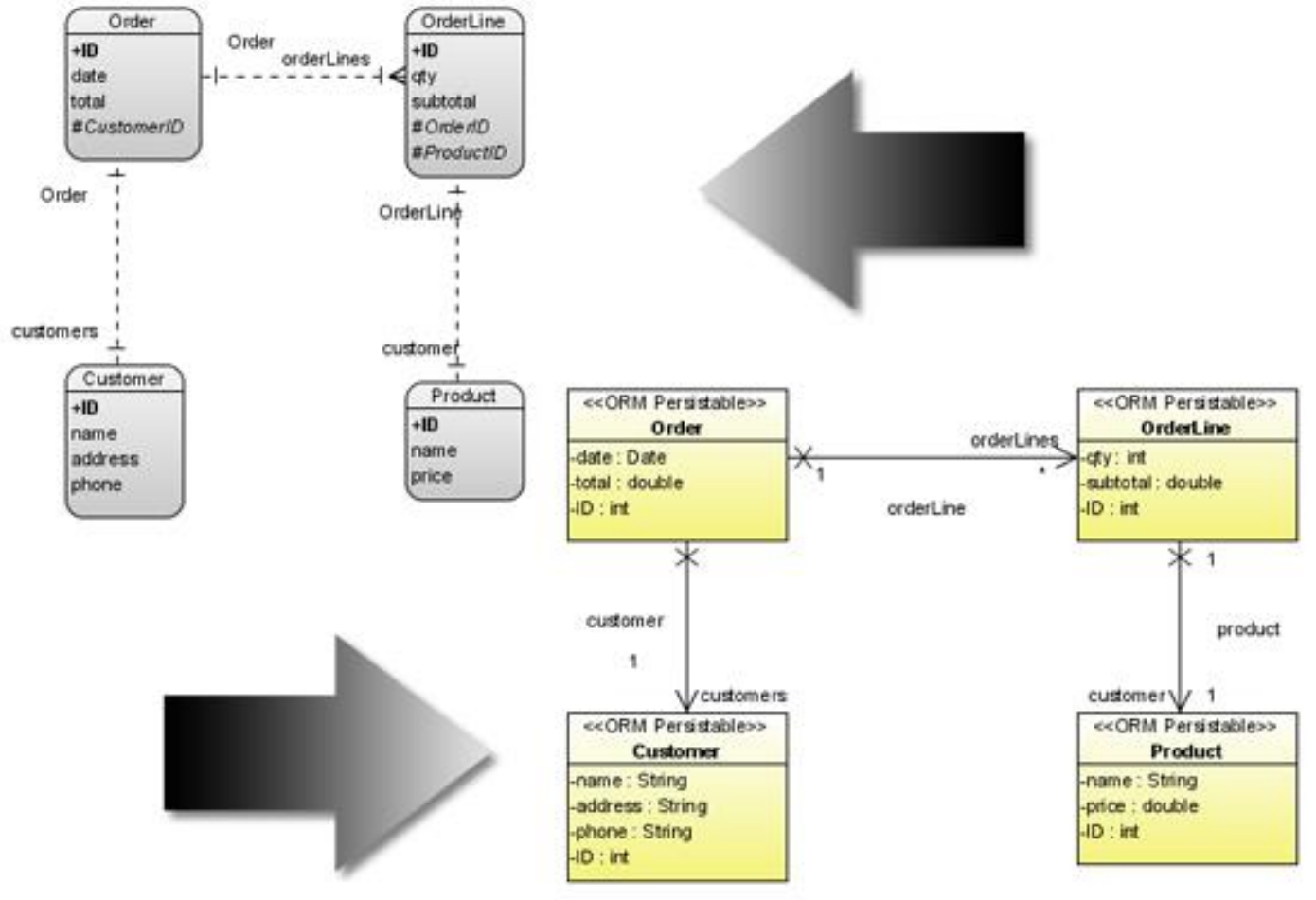
Examples



www.visual-paradigm.com/.../CodeReverse.html

Διαγράμματα Κλάσεων (Class Diagrams)

Examples



www.visual-paradigm.com/.../index.html

Βιβλιογραφία - Further Reading

1. Robert Cecil Martin, The principles, Petterns and Practices of Agile Software Development, Prentice-Hall, 2003
2. Software Engineering: (Update) (8th Edition) (International Computer Science Series); Ian Sommerville
3. UML Distilled: A Brief Guide to the Standard Object Modeling Language
   Martin Fowler, Kendall Scott
4. IBM Rational http://www-306.ibm.com/software/rational/uml/
5. UML basics: The class diagram, An introduction to structure diagrams in UML 2, Donald Bell (bellds@us.ibm.com), IT Specialist, IBM, http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html#N102EE
6. http://www.therationaledge.com/content/nov_03/t_modelinguml_db.jsp, Copyright Rational Software 2003
7. Practical UML --- A Hands-On Introduction for Developers http://www.togethersoft.com/services/practical_guides/umlonlinecourse/
8. Software Engineering Principles and Practice. Second Edition; Hans van Vliet.
9. http://www-inst.eecs.berkeley.edu/~cs169/
10. UML an overview By: DiGitAll
11. The UML Class Diagram: Part 1 By Mandar Chitnis, Pravin Tiwari, & Lakshmi Ananthamurthy
    http://www.developer.com/design/article.php/10925_2206791_1
12. Practical UML: A Hands-On Introduction for Developers By: Randy Miller, http://dn.codegear.com/article/31863#classdiagrams