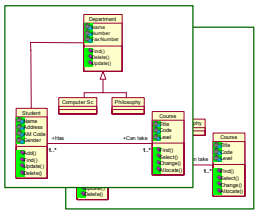
HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Μοντελοποίηση Δομής (II)
(Structural Modeling)



Γιάννης Τζίτζικας

Διάλεξη : 10
Ημερομηνία :
Θέμα :

Διάρθρωση

- Enumerations
- Class Scope for Operations and Attributes
- Derived Associations and Attributes
- Frozen
- Aggregation and Composition
- Collections for Multivalued Association Ends
- Association Class
- Qualified Associations
- Multiple and Dynamic Classification
- Parameterized Class
- Interfaces and Abstract Classes

Enumerations

Χρησιμοποιούνται για να δείξουν ένα προκαθορισμένο σύνολο τιμών οι οποίες δεν έχουν άλλες ιδιότητες (πέραν του ονόματός τους)

Συμβολισμός: <<enumeration>>

<<enumeration>> **Color**
white
black
red
green
blue

<<enumeration>> **Boolean**
False
True

<<enumeration>> **Status**
idle
working
error

Implementation in Java:

```
public enum Color  {
    WHITE, BLACK, RED, GREEN, BLUE
}
```


Εμβέλεια Κλάσης σε Λειτουργίες και Γνωρίσματα (Class Scope Operations and Attributes)

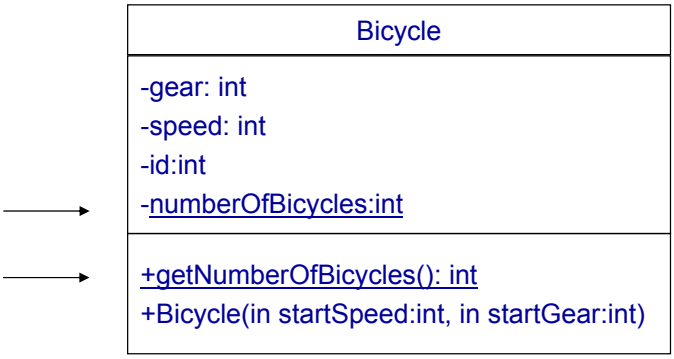
- Γνωρίσματα ή λειτουργίες που αφορούν την κλάση (και όχι τα στιγμιότυπα)
  - (class variables and methods in Java)
- Τα συμβολίζουμε υπογραμμίζοντας τα

Book
-copyID: String +title: String numberOfCopies: int

Σε Java:

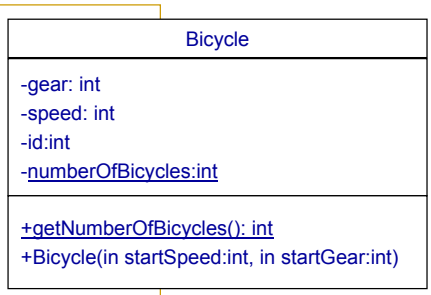
```
public class Book{
      private String copyID;
      public   String title;
      private static int numberOfCopies = 0;
}
```

παράδειγμα



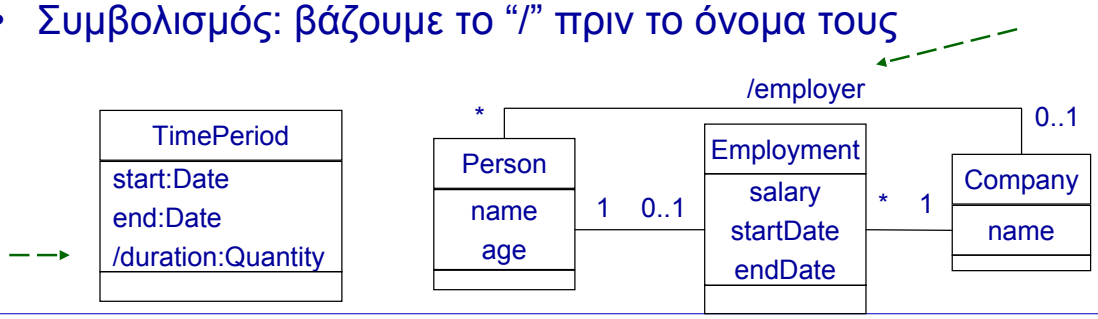
Παράδειγμα> υλοποίηση σε Java

```
public class Bicycle{
   private int gear;
   private int speed;

   private int id;
   private static int numberOfBicycles = 0;

   public Bicycle(int startSpeed, int startGear){
      gear = startGear;
      speed = startSpeed;
      // increment number of Bicycles and assign ID number
      id = ++numberOfBicycles;
   }
   public static int getNumberOfBicycles() {
         return numberOfBicycles;
   }
}
```



Παραγόμενες Συσχετίσεις και Γνωρίσματα (Derived Associations and Attributes)

- Μπορούν να υπολογιστούν από άλλες Συσχετίσεις/Γνωρίσματα
  - Π.χ. μπορούμε να υπολογίζουμε το Person.age από το Person.birthDate
- Συμβολισμός: βάζουμε το "/" πριν το όνομα τους



- προοπτική *προδιαγραφής (specification)*:
  - Τα παραγόμενα χαρακτηριστικά (derived features) υποδηλώνουν περιορισμούς μεταξύ των τιμών. Δεν καθορίζουν με ακρίβεια τι αποθηκεύεται και τι υπολογίζεται. Απλά λένε στον προγραμματιστή να σεβαστεί αυτόν τον περιορισμό.
- προοπτική *υλοποίησης (implementation)*
  - Αν υλοποιούνται αυτό γίνεται για λόγους ταχύτητας (π.χ. για αποφυγή επαναυπολογισμού)

Παγωμένο (Frozen)

- Είναι ένας περιορισμός ο οποίος μπορεί να χαρακτηρίσει ένα γνώρισμα (attribute) ή ένα άκρο συσχέτισης (association end)
  - Σε αυτήν την περίπτωση η τιμή (του γνωρίσματος/άκρου συσχέτισης) δεν πρέπει να αλλάξει κατά τη διάρκεια ύπαρξης του σχετικού αντικειμένου
  - Η αρχική του τιμή (ακόμα και είναι null) διατηρείται
  - Συνήθως οι κατασκευαστές των κλάσεων (constructors) ορίζουν αυτές τις τιμές
- Αν ο περιορισμός εφαρμοστεί σε κλάση τότε:
  - Όλα τα γνωρίσματα και άκρα συσχέτισης που σχετίζονται με αυτήν την κλάση παγιώνονται
- Frozen ≠ Read Only
  - Π.χ. Ένα γνώρισμα **age** (που είναι παραγόμενο) μπορεί να είναι read-only αλλά όχι frozen
  - (read only is not UML standard)
- Συμβολισμός: {frozen}, {read only}

Παράδειγμα υλοποίησης παγιωμένου γνωρίσματος

In Java:

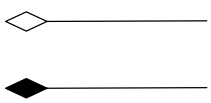
- The **static** modifier, in combination with the **final** modifier, is also used to define constants. The final modifier indicates that the value of this field cannot change.

```
static final double PI = 3.141592653589793;
```

In C++

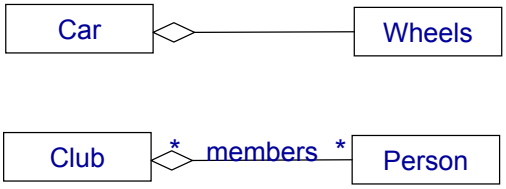
```
const float PI = 3.14159;
```


Συσσωμάτωση και Σύνθεση (Aggregation and Composition)



Συσσωμάτωση
Aggregation

Aggregation; συνάθροιση, συσσωμάτωση
Composition: συγκρότηση, σύνθεση



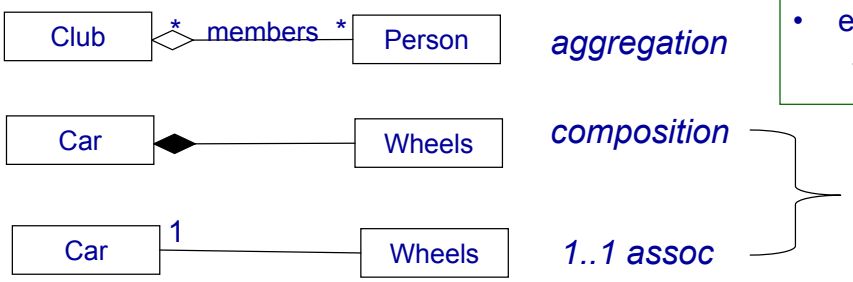
- **Aggregation** ≈ part of (δηλαδή σχέση όλου-μέρους)
- Τι διαφέρει από την απλή συσχέτιση (association);
- Πέραν της διαφορετικής απεικόνισης η UML δεν της δίνει κάποια διαφορετική σημασία
  - Due to vagueness “think of it as a modelling placebo” [3 amigos]

Συσσωμάτωση και **Σύνθεση**
(Aggregation and **Composition**)

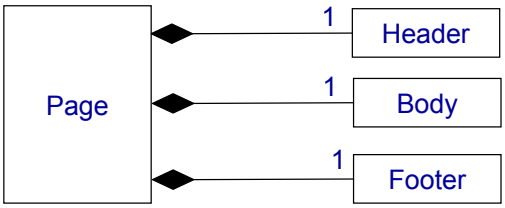
Σύνθεση: μια ισχυρότερη μορφή συσσωμάτωσης

- Το αντικείμενο-τμήμα μπορεί να ανήκει μόνο σε ένα αντικείμενο-όλο.
- Τα τμήματα αναμένεται να «ζουν» και να «πεθαίνουν» με το όλον
  - Διαγραφή του όλου συνεπάγεται διαγραφή των μερών του
  - Αυτό όμως μπορούμε να το πιάσουμε και με την πολλαπλότητα 1..1

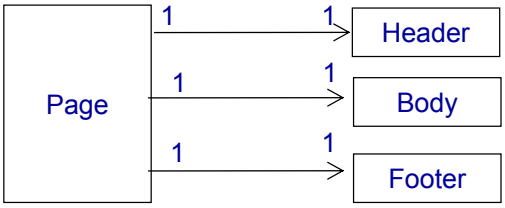
- dependent part
  - delete(X) => delete(part(X))
- exclusive part
  - partOf(X), PartOf(Y) => X=Y



Παραδείγματα σύνθεσης (και σχέση με απλή συσχέτιση με κατεύθυνση)




```
public class Page {
    public Header header;
    public Body body;
    public Footer footer;
}
```




```
public class Page {
    public Header header;
    public Body body;
    public Footer footer;
}
```

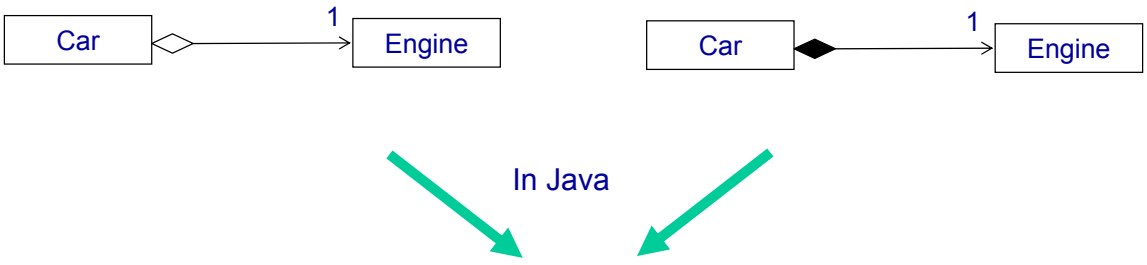

Συσσωμάτωση και Σύνθεση: Υλοποίηση σε C++

In C++

```
class Car
{
 public:
   Engine* getMyEngine();
   void   setMyEngine(Engine *eng)
 private:
   Engine*  myEngine;
}
```

```
class Car
{
 public:
   Engine getMyEngine();
   void   setMyEngine(Engine eng)
 private:
   Engine  myEngine;
}
```

Συσσωμάτωση και Σύνθεση: Υλοποίηση σε Java

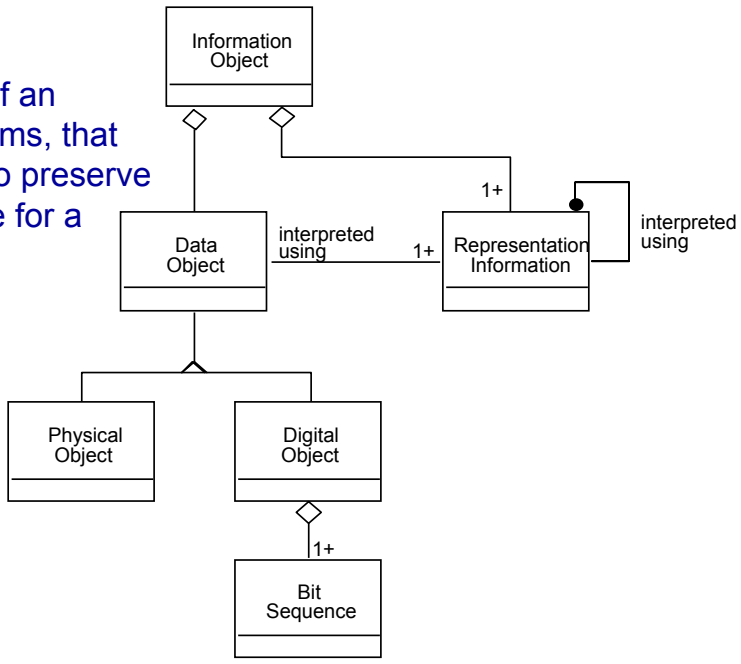



```
public class Car
{
   private Engine myEngine;
   public Engine getMyEngine();
   public void   setMyEngine(Engine eng)
}
```

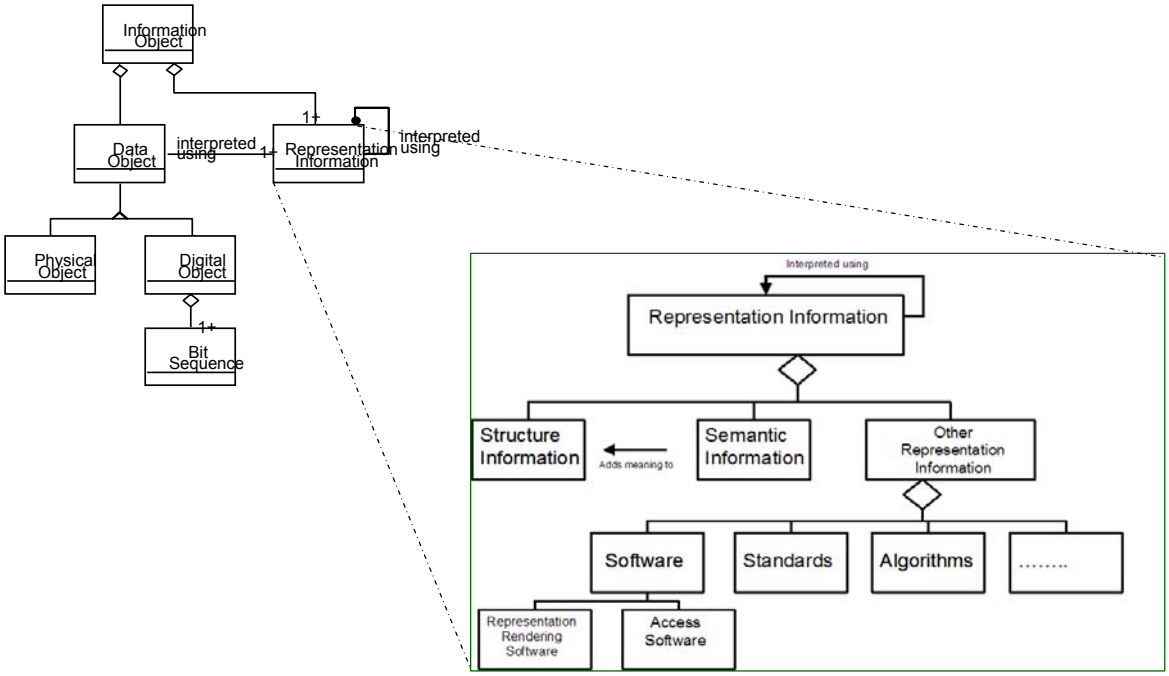
Η διαφορά είναι μόνο στο εννοιολογικό επίπεδο

Παράδειγμα διαγράμματος κλάσεων με συσσωμάτωση: OAIS Information Model

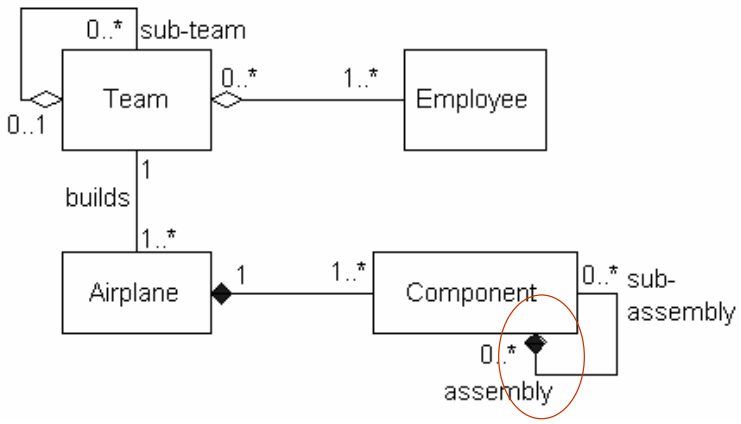
- *Reference Model for an Open Archival Information System (OAIS),* ISO 14721:2003
- OAIS = "An archive, consisting of an organization of people and systems, that has accepted the responsibility to preserve information and make it available for a Designated Community"



Παράδειγμα διαγράμματος κλάσεων με συσσωμάτωση: OAIS Information Model

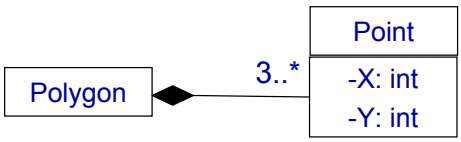


Παράδειγμα με συσσωμάτωση και σύνθεση

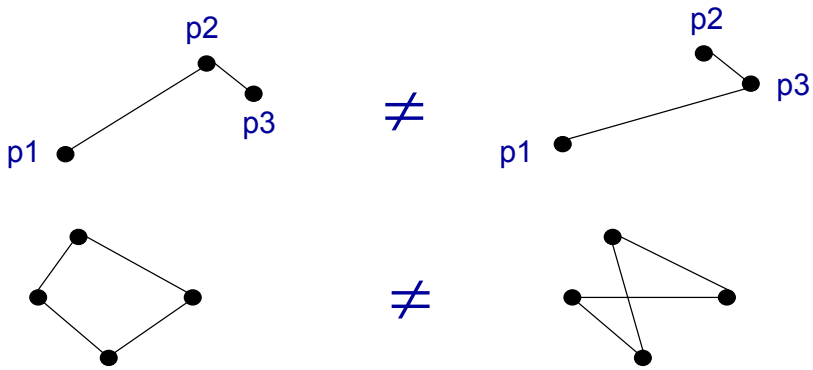


Source:http://www.agilemodeling.com/style/classDiagram.htm

Συσσωμάτωση και Σύνθεση (Aggregation and Composition)



Έστω ένα πολύγωνο που αποτελείται από 3 σημεία: p1, p2, p3. Ποιο όμως είναι;



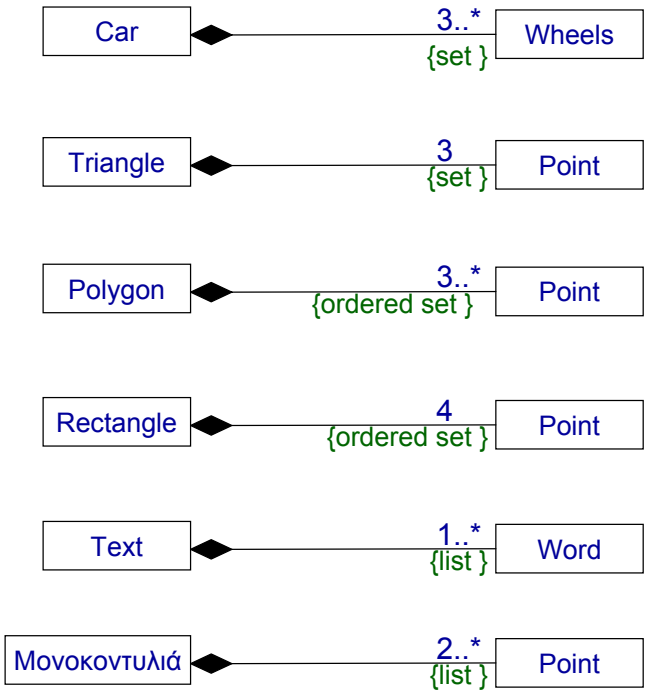
Πως θα δηλώσουμε ότι η σειρά των σημείων έχει σημασία;

Συλλογές από **πλειότιμα άκρα** συσχετίσεων (Collections for **Multivalued** Association Ends)

- **Πλειότιμο άκρο:** το πάνω όριο της πολλαπλότητα του άκρου είναι > 1 (π.χ. *)
- Κοινή σύμβαση: σημαίνει <u>σύνολο</u> (απουσία σειράς, απουσία διπλοεμφανίσεων (duplicates)).

- Μπορούμε όμως να ορίσουμε τι ακριβώς θέλουμε:
  - {set} : απουσία σειράς, απουσία διπλοεμφανίσεων
  - {ordered set} : τα αντικείμενα έχουν σειρά, απουσία διπλοεμφανίσεων
  - {bag}: επιτρέπει διπλοεμφανίσεις (ή πολλές εμφανίσεις) αντικειμένων
  - {list} ή {sequence}: τα αντικείμενα έχουν σειρά και μπορεί να υπάρχουν επαναλήψεις

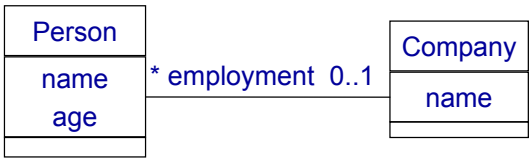
Συλλογές από **πλειότιμα άκρα** συσχετίσεων Παραδείγματα



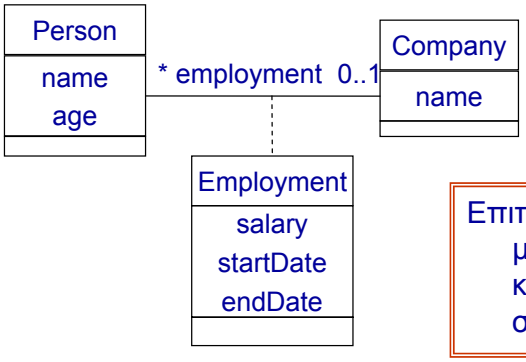
Λείπει κάτι στη μοντελοποίηση της μονοκοντυλιάς;

Κλάση Συσχέτισης (Association Class)

Είναι μια συσχέτιση στην οποία μπορούμε να προσθέσουμε γνωρίσματα, και λειτουργίες.



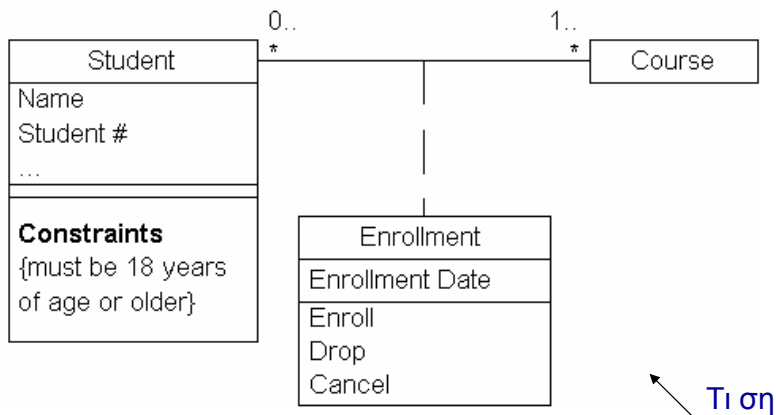
συσχέτιση



κλάση συσχέτισης

Επιπλέον, η κλάση συσχέτισης θέτει τον περιορισμό ότι μπορεί να υπάρχει το πολύ ένα στιγμιότυπο της κλάσης συσχέτισης μεταξύ οποιουδήποτε ζεύγους συσχετισμένων αντικειμένων

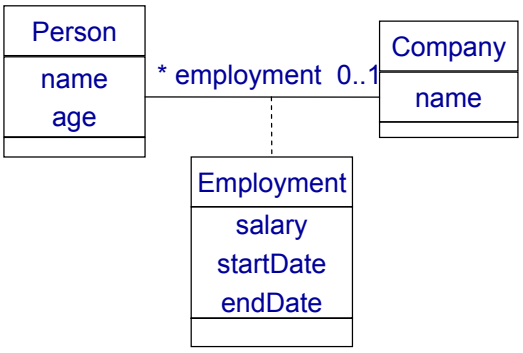
Κλάση Συσχέτισης
(Association Class)

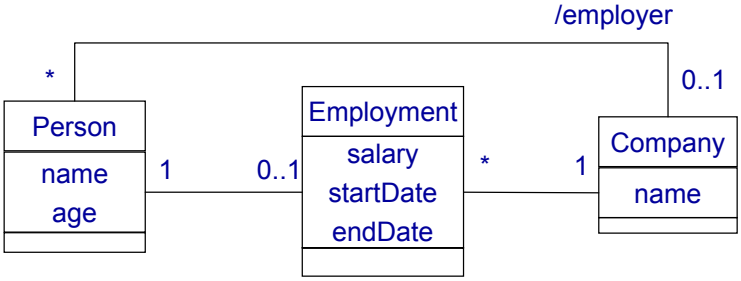


Τι σημαίνει για αυτό το διάγραμμα;

Επιπλέον, η κλάση συσχέτισης θέτει τον περιορισμό ότι μπορεί να υπάρχει το πολύ ένα στιγμιότυπο της κλάσης συσχέτισης μεταξύ οποιουδήποτε ζεύγους συσχετισμένων αντικειμένων

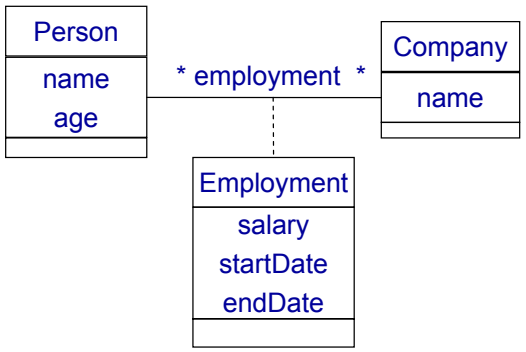
Προάγοντας μια Κλάση Συσχέτισης σε Κλάση
(Promoting an Association Class to a Full Class)





Με αυτόν τον τρόπο συνήθως τις υλοποιούμε σε γλώσσες προγραμματισμού

Προάγοντας μια Κλάση Συσχέτισης σε Κλάση
(Promoting an Association Class to a Full Class)

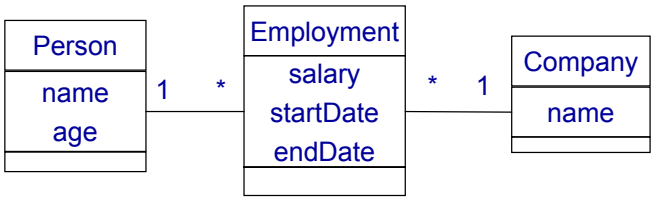


Είναι όμως ισοδύναμα;

ΟΧΙ!

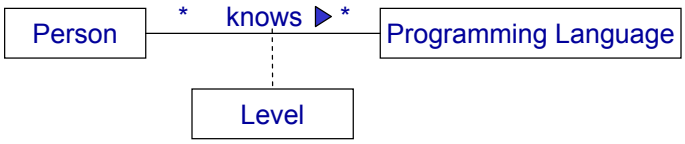
Αυτό το διάγραμμα δεν μπορεί να αναπαραστήσει την περίπτωση ενός ατόμου που έχει εργαστεί για την ίδια επιχείρηση σε διαφορετικές περιόδους

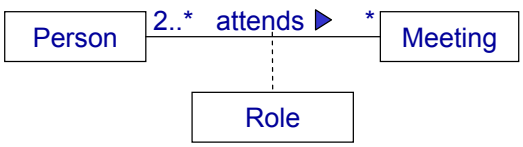
Αυτό το διάγραμμα μπορεί να αναπαραστήσει αυτήν την περίπτωση.



Κλάση Συσχέτισης
Association Class

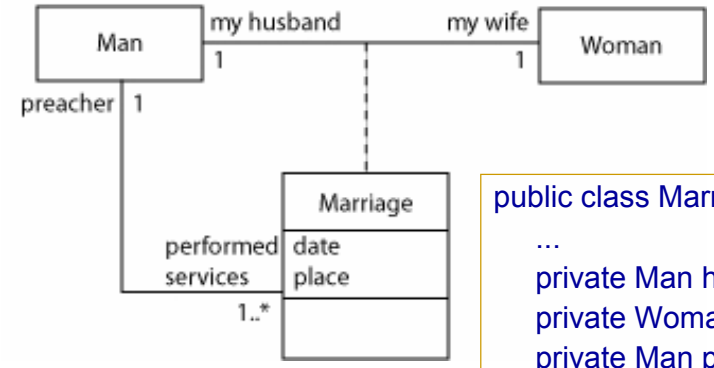
Άλλα παραδείγματα που αναδεικνύουν την χρησιμότητα του περιορισμού των κλάσεων συσχέτισης





Παράδειγμα
(πηγή: http://www.devx.com/enterprise/Article/28576)

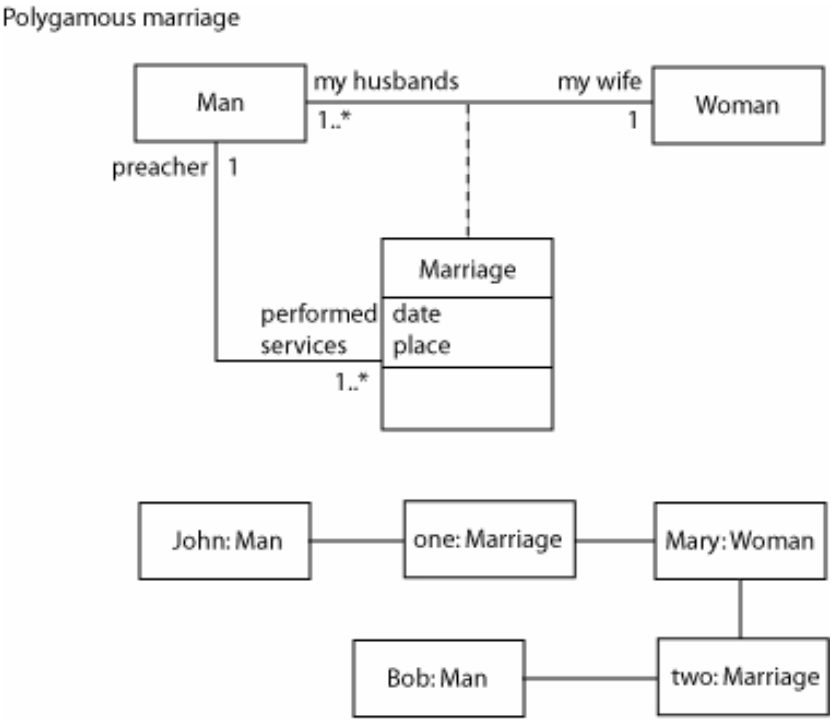
Monogamous marriage as association class



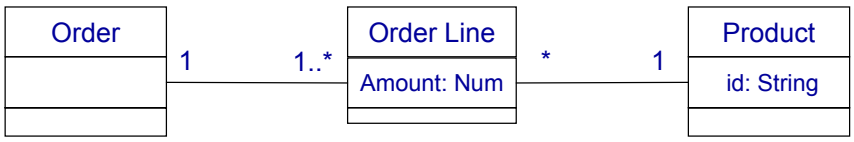

```
public class Marriage {
    ...
    private Man husband = null;
    private Woman wife = null;
    private Man prearcher = null;
    public Marriage(Man a, Woman b, Man preacher ) {
        if ( a != null && b != null && preacher !=null) {
            this.husband = a;
            this.wife = b;
            this.preacher = preacher;
        }
    }
```


Παράδειγμα
(πηγή: http://www.devx.com/enterprise/Article/28576)

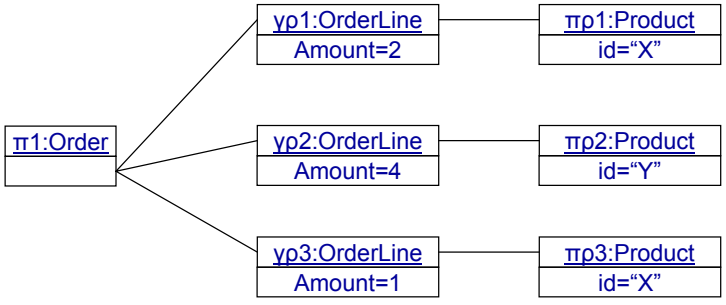
Polygamous marriage



Qualified Associations

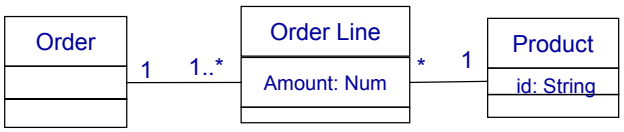


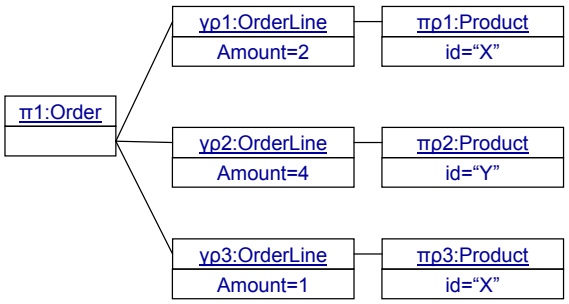
Διάγραμμα Αντικειμένων:



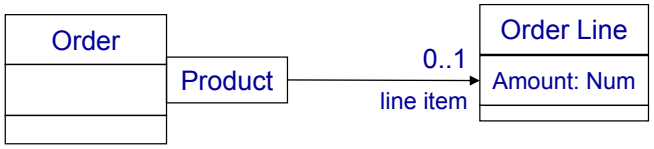
Το ίδιο προϊόν ("X") εμφανίζεται σε δύο παραγγελιογραμμές. Πως μπορούμε να το αποτρέψουμε;

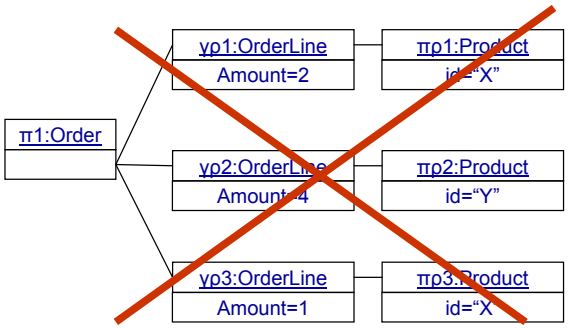
Qualified Associations



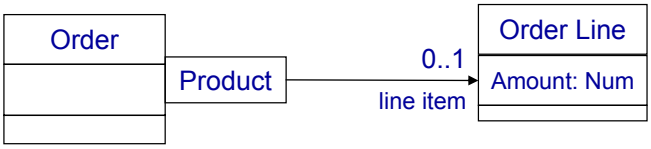


Qualified association

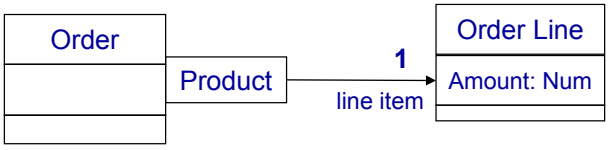




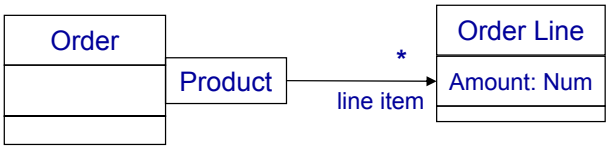
Qualified Associations & Multiplicities



*Σε μια παραγγελία υπάρχει **το πολύ μια** παραγγελιογραμμή για κάθε προϊόν*

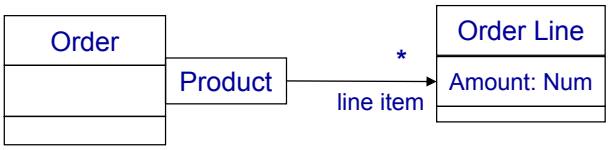


*Σε μια παραγγελία υπάρχει **ακριβώς μια** παραγγελιογραμμή για **κάθε** προϊόν.*



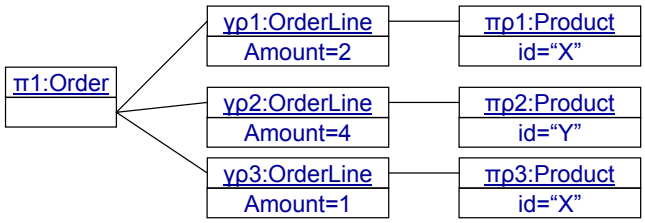
*Σε μια παραγγελία μπορεί να υπάρχει **καμία, μία ή περισσότερες** παραγγελιογραμμές για κάθε προϊόν*

Qualified Associations & Multiplicities

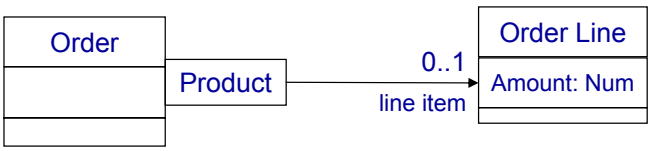


Σε μια παραγγελία μπορεί να υπάρχει ***καμία, μία ή περισσότερες*** *παραγγελιογραμμές για κάθε προϊόν*

Επιτρέπει αυτή την κατάσταση



Qualified Associations (και προοπτικές)



Εννοιολογική προοπτική:

- Σε μια Παραγγελία δεν μπορούμε να έχουμε πάνω από μια Παραγγελιογραμμή για το ίδιο Προιόν

Προδιαγραφική προοπτική:

προδιαγράφει μια διεπαφή της μορφής:

```
class Order {
    public OrderLine getLineItem (Product aProduct);
    public void addLineItem (Number amount, Product forProduct)
```

Υλοποιητική προοπτική:

```
class Order {
    private Map _lineItems;
}
```

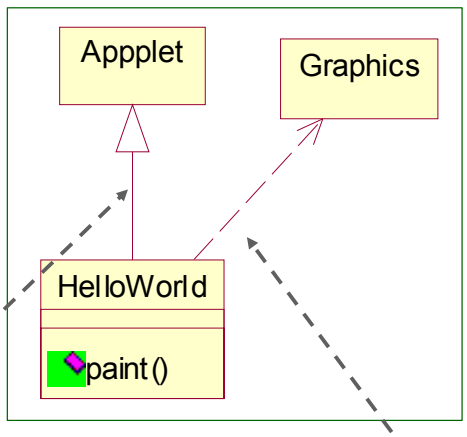
Qualified Associations
Υλοποίηση

Similar constructs of PLs

- associative arrays, maps, dictionaries
- e.g in Java:
  - public interface **Map**
  - An object that maps keys to values. A map **cannot contain duplicate keys; each key can map to at most one value.**
  - The Map interface provides three *collection views*, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings.

Εξαρτήσεις

Απαιτείται η ύπαρξη της Graphics για την αποστολή μηνυμάτων στην HelloWorld.




```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 50, 25);
    }
}
```

Γενίκευση/Εξειδίκευση

Εξάρτηση (διότι δέχεται ως παράμετρο ένα αντικείμενο τύπου Graphics)

Πολλαπλή και Δυναμική Ταξινόμηση
(Multiple and Dynamic Classification)

Πολλαπλή και Δυναμική Ταξινόμηση
(Multiple and Dynamic Classification)

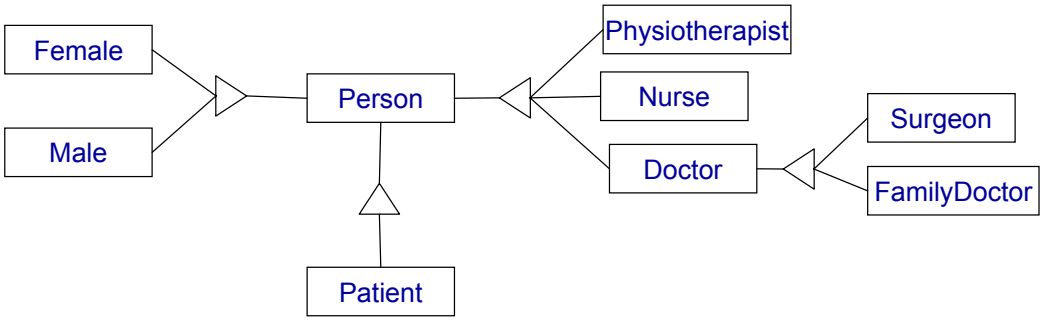
Η έννοια της **Ταξινόμησης** (Classification) αναφέρεται στη σχέση μεταξύ ενός αντικειμένου και του τύπου του.

- Απλή(Single) έναντι Πολλαπλής(multiple) ταξινόμησης
  - Απλή: ένα αντικείμενο ανήκει σε ένα τύπο
  - Πολλαπλή: ένα αντικείμενο ανήκει σε πολλούς τύπους

- Στατική(Static) έναντι Δυναμικής(Dynamic) ταξινόμησης
  - Στατική: ένα αντικείμενο δεν μπορεί να αλλάξει τύπο
  - Δυναμική: ένα αντικείμενο μπορεί να αλλάξει τύπο

Η απλή και στατική ταξινόμηση δεν είναι πολύ ευέλικτη για εννοιολογική μοντελοποίηση

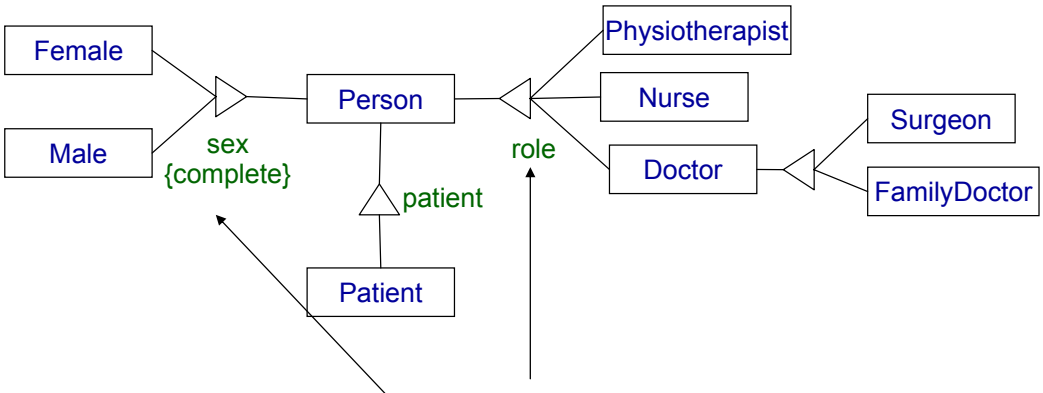
Πολλαπλή Ταξινόμηση (Multiple Classification)



A person can be female and patient and nurse

However the model allows person who are both male and female.

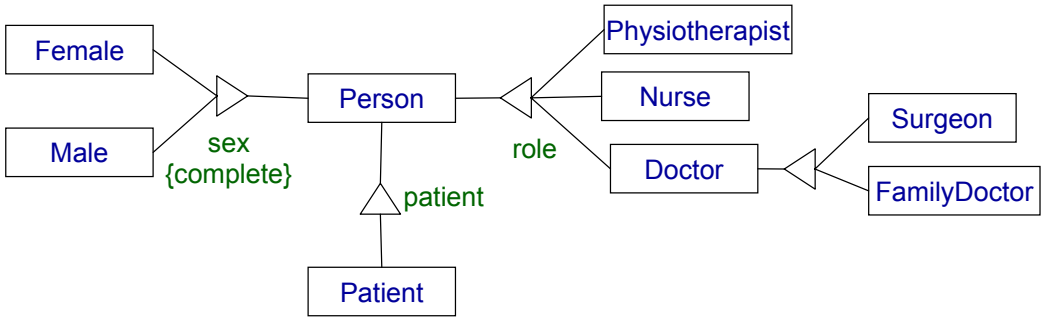
Multiple Classification
Discriminators (UML V2: Generalization set)



Discriminator:

- indication of the basis of the subtyping
- all subtypes with the same discriminator are disjoint
- if a discriminator is marked by {complete} then any instance of the superclass must be an instance of one of the sybtypes of a group (the supertype is then called abstract).

Πολλαπλή Ταξινόμηση (Multiple Classification) Discriminators

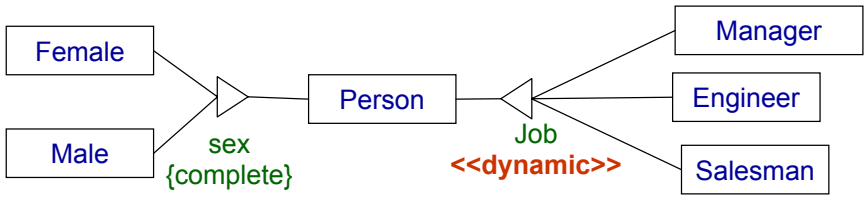


Instantiations:

- (Patient, Doctor)
  - is an illegal object because it misses sex
- (Female, Nurse, Surgeon)
  - is also illegal because it contains >1 types from the discriminator "role"

Δυναμική Ταξινόμηση (Dynamic Classification)

- allows objects to change type within the subtyping structure
  - static classification does not

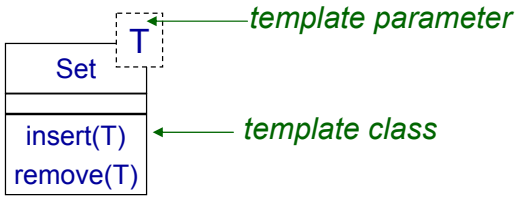


- multiple-dynamic classification needs care at the implementation perspective

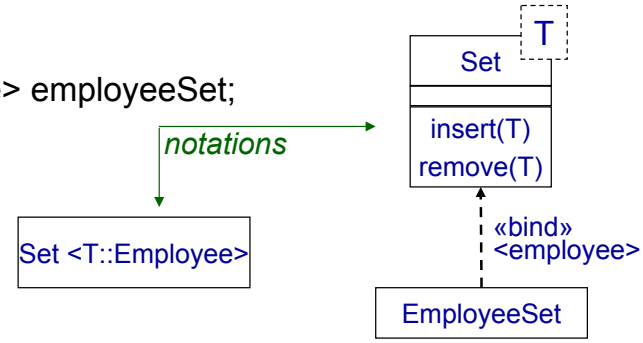
Παραμετροποιημένη Κλάση (Parameterized Class)

C++ has the notion of parameterized class or template useful for working with collections in a strongly typed language

```
Class Set <T> {
    void insert (T newElement);
    void remove (T todelElement);
```



Then we can use Set <Employee> employeeSet;
this is called bound element



Java 1.5 Generics

Διεπαφές και Αφηρημένες Κλάσεις στη UML (Interfaces and Abstract Classes in UML)

Interface: a class that has only public operations with no method bodies

- So all of its features are abstract
- Like interfaces in Java, CORBA

- Notation: <<interface>>

Abstract class: a class that cannot be directly instantiated.

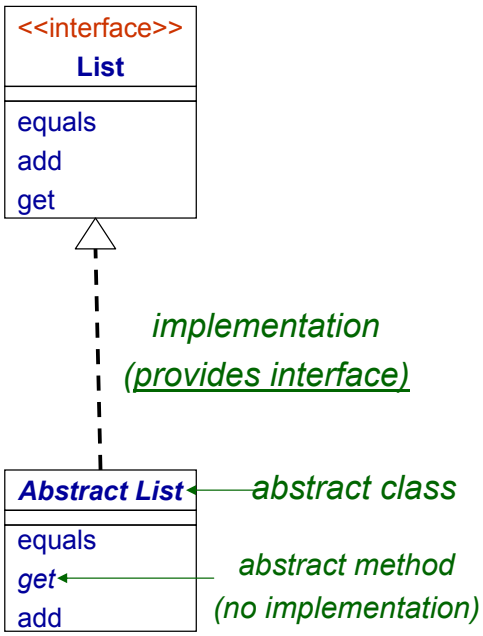
- Typically, an abstract class has one ore more operations that are abstract.
- Abstract operation: an operation with no implementation.

- Notation: *italics*, or {abstract}

Διεπαφές και Αφηρημένες Κλάσεις στη UML (Interfaces and Abstract Classes in UML)

Interface: a class that has only public operations with no method bodies

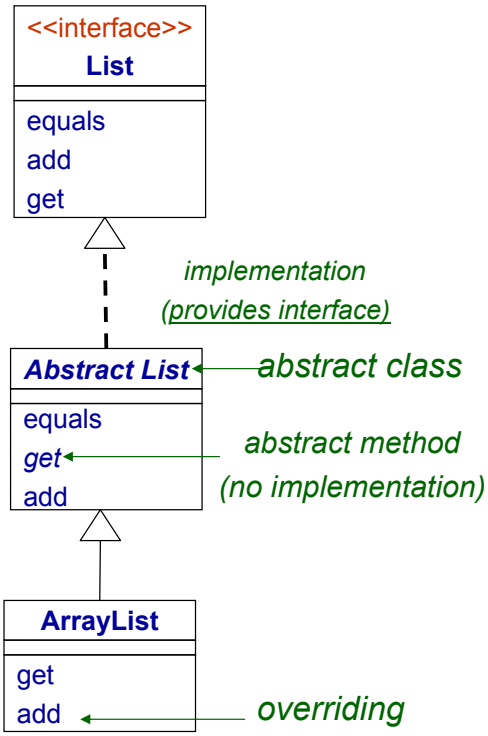
- Notation: <<interface>>



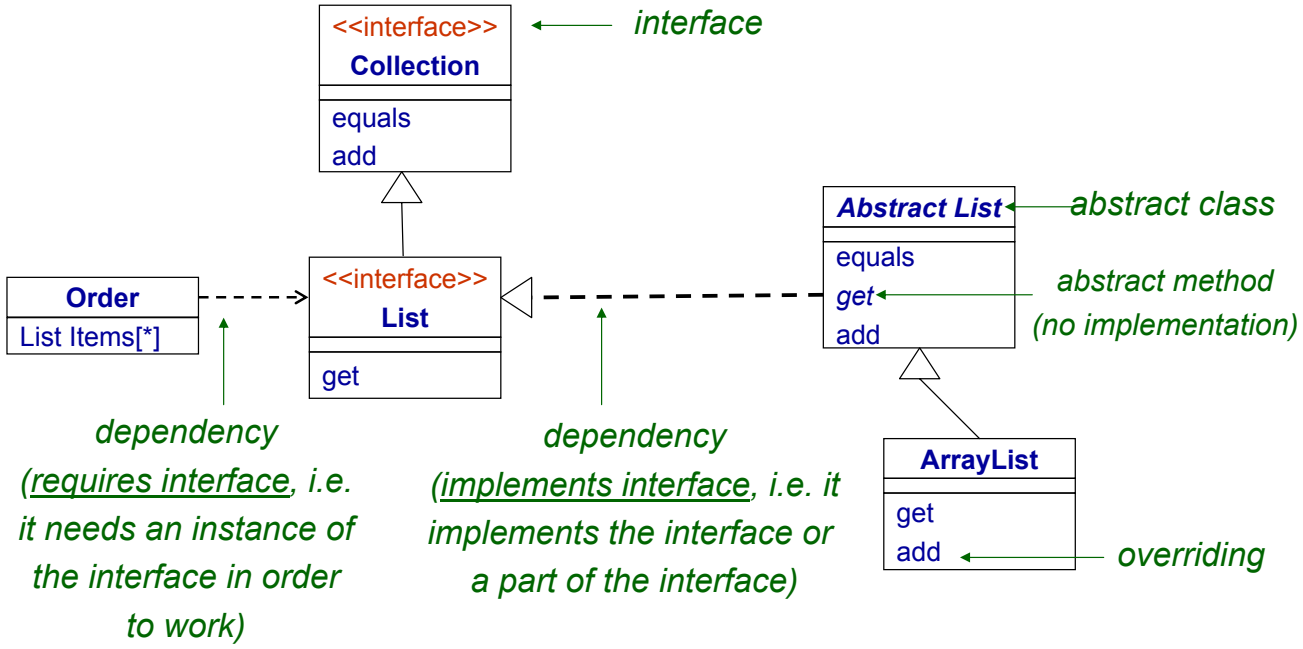
Abstract class: a class that cannot be directly instantiated.

- Notation: *italics*, or {abstract}

Διεπαφές και Αφηρημένες Κλάσεις στη UML (Interfaces and Abstract Classes in UML)

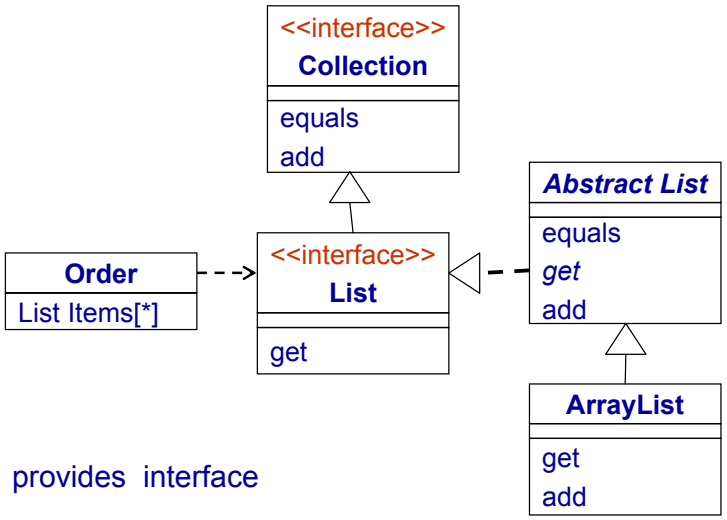


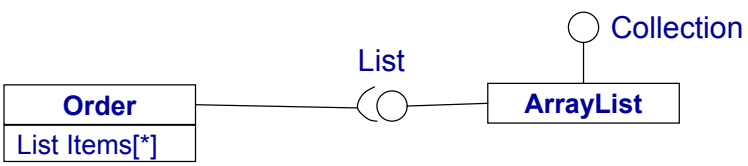
Διεπαφές και Αφηρημένες Κλάσεις στη UML (Interfaces and Abstract Classes in UML): ΠΑΡΑΔΕΙΓΜΑ



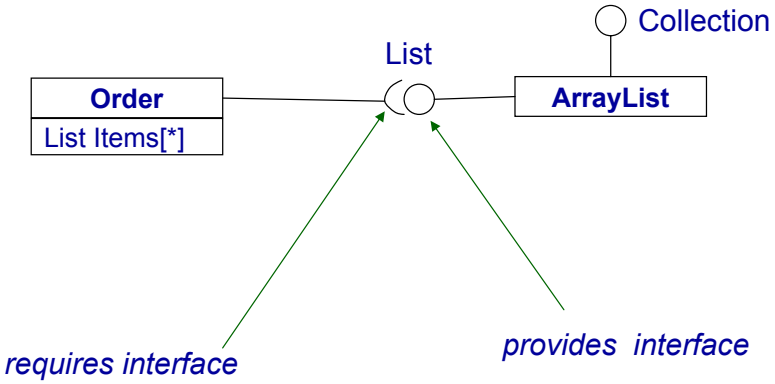
Dependency: w.r.t. updates (if the interface changes the implementation class should change too).

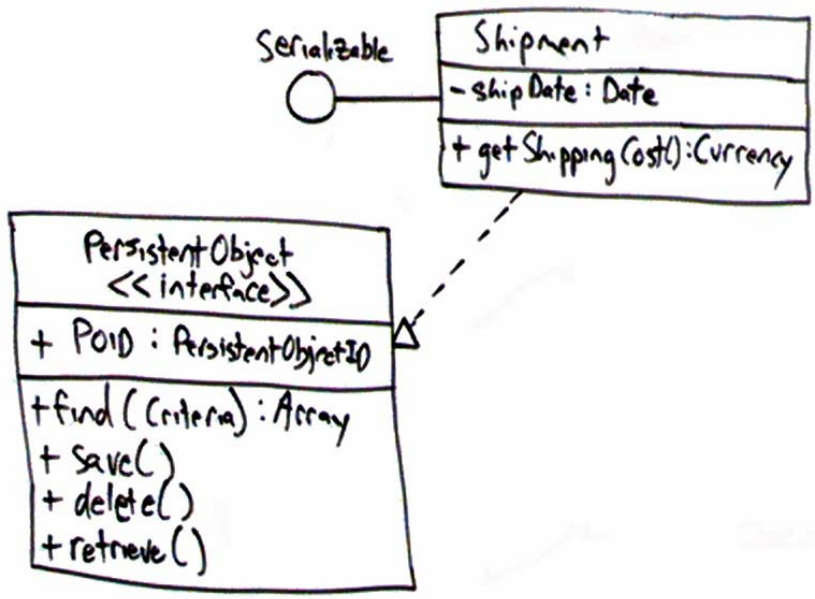
Διεπαφές και Αφηρημένες Κλάσεις Μια πιο συμπαγής αναπαράσταση



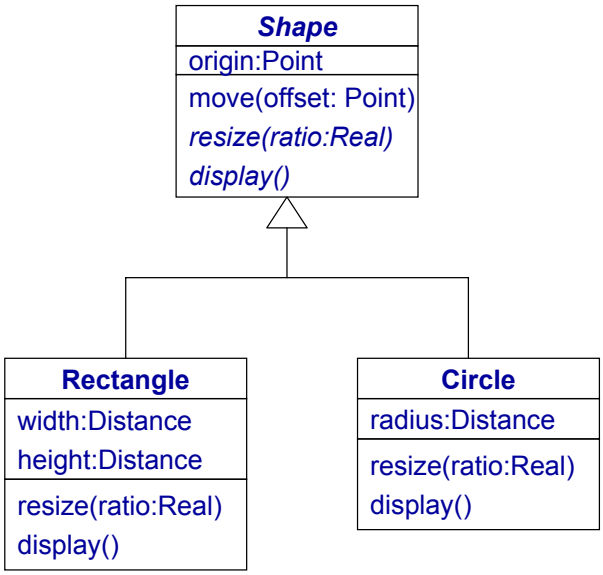


Διεπαφές και Αφηρημένες Κλάσεις: Παράδειγμα
Μια πιο συμπαγής αναπαράσταση

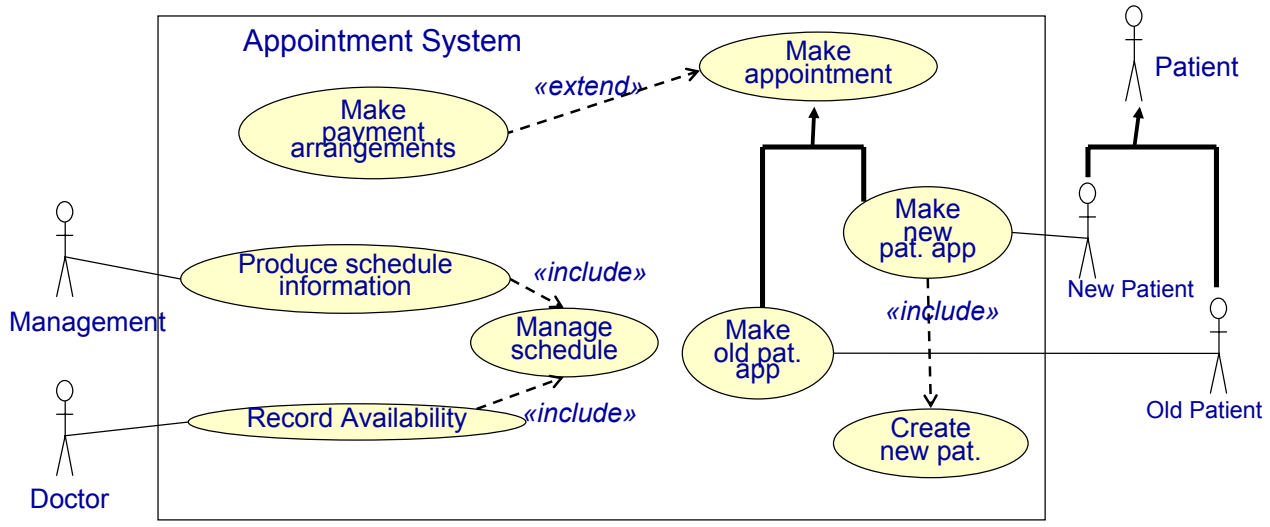




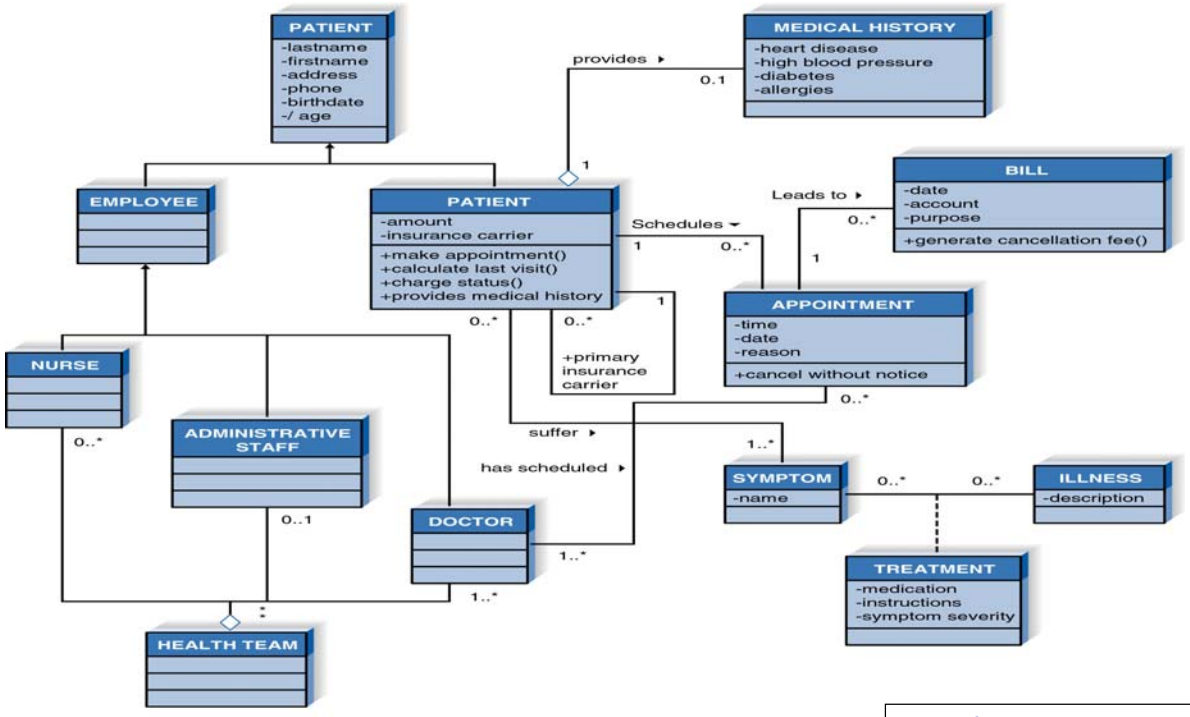
..



Παράδειγμα: Use Case Diagram for an Appointment System

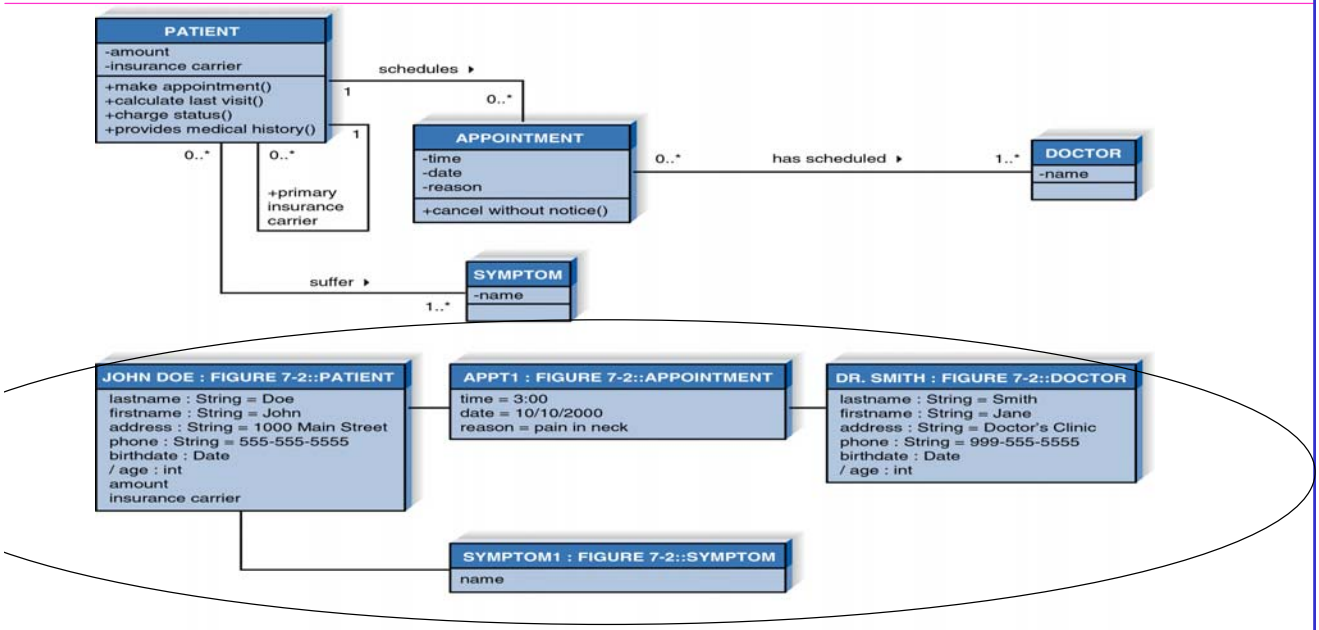


Παράδειγμα:
Class Diagram for the Appointment System



Taken from Dennis et al. 2005

Ένα τμήμα του διαγράμματος αντικειμένων
A part of the Object Diagram



Dennis: SAD
Fig: 7-5 W-27 100% of size
Fine Line Illustrations (516) 501-0400

Taken from Dennis et al. 2005

Παράδειγμα

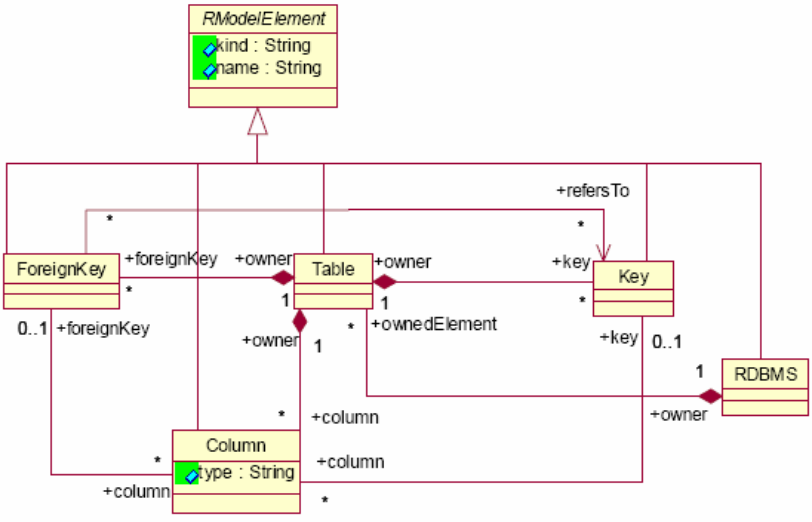


Figure 12 Example of a relational DBMS metamodel

Μερικά μεθοδολογικά ζητήματα για τη σύνταξη διαγραμμάτων κλάσεων

Από Ευθύνες σε Γνωρίσματα και Λειτουργίες (from Responsibilities to Attributes and Operations)

FraudAgent
Responsibilities -- determine the risk of a customer order -- handle customer-specific criteria for fraud

FraudAgent
?
?

- When drawing a class we don't have to show every attribute and every operation at once.
- We can choose to show only some or none of a class's attributes and operations.

Οργάνωση Γνωρισμάτων και Λειτουργιών (Organizing Attributes and Operations)

To better organize long list of attributes and operations, we can prefix each group with a descriptive category by stereotypes

FraudAgent
<<constructor>> new() new(p: Policy) <<process>> process(o:Order) ... <<query>> isSuspect(o:Order) isFraudulent(o:Order) <<helper>> validateOrder(o:Order)

Πηγές και Αναφορές

- **Systems Analysis and Design with UML Version 2.0** (2nd edition) by A. Dennis, B. Haley Wixom, D. Tegarden, Wiley, 2005. CHAPTER 7
- **UML Distilled: A Brief Guide to the Standard Object Modeling Language** (3rd Edition) by Martin Fowler, Addison Wesley, 2004. Chap. 3
- **The Unified Modeling Language User Guide** (2nd edition) by G. Booch, J. Rumbaugh, I. Jacobson, Addison Wesley, 2004, Chap 8 (advanced: 9-10)
- Δείτε Επίσης
  - Υλοποίηση συσχετίσεων σε Java: http://www.jot.fm/issues/issue_2003_09/article4.pdf