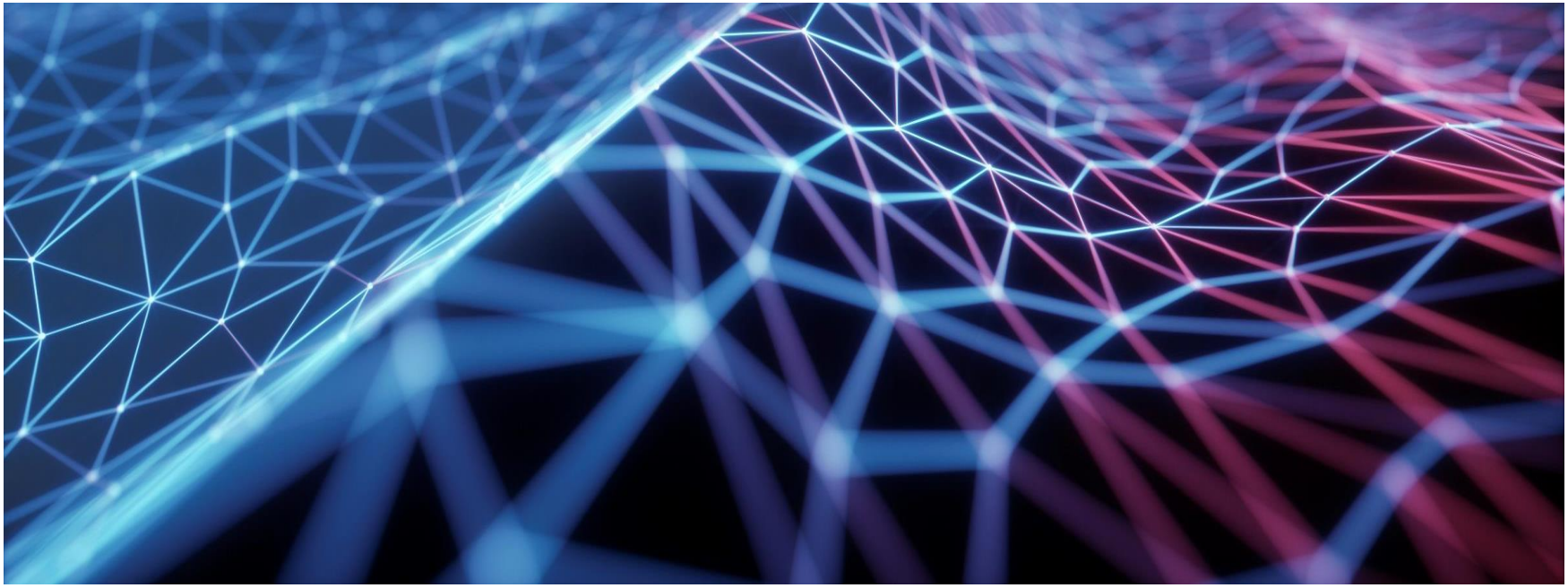
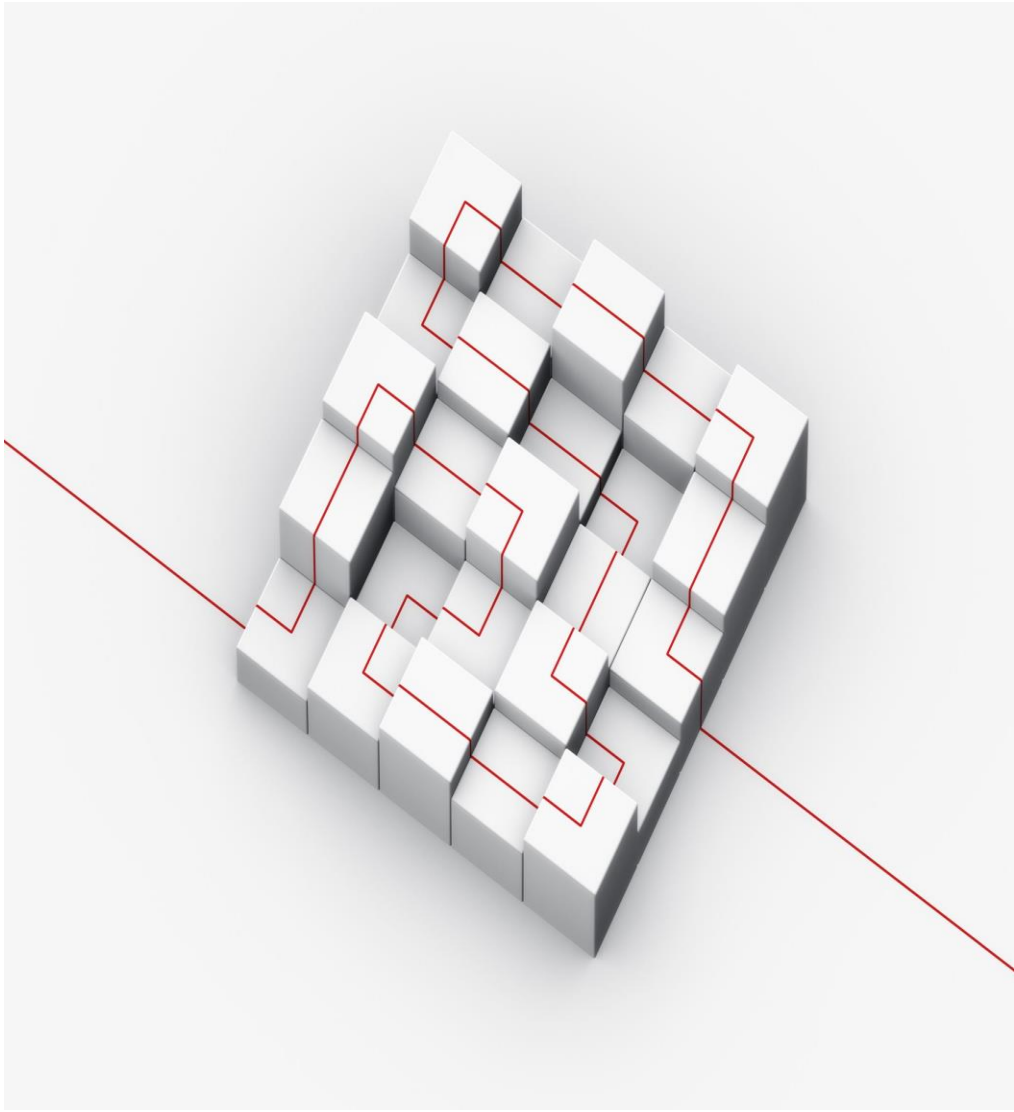
DIS501 Ανάλυση κι Σχεδίαση Πληροφοριακών Συστημάτων

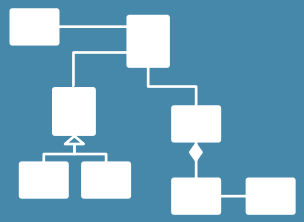
4η Τηλεσυνάντηση:
– Διάγραμμα κλάσεων (conceptual level) και Διαγράμματα Ακολουθίας Αντικειμένων



ΒΥ: ΔΡ ΑΥΓΟΥΣΤΑ ΚΥΡΙΑΚΙΔΟΥ ΖΑΧΑΡΟΥΔΙΟΥ

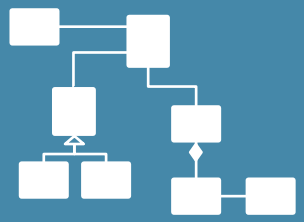
Conceptual Class Diagrams (CCDs)





Conceptual Class Diagrams
not the same as Design Class Diagrams

- ΜΗΝ μπερδεύετε το CCD με τις τάξεις που τελικά κατασκευάζετε.
- Οι κλάσεις στα CCDs είναι η ΒΑΣΗ για αυτές τις κλάσεις, αλλά το πρώτο διάγραμμα κλάσεων που σχεδιάζετε/μοντελοποιείτε για το έργο σας ΔΕΝ θα είναι ακριβώς όπως αυτό που τελικά κωδικοποιείτε
- Τα Conceptual class diagrams μοντελοποιούν **Entity/Business classes μόνο**
- **Κατά τη διάρκεια του σχεδιασμού (design)**
  - συνεχίζουμε να βελτιώνουμε τις κλάσεις οντοτήτων (entity classes) και τις συμπεριφορές τους (μεθόδους τους)
  - Προσδιορίζουμε νέα είδη αντικειμένων/κλάσεων που απαιτούνται για τις αποφάσεις εφαρμογής του νέου συστήματος
  - **Boundary/Interface classes** and **Control classes**



Conceptual Class Diagram - CCD
Κύρια δομικά συστατικά

- **Classes (Entity/Business classes)**
  αντιπροσωπεύουν τα είδη των αντικειμένων

- **Associations**
  αναπαριστούν σχέσεις (αλληλεπίδραση) μεταξύ κλάσεων

- **Attributes**
  είναι δεδομένα (που θα θέλαμε να αποθηκεύσουμε) και τα οποία βρίσκονται σε κλάσεις. Το όνομα κάθε χαρακτηριστικού μπορεί να ακολουθείται από προαιρετικές λεπτομέρειες όπως ο τύπος και η προεπιλεγμένη τιμή (type and default value).

- **Operations or Methods**
  λειτουργίες που εκτελούνται από τις κλάσεις και τα στιγμιότυπά τους. Τι μπορούν να κάνουν τα αντικείμενα της κλάσης; Σε τι θα ανταποκριθούν τα αντικείμενα της κλάσης;

- **Generalisation/Specialisation**
  ομαδοποίηση κλάσεων σε ιεραρχίες κληρονομικότητας

Step 1. Collecting Domain Information

Customer presentation

business forms

operating procedures

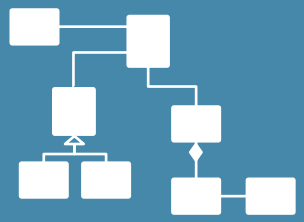
regulations & standards

Literature survey

Stakeholder survey

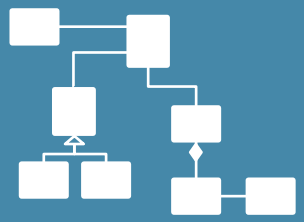
User interviews

Writing user stories



Entity (business) Classes
Που τις βρίσκουμε? Που να κοιτάξουμε?

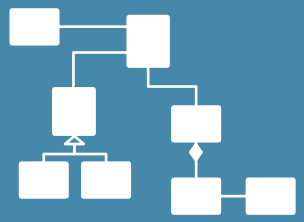
- **Things** that have common, similar attributes, behaviours, and common semantics (e.g. Employee, Product)
- **Occurrences or events** that are part of the information domain for the problem (e.g. payroll system initiating bank transfers)
- **Roles** played by people who interact with the system(e.g. managers, students)
- **Organisational units** (e.g. working teams, departments) that are relevant to the system under development
- **Places** (e.g. a manufacturing floor) that establish the context of the problem and the overall function of the system
- **Structures** that define a class (e.g. four-wheeled vehicles, building, book/chapters)



Step 2. Brainstorming

Η ομάδα συγκεντρώνεται και εντοπίζει:

- **Εξειδικευμένα ουσιαστικά και φράσεις ουσιαστικών** που σχετίζονται με τον τομέα/σύστημα. Ουσιαστικά = πιθανές κλάσεις
- **Εκφράσεις της μορφής "Χ του Υ"** (όπου Χ και Υ είναι ουσιαστικά ή φράσεις ουσιαστικών, π.χ., **χρώμα του αυτοκινήτου**).
- **Μεταβατικά ρήματα** (π.χ., **οι φοιτητές εγγράφονται σε μαθήματα**). – ρήματα = πιθανές μέθοδοι κλάσεων
- **Επίθετα**
- **Αριθμητικές και ποσοτικές εκφράσεις**.
- **Εκφράσεις κτήσης** (π.χ., **έχει/διαθέτει/κατέχει**).
- **Συστατικά στοιχεία και σχέσεις "μέρος του"**.
- **Εκφράσεις περιεκτικότητας ("περιέχει/περιλαμβάνει")**.
- **Εκφράσεις της μορφής "Χ είναι Υ"** (όπου Χ και Υ είναι ουσιαστικά ή φράσεις ουσιαστικών).

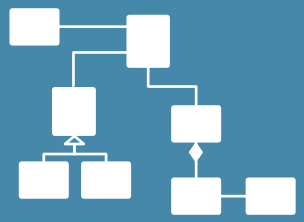


Identifying Entity/Business Classes
Που τις βρίσκουμε? Που να κοιτάξουμε?

- Κοιτάζουμε τα use-case diagrams
  - οι περιπτώσεις χρήσης είναι το καλύτερο μέρος για να αναζητήσετε κλάσεις/αντικείμενα οντοτήτων.
  - σκεφτείτε τις αλληλεπιδράσεις μεταξύ αντικειμένων που υποστηρίζουν μια περίπτωση χρήσης
  - βασιστείτε στο διάγραμμα της περίπτωσης χρήσης και στις περιγραφές σεναρίων για να βρείτε ένα σύνολο κλάσεων που μπορούν να αλληλεπιδράσουν προκειμένου να υλοποιηθεί η εν λόγω περίπτωση χρήσης
  - **Για κάθε περίπτωση χρήσης και σενάριο περίπτωσης χρήσης βρείτε "ουσιαστικά" που αντιστοιχούν σε επιχειρηματικά γεγονότα. Τα ουσιαστικά αντιστοιχούν σε κλάσεις.**
  - **Τα ρήματα (Verbs)** μας δίνουν στοιχεία για τις **μεθόδους (methods)**

- Εξαγωγή κλάσεων από τις απαιτήσεις των πελατών -
- Κάρτες CRC – Collaboration and responsibility cards

Nouns: member, order
Verbs: place_order



Collaboration and Responsibility Cards

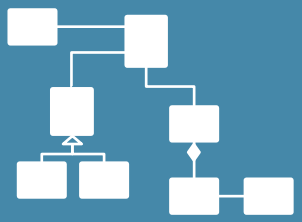
identifying Entity classes

- Μια τεχνική αντικειμενοστρεφούς ανάλυσης που βοηθά τους developers να αναγνωρισουν κλάσεις, τις συμπεριφορές τους και τις συσχετίσεις τους με άλλες κλάσεις

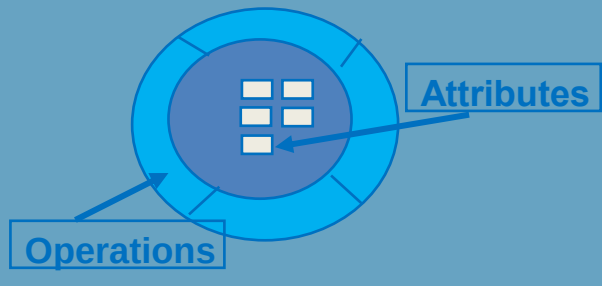
- 3 βασικα μέρη:
  - **Class (κλάσεις):** το όνομα βρίσκεται το πάνω μέρος της κάρτας. Μια κάρτα για κάθε κλάση
  - **Responsibilities (ευθύνες):** κάτι που η κλάση γνωρίζει ή κάνει
  - **Collaborator (Συνεργάτες):** μια άλλη κλάση με την οποία η τρέχουσα κλάση πρέπει να συνεργαστεί για να ολοκληρώσει τις ευθύνες της.
    - Μια κλάση που έχει πληροφορίες που χρειαζόμαστε
    - Που βοηθάει στην εκτέλεση μιας εργασίας

Class Name	
Responsibility	*Collaborator*
Κατάλογος με τις ευθύνες αυτής της κλάσης	Τα ονόματα των κλάσεων που θα εξυπηρετήσουν αυτή την κλάση

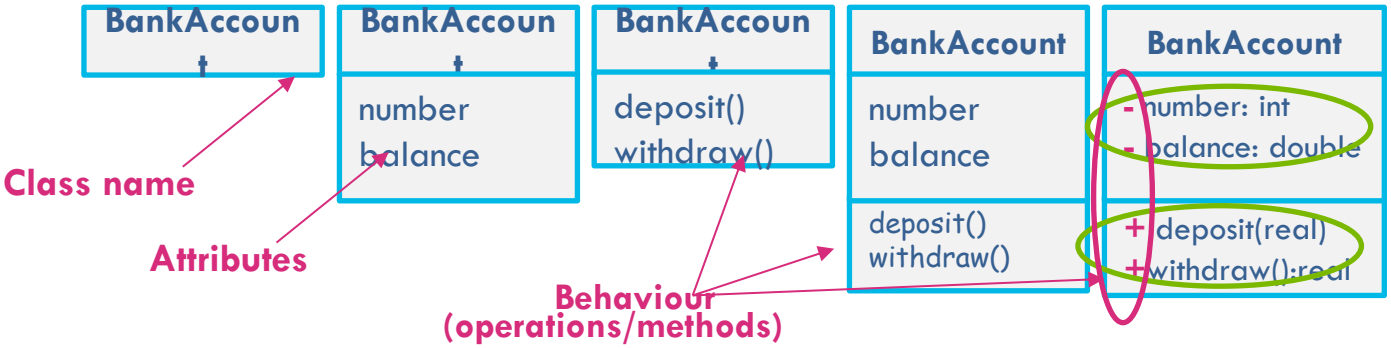
Class Book	
Responsibility	Collaborator
Να διατηρεί τον τίτλο, τον συγγραφέα και τον αριθμό αντιτύπων	***LibraryCatalog (για να ενημερώνεται ο κατάλογος όταν αλλάζει η διαθεσιμότητα)***
Να ενημερώνει τη διαθεσιμότητα του βιβλίου	***Member (για να δεσμευτεί ή να επιστραφεί ένα βιβλίο)***



UML Notation
Class - Κλάσεις



- Απλά αναπαρίσταται ως ένα κουτί με το όνομα της κλάσης μέσα
  - το οποίο δείχνει επίσης τα χαρακτηριστικά (attributes) και τις λειτουργίες (attributes and operations)
  - the complete signature of an attribute /operation is:
    - <attributeName>:<type>
    - <operationName(parameterName:paramType…)>:<returnType>
    - **Visibility**: private (**-**), public (**+**), protected (**#**), or package(**~**)



Attributes and Operations/Methods

Οι πιο συνηθισμένες μεθόδοι σε κάθε Κλάση είναι:

- Get (View/Display)
- Set (Edit)
- Create or Add (Constructor)
- Delete (Destructor)

Example: Class Door

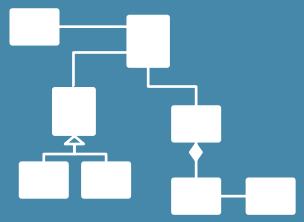
Attributes: Woodtype, Colour

Operations: Open, Shut, Lock, Unlock

We don't define primary and foreign keys

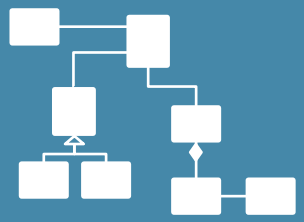
Staff
Staff_name: String Staff_no: Integer
Create/Add() Set/Edit() Delete() Get/Display()

- **Όλες αυτές οι συμπεριφορές (μια πόρτα μπορεί να ανοίξει, ή μπορεί να κλειδώσει κ.λπ.) σχετίζονται με την κλάση πόρτα και επιτυγχάνονται από την πόρτα και από κανένα άλλο αντικείμενο!!!**
- Σημαντικό όταν σχεδιάζετε τις κλάσεις σας. Μέθοδοι σχετικές μόνο με τη συγκεκριμένη κλάση. Για οτιδήποτε άλλο η κλάση πρέπει να αλληλοεπιδρά/συσχετίζεται με άλλες κλάσεις



Conceptual Class Diagram
Σχέσεις μεταξύ κλάσεων

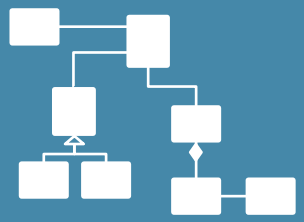
- Αντιπροσωπεύουν συνδέσεις μεταξύ πολλαπλών κλάσεων ή μιας κλάσης και του εαυτού της
- 3 basic categories:
  - σχέσεις συσχέτισης (association relationships)
  - σχέσεις γενίκευσης/ειδίκευσης (generalisation/specialisation relationship)
  - σχέσεις συνάθροισης/σύνθεσης (Aggregation/composition relationships)



Class Relationships
Σχέση Συσχέτισης - Association

- **Association:** σύνδεση μεταξύ κλάσεων, εκφράζει κάποια σχέση μεταξύ των δύο, προκύπτει από κάποιο ρήμα στις προδιαγραφές
  - Δείχνει πως δυο κλάσεις συσχετίζονται
  - Πως αλληλεπιδρούν μεταξύ τους

- Criteria for Useful Associations
  - Για να προσδιορίσετε ελέγξετε την περιγραφή του προβλήματος (ή τα σενάρια περιπτώσεων χρήσης)
  - Εντοπίστε τα ρήματα (having verbs) τα οποία μας δείχνουν σχέσεις μεταξύ δύο κλάσεων



Associations
identifying association relations

- Πότε υπάρχει συσχέτιση μεταξύ μιας κλάσης Α και μιας κλάσης Β:
  - αντικείμενο της Α στέλνει μήνυμα σε αντικείμενο της Β
  - αντικείμενο της Α δημιουργεί αντικείμενο της Β
  - αντικείμενο της Α έχει πεδίο δεδομένων με τιμές αντικείμενα της Β
  - αντικείμενο της Α δέχεται μήνυμα με παράμετρο αντικείμενο της Β

- Γενικά το αντικείμενο της Α πρέπει να γνωρίζει κάποιο αντικείμενο της Β

Σχέση Συσχέτισης - Association

UML Notation

- Μια συσχέτιση αναπαρίσταται ως μια γραμμή μεταξύ κλάσεων με ένα όνομα συσχέτισης

Συσχέτιση από το A στο B σημαίνει ότι το A χρησιμοποιεί το B άμεσα (για παράδειγμα, καλώντας τις μεθόδους του))

  - Κάθε συσχέτιση μπορεί να σημανθεί (labelled), για να γίνει σαφής η φύση της συσχέτισης
  - Οι συσχετίσεις είναι εγγενώς αμφίδρομες - δηλαδή, από τα στιγμιότυπα της μιας ή της άλλης κλάσης είναι δυνατή η λογική μετάβαση στην άλλη.

Αυτή η διαδρομή είναι καθαρά αφαιρετική- δεν αποτελεί δήλωση σχετικά με τις συνδέσεις μεταξύ οντοτήτων λογισμικού

Σχέσεις Συσχέτισης(Associations) και Πληθικότητα (Multiplicity)
UML Notation

- Συνήθως στο τέλος μια συσχέτισης υπάρχει ένδειξη πληθικότητας(multiplicity) η οποία υποδηλώνει μια αριθμητική σχέση μεταξύ των αντικειμένων των κλάσεων

Ένας PA εργάζεται για 1 διευθυντή
Ένας διευθυντής μπορεί να έχει πολλούς PA.

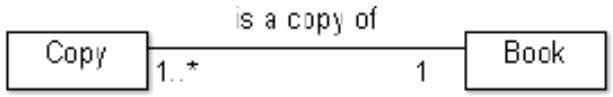
Μια παραγγελία περιέχει πολλά προϊόντα και ένα προϊόν ανήκει σε πολλές διαφορετικές παραγγελίες.

Σχέσεις συσχέτισης Associations - Πληθικότητα (Multiplicity)
UML Notation

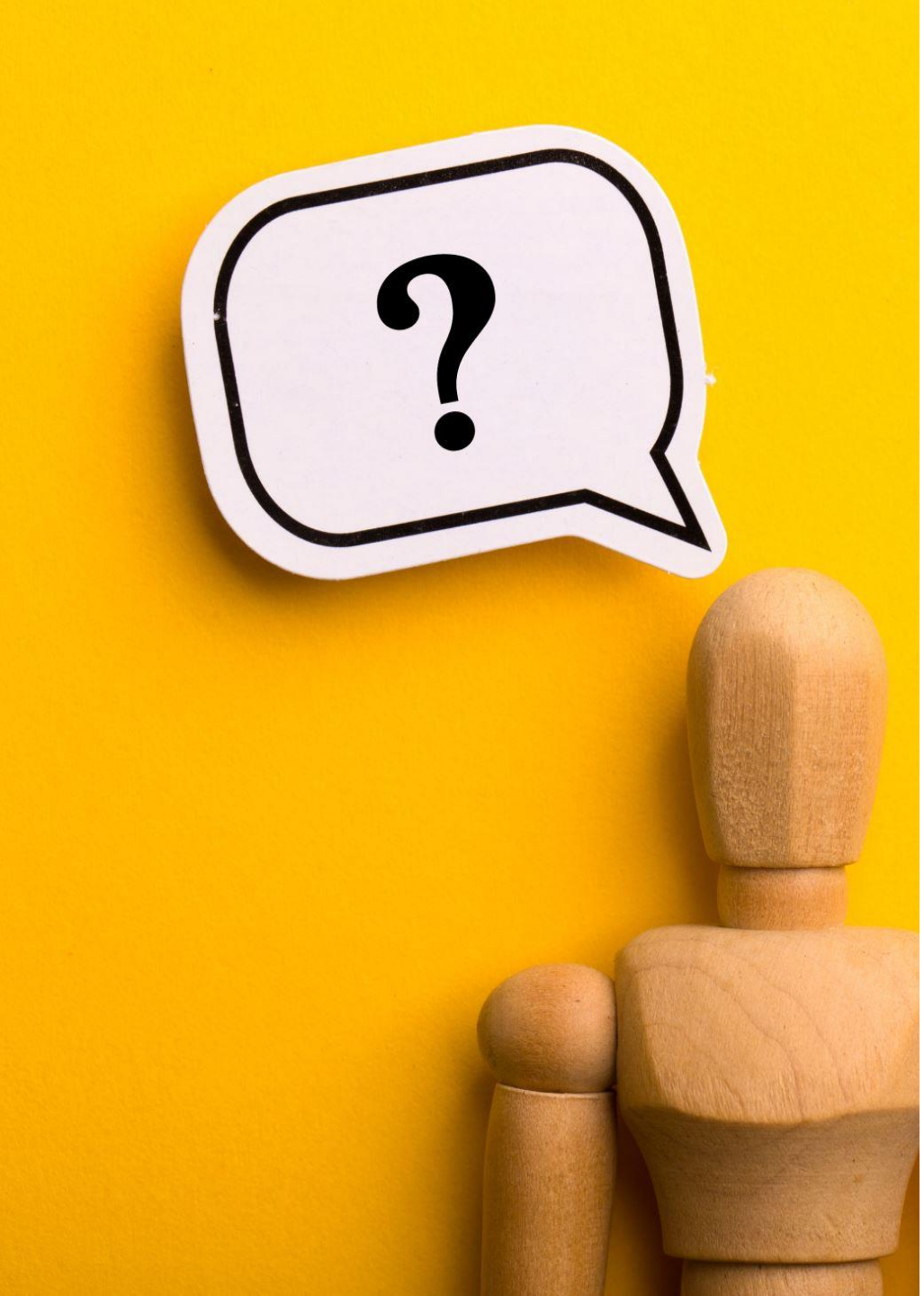
- Δηλώνει τον **ελάχιστο .. Μέγιστο αριθμό** περιπτώσεων που μπορεί να έχει μια συσχέτιση κλάσης.

0..* or 0..m or *	Zero or more
1..* or 1..m	One or more
0..1	Zero or one
2..4	Specified range
1..3, 5	Multiple, disjoint ranges
1 or no number or range	Exactly one

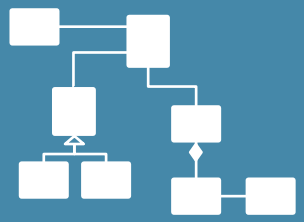
Μπορούμε να χρησιμοποιήσουμε:
- συγκεκριμένους αριθμούς
- εύρος αριθμών
- απροσδιόριστο πλήθος: *



Καλύτερα να χρησιμοποιήσουμε το …* από το ..m., παρόλο που και τα δύο είναι σωστά. Το * χρησιμοποιείται από τα UML εργαλεία



BREAKOUT SESSION



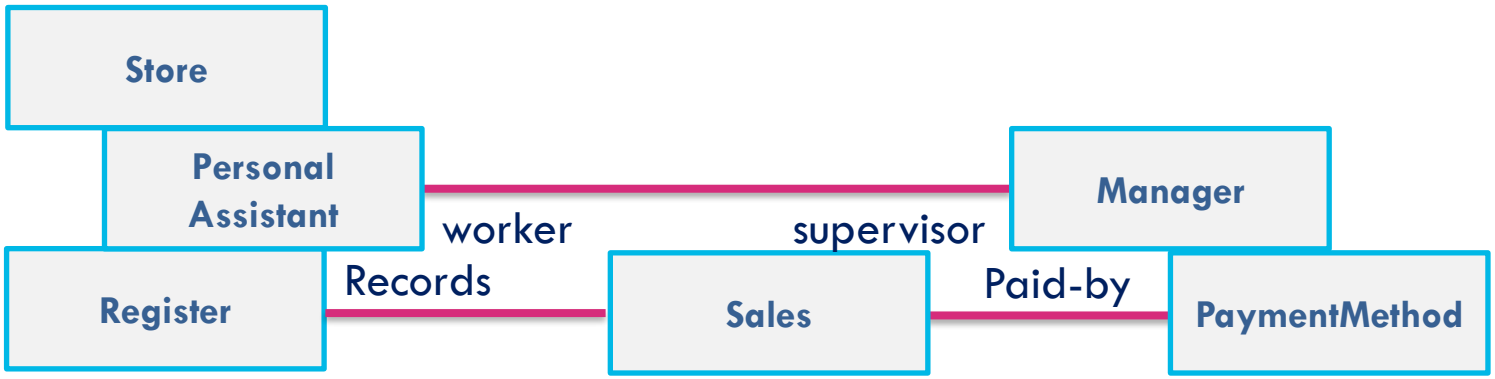
Βρέστε το Multiplicity μεταξύ αυτών των κλάσεων

Κλάση Α	Κλάση Β	Multiplicity
Patient	Appointment	1: 1..*
Doctor	Appointment	1: 1..*
Patient	Room	* : 1
Course	Student	*: *

Ονομασία Συσχετίσεων - Labelling Associations
UML Notation

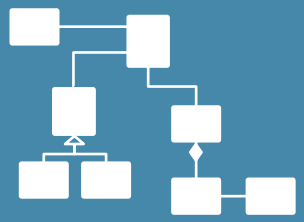
- Είναι καλό να ονομάζουμε μια συσχέτιση με το να χρησιμοποιούμε μια ρηματική φράση (*VerbPhrase*)

- Κάθε άκρο μιας συσχέτισης ονομάζεται ρόλος
  - κάθε άκρο μπορεί προαιρετικά να σημανθεί αναλόγως για να καταστήσει σαφή τη φύση του ρόλου



Multiple Associations
between classes

- Δύο κλάσεις μπορεί να έχουν πολλαπλές συσχετίσεις μεταξύ τους



Multiple Associations
between classes

Οι πολλαπλές συσχετίσεις μεταξύ δύο κλάσεων **μας βοηθούν να καταγράψουμε διαφορετικούς ρόλους** ή **τρόπους αλληλεπίδρασης** μεταξύ των ίδιων αντικειμένων. Και **έχουν αντίκτυπο στην υλοποίηση**!

1. Αποσαφηνίζουν τη σημασία της σχέσης

Στη UML, δεν λέμε απλώς "αυτές οι δύο κλάσεις σχετίζονται" — **λέμε ΠΩΣ σχετίζονται**.
Η κλάση Flight σχετίζεται με την Airport **δύο φορές**:

- flies-from: από πού ξεκινά η πτήση
- flies-to: πού προσγειώνεται η πτήση

➡️ Δύο διαφορετικές σημασιολογίες = δύο συσχετίσεις.

2. Καθορίζουν διαφορετικά πεδία (attributes) στην υλοποίηση

Σε γλώσσα όπως η Java, αυτό οδηγεί σε διαφορετικά attributes μέσα στην ίδια κλάση:

```
class Flight {
Airport departureAirport;// flies-from
Airport arrivalAirport; // flies-to
}
```

Αν είχαμε μόνο μία σύνδεση με το Airport, θα ήταν ασαφές αν είναι για αναχώρηση ή άφιξη!

3. Βοηθούν στον σωστό σχεδιασμό λειτουργιών

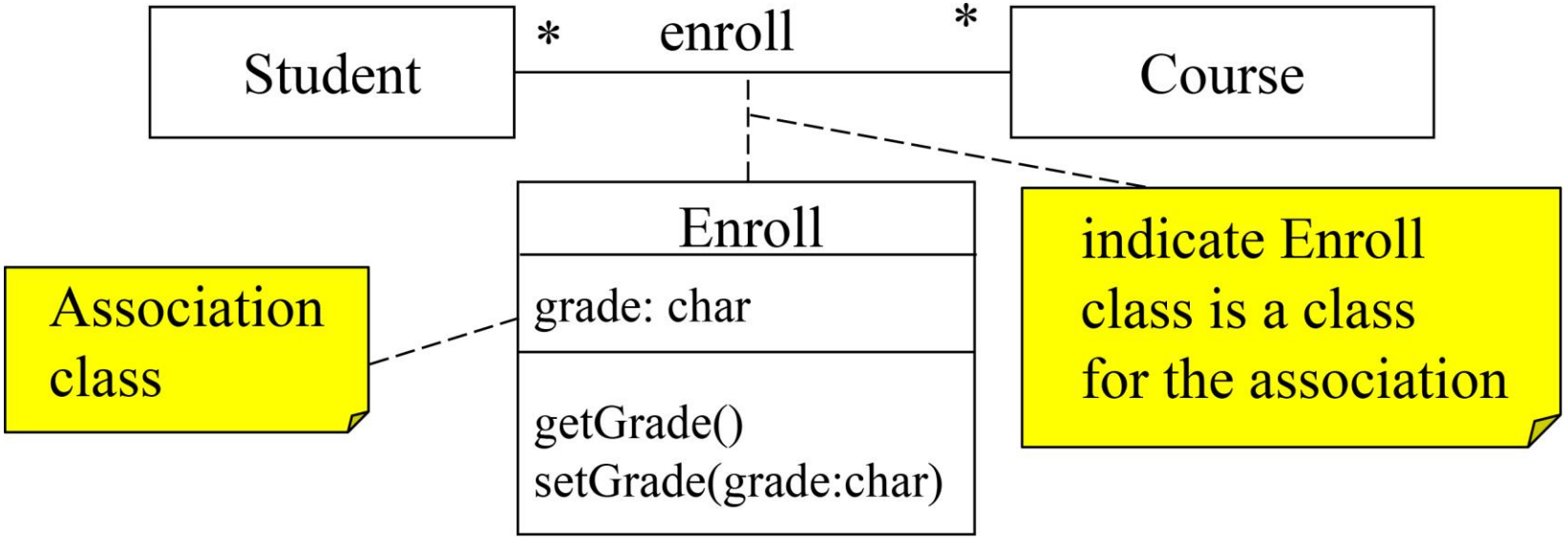
Αν έχεις πολλές συσχετίσεις, μπορείς να φτιάξεις διαφορετικές μεθόδους ανάλογα:

```
Airport
{ getDepartureAirport();
Airport getArrivalAirport(); }
```

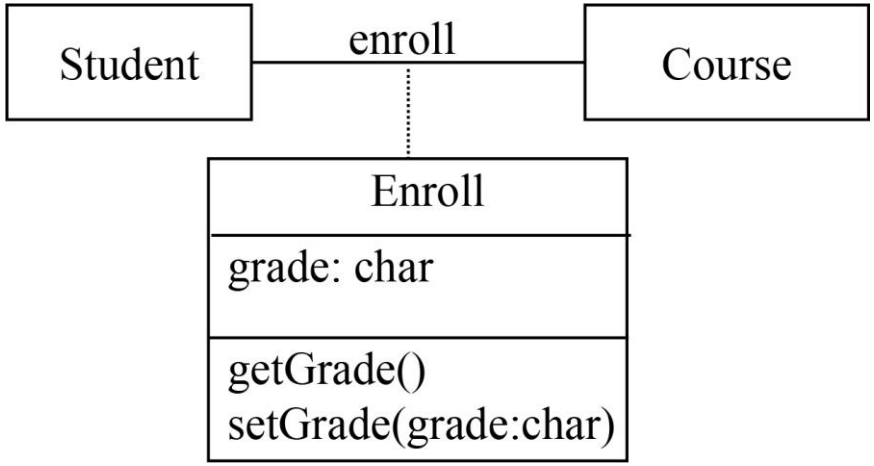
Association Classes
Avoid Many – Many Multiplicity

- Είναι καλή πρακτική να εξαλείφετε τις πολλές προς πολλές συσχετίσεις (many to many associations)
  - περιέχει χαρακτηριστικό(-α) που αφορά δύο συσχετιζόμενες κλάσεις και δεν μπορεί να τοποθετηθεί σε καμία από τις κλάσεις
  - εισάγετε μια κλάση συσχέτισης μεταξύ των δύο κλάσεων
  - η σχέση γίνεται τότε μία προς πολλές και πολλές προς μία (one-to-many and many-to-one)

Exercise : School Information System
conceptual class diagram



Exercise : School Information System
Implementing the Enroll class



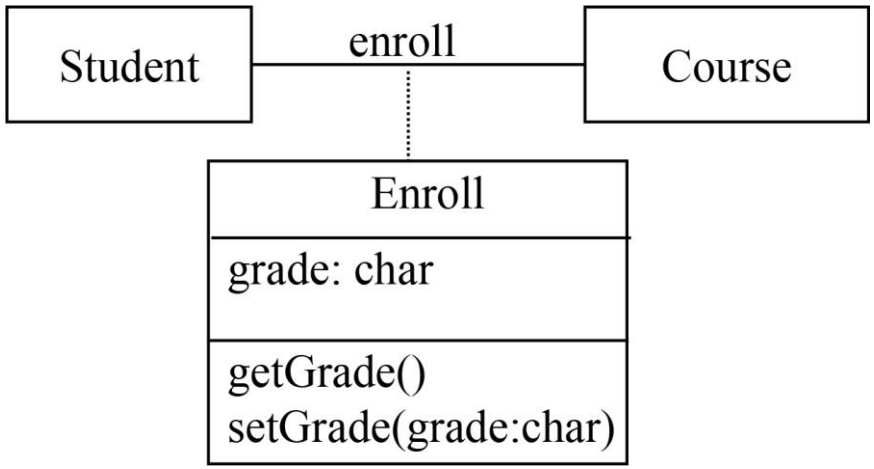

```
Student alex=new Student(...);
Course ai=new Course (...);
...
Enroll e=new Enroll(alex, ai);
...
e.setGrade('A'); // at end of
semester
```

```
public class Student {...}
public class Course {...}

public class Enroll {
 private char grade;
 private Student student; (holds a
reference to student object involved in
enrolment)
 private Course course; (holds a reference
to course object involved in enrolment)

 public Enroll (Student s, Course c) {
  student=s; course=c;
 }
 public char getGrade() {...}
 public void setGrade(char grade)
 {...}
 }
```

Exercise : School Information System
Implementing the Enroll class



OO Realm

Student

sid	name	phone	…	…
001	Alex	…	…	…
002	Eric	…	…	…

Course

cn	title	desc	…	…
c1	oose	…	…	…
c2	AI	…	…	…

Enroll

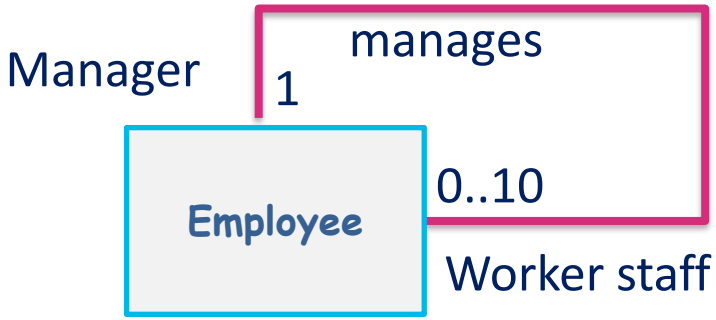
sid	cn	grade	…	…
001	c1	A	…	…
001	c2	A	…	…
002	c1	B	…	…

- RDB Realm

Αντανακλαστική σύνδεση (Reflexive Association)

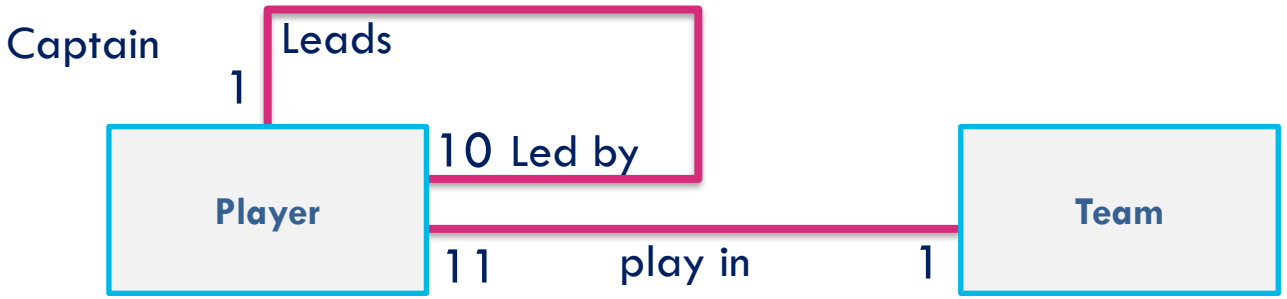
- Μια συσχέτιση που συνδέει μια κλάση με τον εαυτό της
- Παράδειγμα:
  - Ένας διευθυντής είναι υπεύθυνος για έως και 10 υπαλλήλους.
  - Ένας εργαζόμενος διοικείται από 1 διευθυντή.

Οι διευθυντές και οι εργαζόμενοι είναι και οι δύο Κριτήρια για χρήσιμες συσχετίσεις

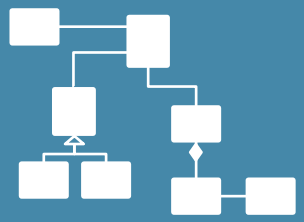


Αντανακλαστική σύνδεση (Reflexive Association)

Παράδειγμα

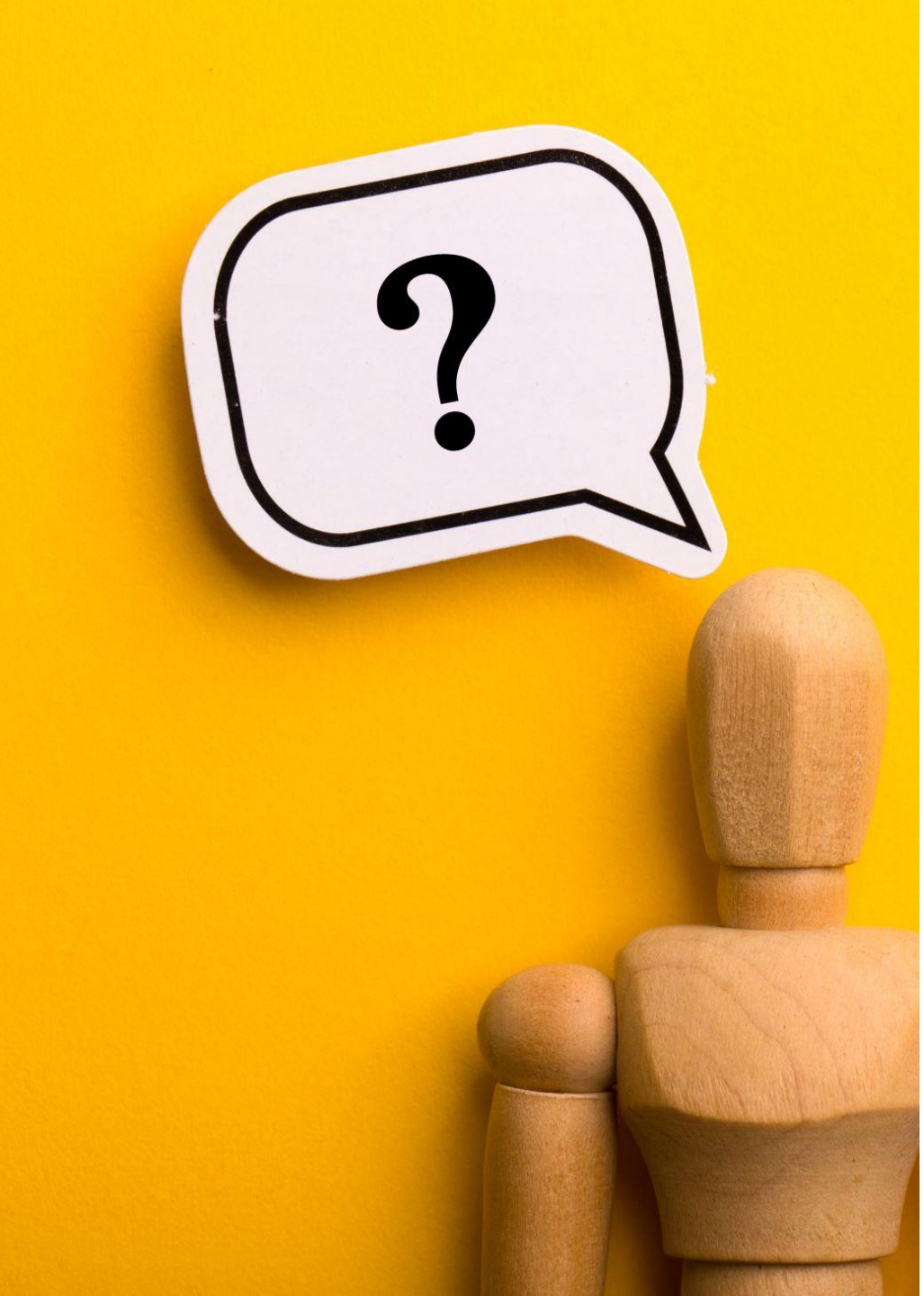


- Μια ομάδα κρίκετ έχει 11 παίκτες.
- Ένας από αυτούς είναι ο αρχηγός.
- Ο αρχηγός ηγείται των μελών της ομάδας.
- Ένας παίκτης μπορεί να παίξει μόνο για μια ομάδα.

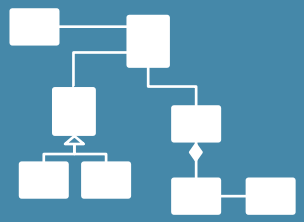


Αντανακλαστική σύνδεση (Reflexive Association)
Παράδειγμα

🔍	**Reflexive Association**	**Κληρονομικότητα (Inheritance)**
➡️ Σχέση	"Ένα αντικείμενο σχετίζεται με άλλο του ίδιου τύπου"	"Μια κλάση κληρονομεί χαρακτηριστικά/συμπεριφορά από άλλη"
💡 Παράδειγμα	Employee διαχειρίζεται άλλους Employee	Manager extends Employee
🔁 Τύπος Σχέσης	**Συσχέτιση (has-a)**	**Κληρονομικότητα (is-a)**
🧠 Λογική	Δείχνει ρόλους/σχέσεις ανάμεσα σε αντικείμενα της ίδιας κλάσης	Δείχνει ιεραρχία τύπων και επαναχρησιμοποίηση κώδικα
📦 Υλοποίηση	Μέσα σε Employee έχουμε Employee manager; και List<Employee> subordinates;	Έχουμε class Manager extends Employee



BREAKOUT SESSION

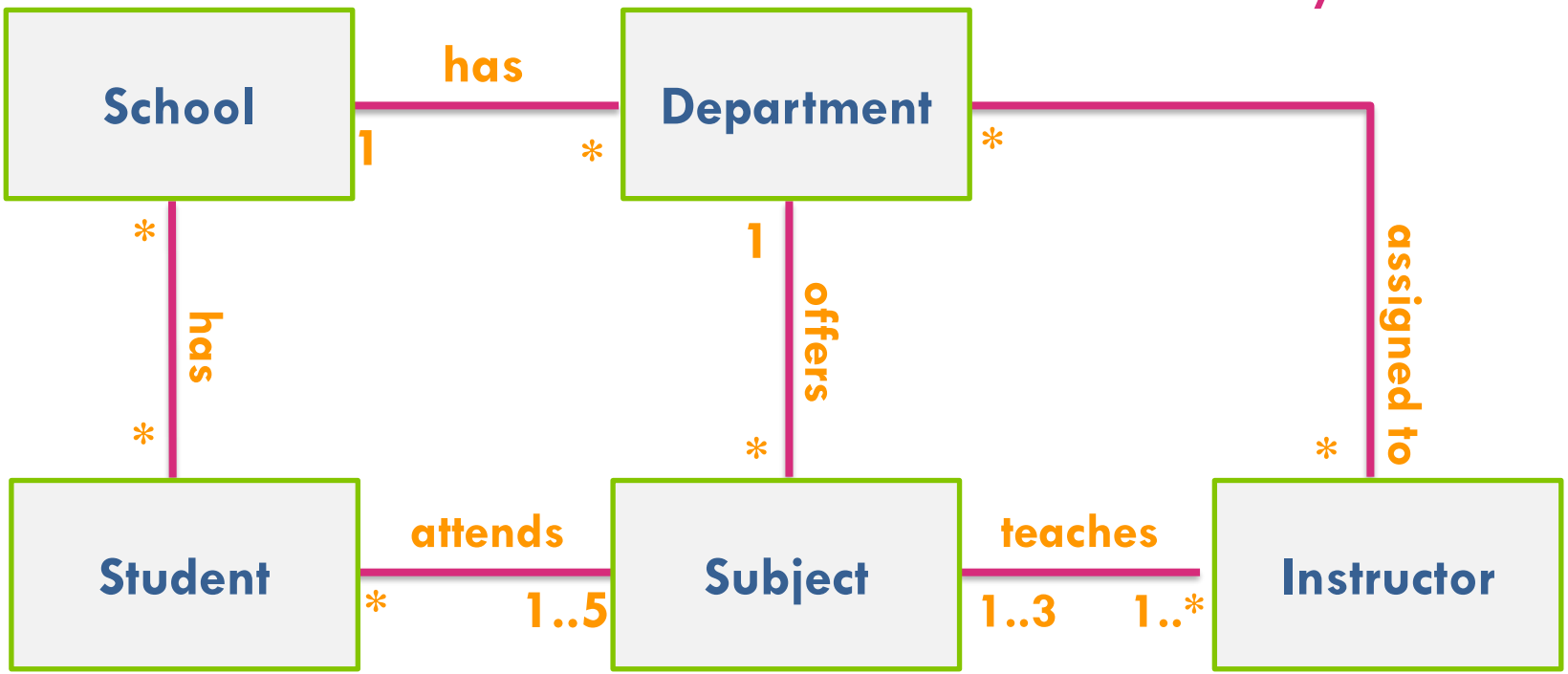


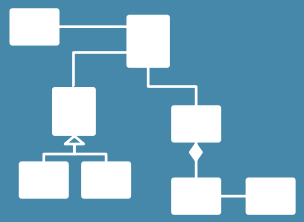
Case Study 1 School Information System
exercise

- A School has one or more departments.
- A department offers one or more subjects. A particular subject will be offered by only one department.
- Departments have instructors and instructors can work for one or more departments.
- Students can enrol in up to 5 subjects in a school.
- Instructors can teach up to 3 subjects.
- The same subject can be taught by different instructors.
- Students can be enrolled in more than one school.

Exercise : School Information System conceptual class diagram

Remember that it's good practice to eliminate many-to-many relationships



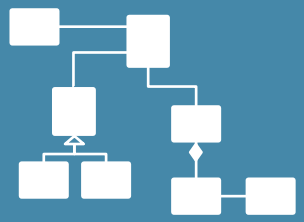


Class Relationships
Σχέση Συνάθροισης - Aggregation

- Οι συναθροίσεις είναι ειδικές συσχετίσεις που αναπαριστούν σχέσεις ***"όλου-μέρους"*** (***'whole-part'*** relations)
  - Όταν αντικείμενα μια κλάσης περιέχουν αντικείμενα κάποιας άλλης κλάσης
  - Το όλο ('whole' side – container) συνήθως ονομάζεται *assembly* or the *aggregate*
  - Το μέρος ('part' side) θεωρείται σαν μέρος του όλου (*is-part-of* the 'whole')
  - ειδική κατηγορία συσχέτισης εξάρτησης - 'has-a' of 'is-part-of' relations
    eg Team *has* Player(s), a Player *is-part-of* a Team

- Η ζωή του μέρους δεν εξαρτάται από την ζωή του όλου (Life-cycle of the 'part' is NOT dependent on the 'whole')
  - ie. To 'part' can exist on its own without the 'whole

- Η συνάθροιση ΔΕΝ συνεπάγεται κληρονομικότητα
  το αντικείμενο Β δεν κληρονομεί συμπεριφορά και χαρακτηριστικά από το αντικείμενο Α



Aggregation
UML Notation

- Αντιπροσωπεύεται από ένα άδειο διαμάντι στην πλεύρα του "όλου". (**hollow diamond** at the side of the 'whole')
  - Το σύμβολο είναι συντομογραφία για τη σχέση 'is-part-of'

  - Η συνάθροιση από το **A** στο **B** σημαίνει ότι το **B** είναι μέρος του **A** (σημασιολογικά) αλλά το **B** μπορεί να μοιραστεί με άλλες κλάσεις και αν το **A** διαγραφεί, το **B** δεν διαγράφεται.

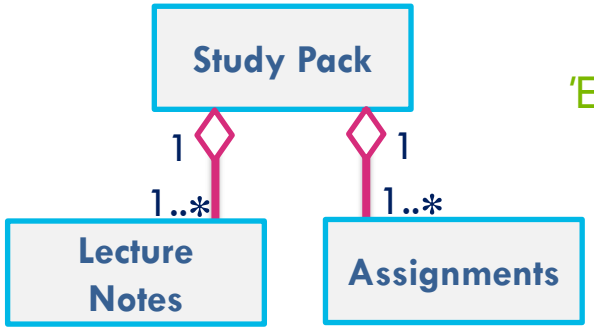
  - Οι σχέσεις συνάθροισης πολλαπλασιάζουν τη συμπεριφορά
    - behaviour applied to the whole is automatically applied to the parts
    - eg. if object A is sent to a customer, then object B is also sent

Aggregation
UML Notation

- Δεν εισάγει **νέους κανόνες** - απλώς προσθέτει **λεξιλόγιο** στη σημειογραφία
  παρέχει σαφήνεια για το σχεδιασμό και την υλοποίηση
- Aggregation in UML means '*has-a*' or '*is-part-of*'

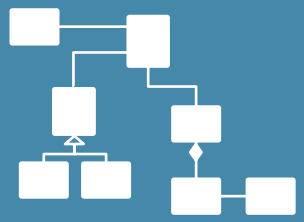
1 module είναι μέρος (is part of) 1 η περισσότερων προγραμμάτων

1 πρόγραμμα αποτελείται από 1…6 modules



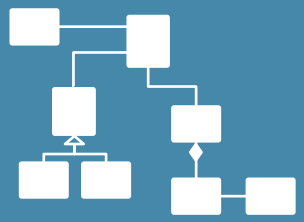
'Ενα study pack περιέχει 1 η περισσότερα lecture slides και 1 η περισσότερα assignments. Ένα (1) lecture note and ένα assignment (1)είναι μέρος ενός(1) study pack

Τα **Lecture notes και assignments μπορούν να υπάρχουν ανεξάρτητα.**



Aggregation vs Association

- Ο σύνδεσμος συσχέτισης (association link) μπορεί να αντικαταστήσει το σύνδεσμο συνάθροισης (aggregation link)
- Γιατί ασχολούμαστε με τις συναθροίσεις;
  - Ο Martin Fowler (συγγραφέας της UML Distilled) προτείνει ότι ο σύνδεσμος συνάθροισης θα μπορούσε να αποφευχθεί, επειδή δεν έχει καμία προστιθέμενη αξία και διαταράσσει τη συνοχή.
  - Θα μπορούσαμε να επιτύχουμε το ίδιο αποτέλεσμα χρησιμοποιώντας **σημασμένες συσχετίσεις (labelled associations)**
  - Κάποιοι λένε ότι η χρήση του συμβόλου συνάθροισης but κάνει το μοντέλο μας πιο καθαρό και σαφές.



Class Relationships
Σχέση Σύνθεσης - Composition

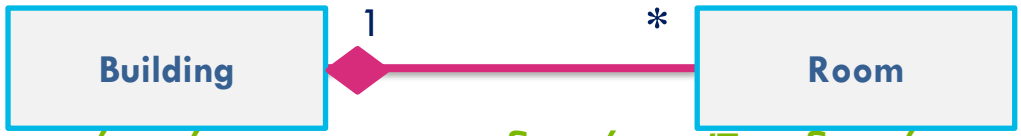
- A 'whole-part' relation
  - πιο ισχυρή μορφή συνάθροισης
  - identified as a **'contains'** relations
- Πιο ισχυρή μορφή συνάθροισης (aggregation) με ισχυρό ownership and dependent lifecycles
- Ο κύκλος ζωής του "μέρους" εξαρτάται από το "όλον (Life-cycle of the 'part' is dependent on the 'whole')
  - Αν το "όλον" καταστραφεί τότε και τα "μέρη" καταστρέφονται
  - το **contained object (μέρος-part) δεν μπορεί να υπάρχει χωρίς το container object (όλον – whole)**
- Πληθικότητα (Multiplicity) από την μεριά του whole πρέπει να είναι zero or one

  ie. the 'part(s)' ανήκουν μόνο σε ΕΝΑ'whole'

Σχέση Σύνθεσης - Composition
UML Notation

- Αντιπροσωπεύεται από ένα γεμάτο διαμάντι στην πλεύρα του "όλου" (**solid diamond** at the side of the 'whole')
  - Το σύμβολο είναι συντομογραφία για τη σχέση'is-part-of' but 'cannot-exist-without' relations

  - Class A (composite class - **whole**) είναι υπεύθυνη για τη δημιουργία και την καταστροφή της κλάσης B (the composed class's object - **Part**) *create/add () and delete() μεθόδοι συμπεριλαμβάνονται στο composite class*



Ένα κτίριο μπορεί να έχει o-to-many δωμάτια. Ένα δωμάτιο ανήκει σε ένα μόνο κτίριο. Αν το κτίριο καταστραφεί, καταστρέφονται και τα δωμάτια. Ένα δωμάτιο δεν μπορεί να υπάρχει από μόνο του ενώ το κτίριο μπορεί (open plan!)

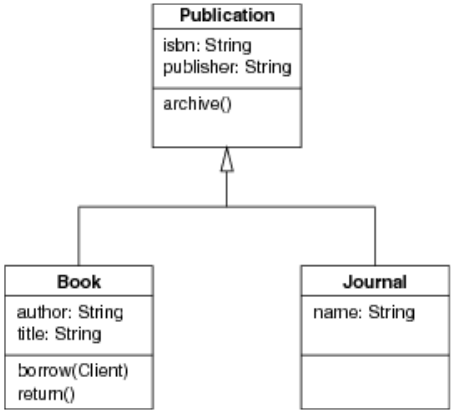
Σχέση Σύνθεσης - Composition
UML Notation

- Εισάγει **νέους κανόνες** καθώς τα αντικείμενα είναι εγγενώς συνδεδεμένα μεταξύ τους
- Composition in UML σημαίνει 'contains' / 'belongs-to'
  eg. Library *contains* Books, Order *contains* PurchaseItems
- Παράδειγμα:

1 printer object contains between 1 and 2 printBuffer objects. Delete the printer and you also delete the printBuffer

Class Relationships
Generalisation / Specialisation (Ειδίκευση/ Γενίκευση)

- **Είναι η ιδέα της κληρονομικότητας (inheritance)**
  - Οι κλάσεις είναι οργανωμένες σε μια ιεραρχία
  - Στην κορυφή της ιεραρχίας οι κλάσεις που απεικονίζουν τα κοινά χαρακτηριστικά όλων των κλάσεων (super-class or parent class)
  - Οι υπο-κλάσεις (sub-classes or children classes) είναι εξειδικευμένες, καθώς περιέχουν χαρακτηριστικά και συμπεριφορές μοναδικές για το αντικείμενο/κλάση, αλλά κληρονομούν επίσης τα χαρακτηριστικά και τις συμπεριφορές της υπερ-κλάσης.
  - Μηχανισμός επαναχρησιμοποίησης τόσο στον σχεδιασμό όσο και στην υλοποίηση του συστήματος

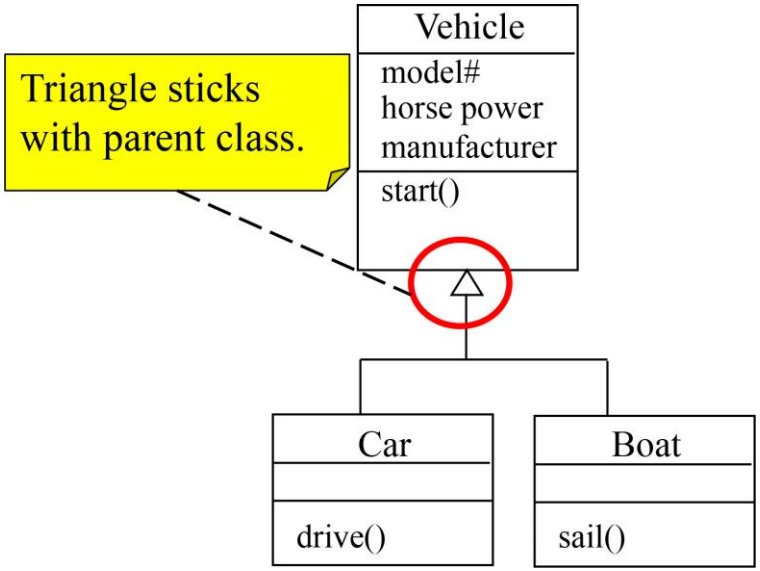


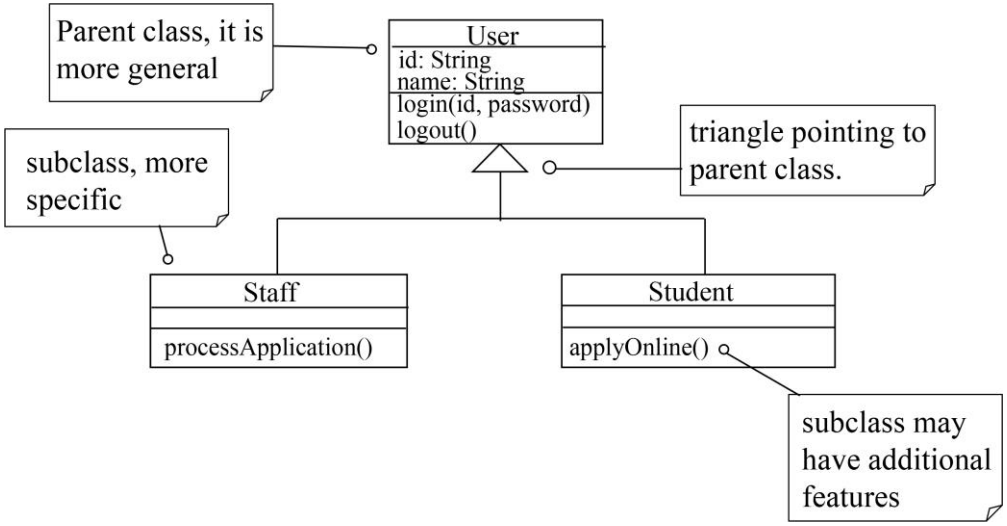
- Προσδιορίζεται ως 'is-a' or 'is-a-kind-of' σχέση

- Κανόνας 100% - Η υπο-κλάση πρέπει να συμμορφώνεται στο 100% των χαρακτηριστικών, της συμπεριφοράς και των συσχετίσεων της υπερ-κλάσης

(Ειδίκευση/ Γενίκευση) Generalisation / Specialisation

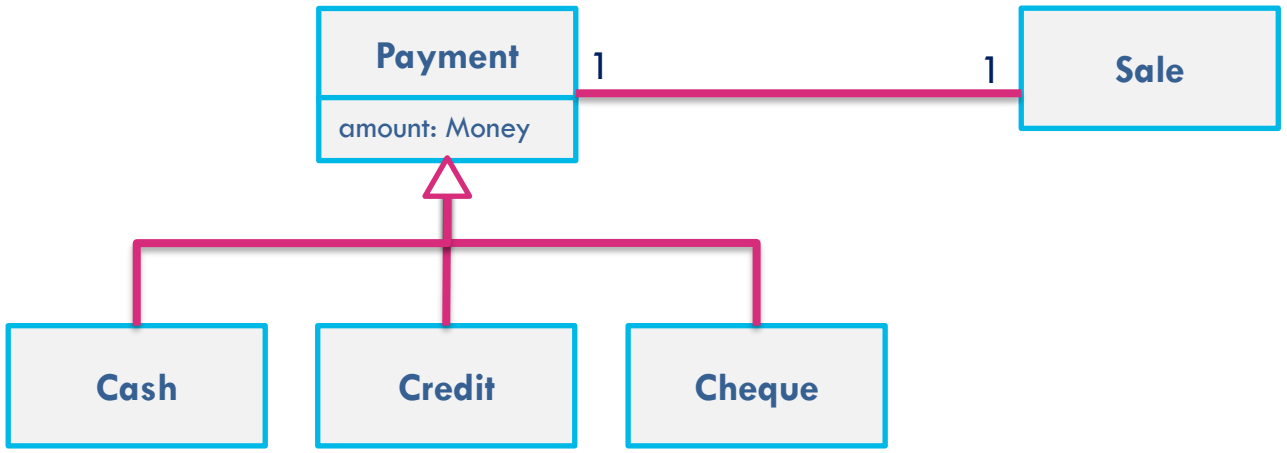
UML Notation



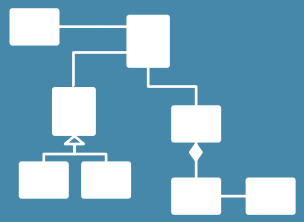


(Ειδίκευση/ Γενίκευση) Generalisation / Specialisation
UML Notation

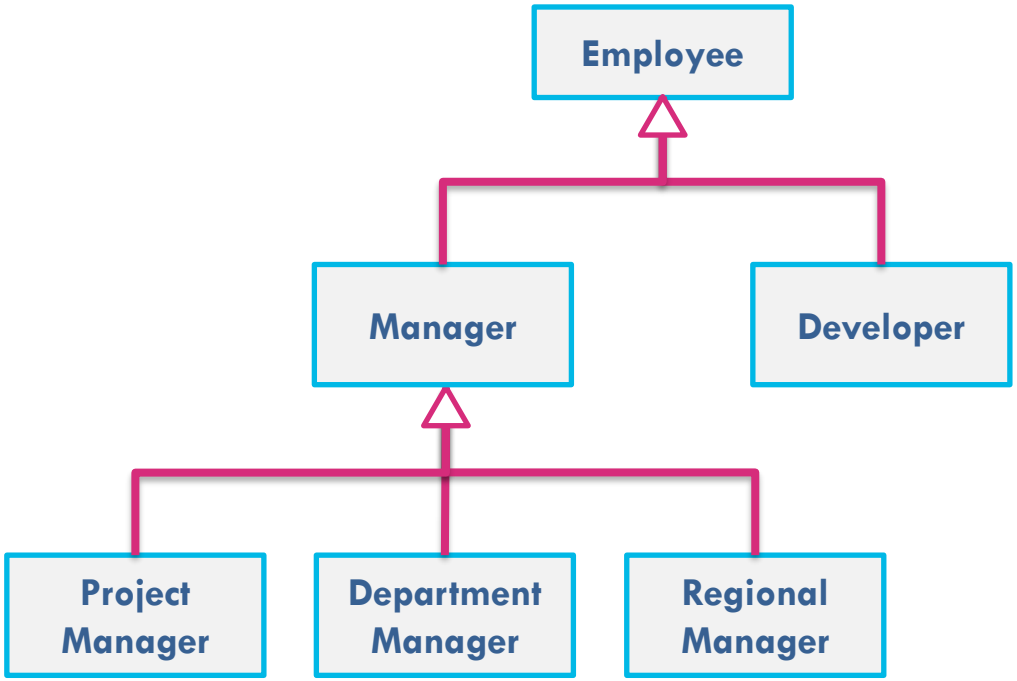
- Αναπαρίσταται ως ένα άδειο τρίγωνο μεταξύ των υπο-κλάσεων και της υπερ-κλάσης. Από τα children στο Parent class



Όλες οι υπο-κλάσεις της κλάσης Payment πρέπει να συμμορφώνονται με την ύπαρξη ενός ποσού (amount) που κληρονομείται από την super class Payment και την πληρωμή μιας Πώλησης (sales).



Generalisation / Specialisation
example

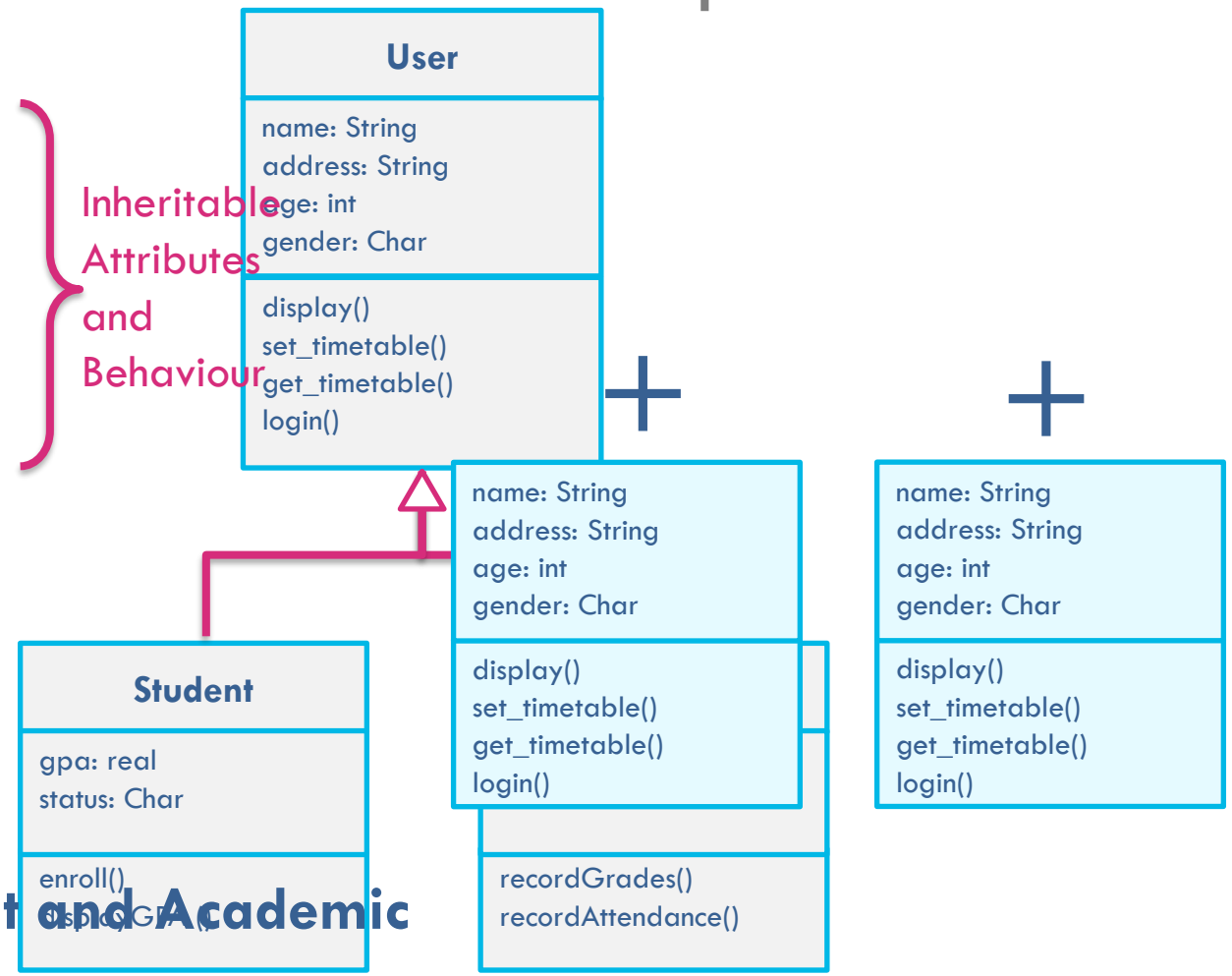


Generalisation

Specialisation

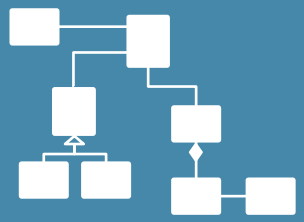
Generalisation / Specialisation
example

Generalisation **Specialisation**

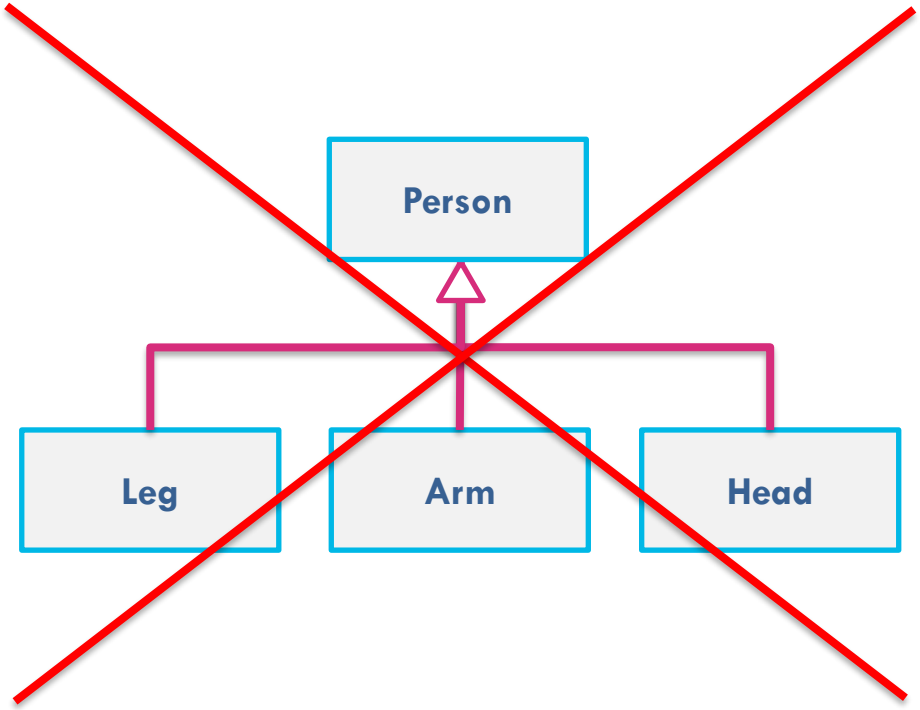


Super-class: User

Sub-class: Student and Academic



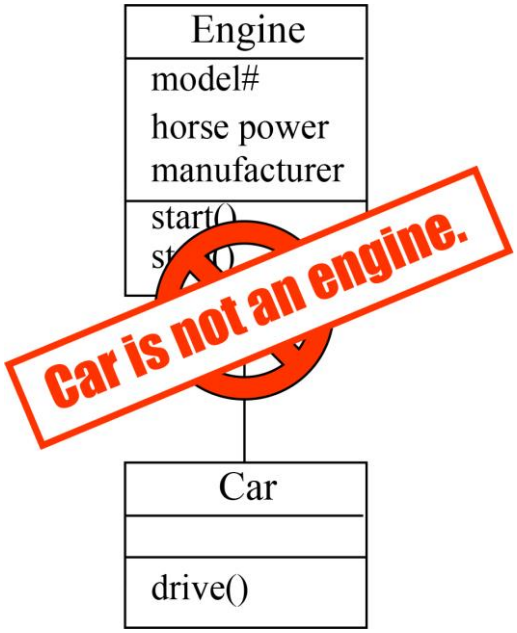
Generalisation / Specialisation
poor example



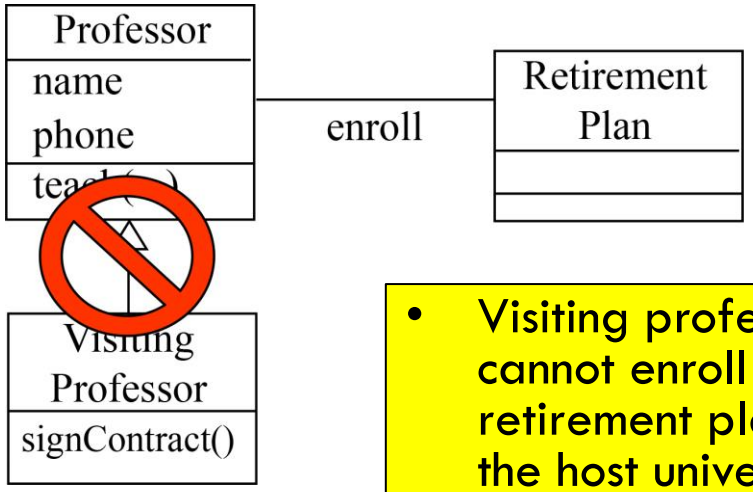
Violates the 'is-a' or 'is-a-kind-of' heuristic

Two Tests for Inheritance

- IS-A test: every instance of a subclass is also an instance of the superclass.



Is every instance of car an instance of engine? No



- Visiting professors cannot enroll in a retirement plan at the host university.

- **Conformance test: relationships of a superclass are also relationships of subclasses.**

Natural Language Helper

identifying objects, classes & relations

- Object? Sub/super-class? Aggregation/Composition?

Is-a ⟷ **Object**

Is-a-kind-of ⟷ **Sub-class**

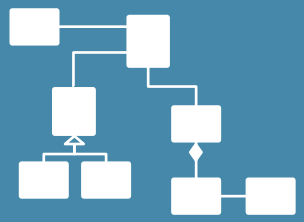
Defines a-kind-of ⟷ **Super-class**

Has-a ⟷ **Association**

but could not exist without ⟷ **Composition**

Life and death coincide (probably Composition)
Created by and within a class (probably Composition)

can exist without ⟷ **Aggregation**

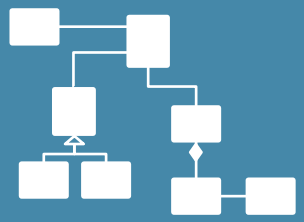


Class Relationships – Best Practice
Σύζευξη - Coupling

- **Coupling** (Σύζευξη): είναι ένα μέτρο της δύναμης της σύνδεσης μεταξύ διαφορετικών κλάσεων εντός του συστήματος
  - In OO, αναφέρεται στο **σύνολο/αριθμό των αλληλεπιδράσεων μεταξύ των αντικειμένων/κλάσεων (amount of interaction between objects)**
  - είναι ένα ποιοτικό μέτρο του πόσες συνδέσεις υπάρχουν σε ένα διάγραμμα κλάσεων μεταξύ των κλάσεων
  - **είναι επιθυμητό να διατηρείται ένα χαμηλό επίπεδο σύζευξης a low level coupling**

    όσο λιγότερες σχέσεις μεταξύ των κλάσεων τόσο καλύτερο το μοντέλο

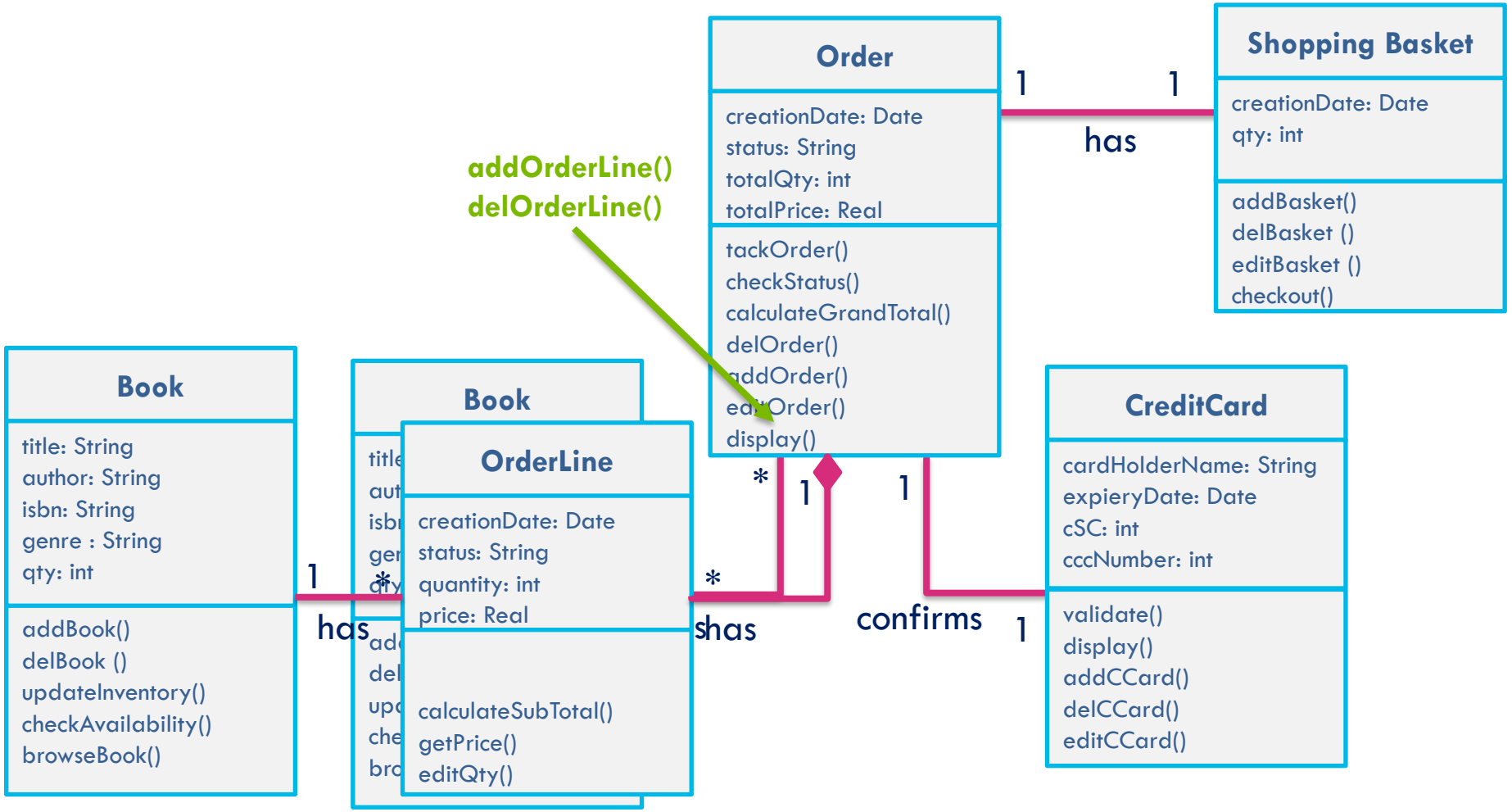
- **E.g customer class έχει συσχέτιση με το order class και το order με το product class, τότε δεν χρειάζεται να συσχετίσουμε και τον customer με το product επίσης**

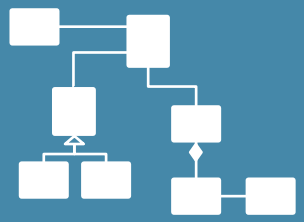


Class Relationships – Best Practice
Συνοχή - Cohesion

- **Cohesion - Συνοχή:** είναι ένα μέτρο της δύναμης της εσωτερικής συνδεσιμότητας μιας κλάσης (the strength of the internal connectivity of an individual class)
  - in OO, το cohesions αναφέρεται στη συνοχή των μεθόδων/operations **μέσα σε μια κλάση**
  - είναι ένα μέτρο του πόσο εστιασμένη είναι μια κλάση σε ένα σκοπό π.χ. η κλάση Customer θα πρέπει να έχει μόνο μεθόδους που ανήκουν στον πελάτη και ΟΧΙ μεθόδους όπως placeOrder() ή makepayment(). Αυτές είναι μεθόδοι άλλων κλάσεων όπως του Order και Payment κλασεων.
  - ***είναι επιθυμητό να διατηρείται υψηλό επίπεδο συνοχής μέσα σε μια κλάση (high level of cohesion within a class)***
    π.χ. για να διασφαλιστεί ότι όλα τα συστατικά μιας κλάσης συμβάλλουν σε μια ενιαία εργασία.

Book Online Store Example





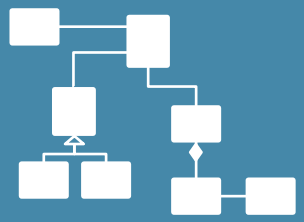
Είναι πραγματικά μια κλάση ή ένα χαρακτηριστικό ή μια μέθοδος;

- Σε ορισμένες περιπτώσεις προκαλεί σύγχυση, επειδή κάτι μπορεί να είναι χαρακτηριστικό σε έναν τομέα, αλλά μπορεί να είναι κλάση σε έναν άλλο, ανάλογα με τις απαιτήσεις.

- **Example:**
  - StartDate είναι χαρακτηριστικό της κλάσης Staff, επειδή επιτρέπει τη διενέργεια κατάλληλων υπολογισμών μισθού, μπόνους και μοριοδότησης.

  - **ΑΛΛΑ,** σκεφτείτε μια υπηρεσία πρόγνωσης καιρού, η οποία τηρεί καθημερινά αρχεία των ατμοσφαιρικών συνθηκών και παράγει αναλύσεις για διάφορες εβδομάδες, μήνες και έτη.

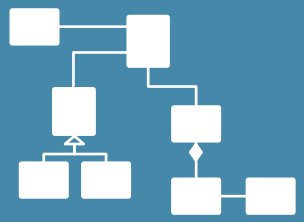
  - Η ημερομηνία (date) σε αυτή την περίπτωση περιγράφεται και από άλλα χαρακτηριστικά όπως average temperature, hours of sunshine etc..
    - Άρα, το Date μπορεί να μοντελοποιηθεί ως κλάση με τα άλλα χαεακτηριστικά ως τα attributes της.



Είναι πραγματικά μια κλάση ή ένα χαρακτηριστικό ή μια μέθοδος;

Γενικός κανόνας:

- Αν ένα χαρακτηριστικό (attribute), μια μέθοδος (method/operation) ή μια συσχέτιση (association) είναι κάτι που πρέπει να περιγράψουμε με περαιτέρω χαρακτηριστικά
- Για παράδειγμα, αν είναι προφανές ότι έχει δικά του χαρακτηριστικά και δική του συμπεριφορά, τότε θα πρέπει να μοντελοποιηθεί ως κλάση.



Conceptual Class Diagram
Build Process

Βήμα 1: Εντοπισμός πιθανών κλάσεων/αντικειμένων

Αναλύστε κάθε use case και την περιγραφή του για να εντοπίσετε ουσιαστικά που αντιστοιχούν σε βασικές επιχειρησιακές οντότητες.

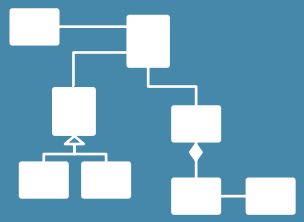
Βήμα 2: Φιλτράρισμα των προτεινόμενων κλάσεων/αντικειμένων

Περιορίστε τη λίστα από το Βήμα 1 αφαιρώντας:

- Συνώνυμες έννοιες
- Ουσιαστικά που βρίσκονται εκτός του πεδίου του συστήματος
- Ρόλους που δεν έχουν μοναδική συμπεριφορά
- Ασαφείς όρους
- Ουσιαστικά που στην πραγματικότητα αντιστοιχούν σε μεθόδους ή ιδιότητες

Βήμα 3: Καθορισμός συσχετίσεων και πολλαπλότητας μεταξύ των κλάσεων

Προσδιορίστε τις σχέσεις μεταξύ των κλάσεων και ορίστε το multiplicity των σχέσεών τους.



Conceptual Class Diagram
Build Process *contd.*

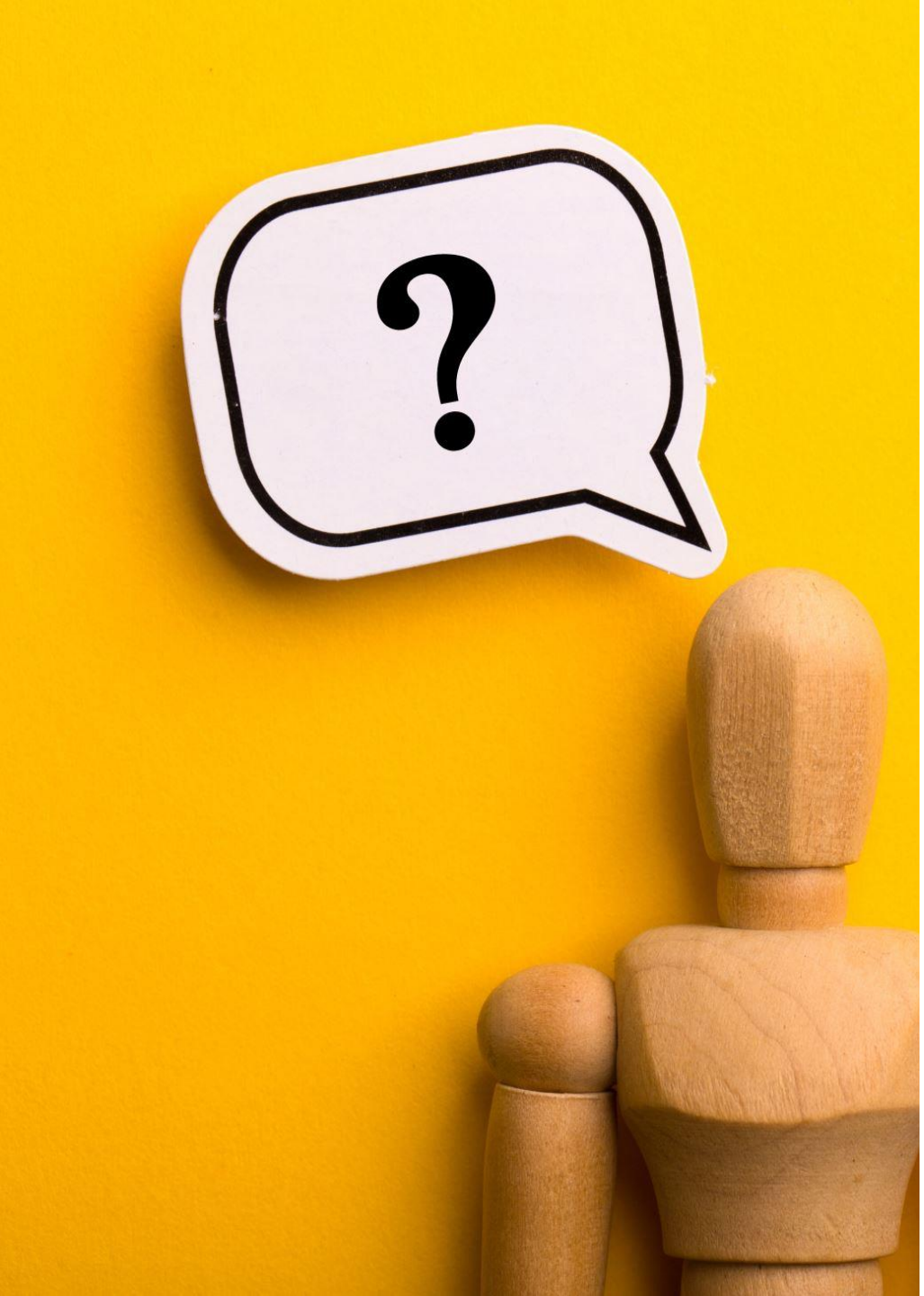
Βήμα 4: Εντοπισμός σχέσεων γενίκευσης/εξειδίκευσης

Αναζητήστε αντικείμενα που έχουν κοινά χαρακτηριστικά και συμπεριφορά.

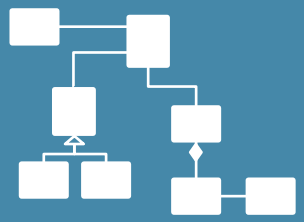
Βήμα 5: Εντοπισμός σχέσεων συγκέντρωσης/σύνθεσης

Αναζητήστε αντικείμενα που μπορούν να αποτελούν μέρος άλλων αντικειμένων.

Βήμα 6: Ολοκλήρωση όλων των στοιχείων σε ένα εννοιολογικό διάγραμμα κλάσεων



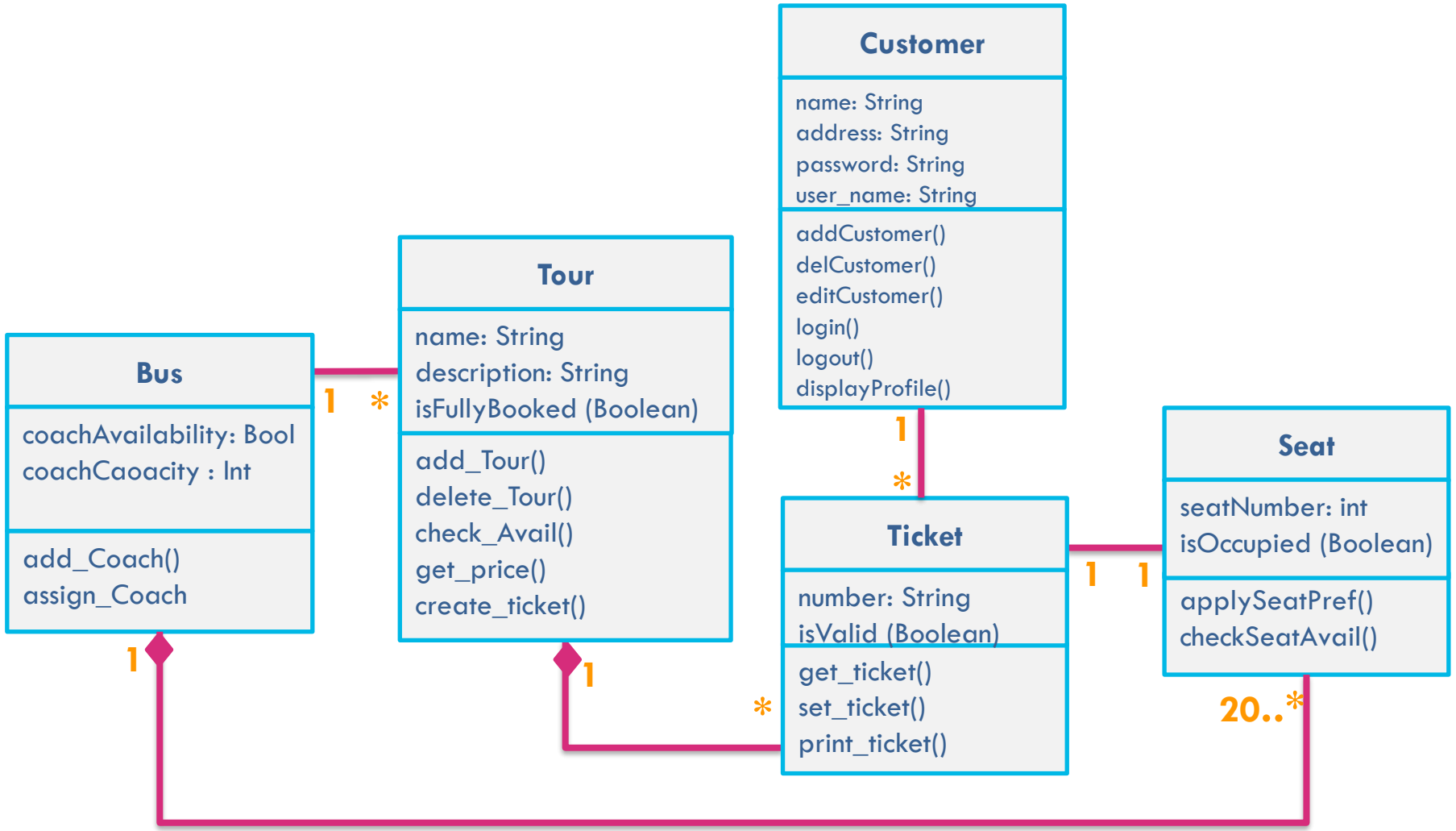
BREAKOUT SESSION



Case Study 2 - Travel Company
exercise

- A travel bus company known for their day tours in central London owns a number of buses.
- The smallest bus has 20 seats.
- Each bus carries out a number of tours.
- When a new tour is scheduled, the company assigns a bus to it and issues one ticket for each seat on the bus.
- Customers buy tickets for a tour. In order to travel, each customer must have a valid ticket.
- Some popular tours are totally booked, but on some less popular ones a bus may travel with some empty seats.

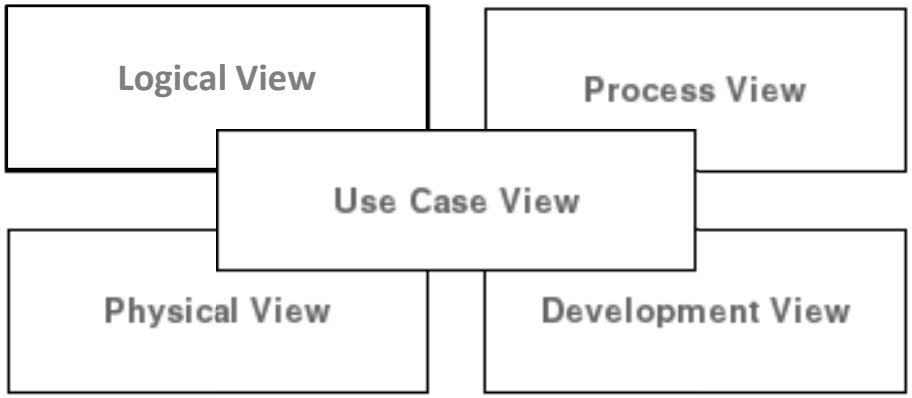
Exercise – Travel Company
conceptual class diagram



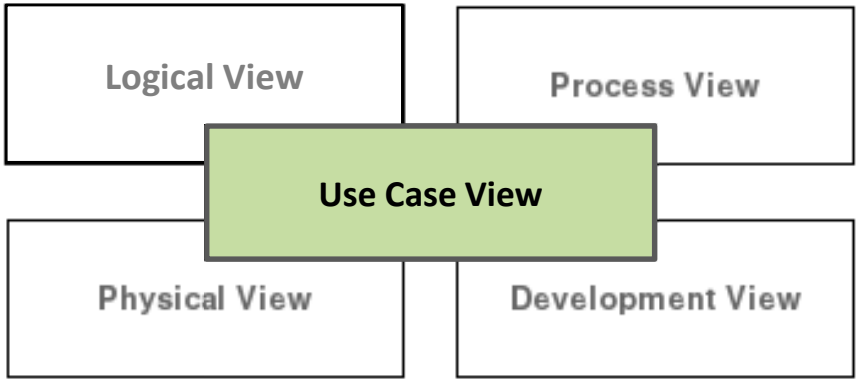
OBJECT SEQUENCE DIAGRAMS

System Analysis
learning check...

Για να μπορέσουμε να κατανοήσουμε ένα σύστημα πρέπει να το σχεδιάσουμε από διαφορετικές όψεις, διαφορετικές οπτικές γωνίες.

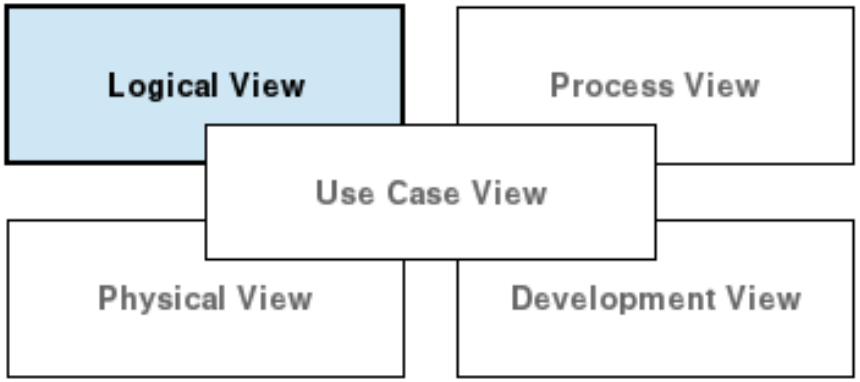


System Analysis
learning check...

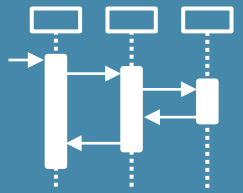


- **Διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams)**
  - Οι λειτουργικές απαιτήσεις (functional requirements) απεικονίζονται ως περιπτώσεις χρήσης (use cases)
  - Περιγραφή του συστήματος με γνώμονα τον χρήστη
  - Απεικονίζει τον τρόπο με τον οποίο οι τελικοί χρήστες αλληλοεπιδρούν με το σύστημα

System Analysis
learning check...



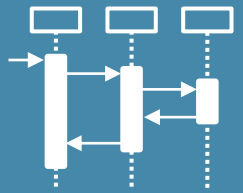
- **Διάγραμμα Κλάσεων Ανάλυσης (Conceptual Class Diagram)**
  - Ένα αφαιρετικό μοντέλο που αποτυπώνει τις σημαντικές κλάσεις και τις σχέσεις τους, όπως αυτές υπάρχουν μέσα στο πρόβλημα που έχουμε να επιλύσουμε.
  - Μια "στατική" εικόνα ('static' picture) των κύριων αντικειμένων/κλάσεων (Objects/classes) στον τομέα μελέτης



Conceptual Class Diagram

Για να ορίσουμε ένα αντικείμενο, πρέπει να ρωτήσουμε:

- Τι πρέπει να γνωρίζω για να εκτελέσω το ρόλο μου; ("What do I need to know to perform my role?")
  - Ποια είναι τα χαρακτηριστικά μου (My Attributes)
- Ποιον πρέπει να γνωρίζω για να εκτελέσω τον ρόλο μου; ("Whom do I need to know to perform my role?")
  - Οι συσχετίσεις (associations) μου με άλλα αντικείμενα
- Τι πρέπει να μπορώ να κάνω για να εκτελέσω το ρόλο μου; ("What do I need to be able to do to perform my role?")
  - Οι μέθοδοι/λειτουργίες (methods/operations) μου.
  - Οι πιο συνηθισμένες είναι οι εξής: Create, delete, set(edit), get(display/view))



HOWEVER…
Conceptual Class diagram

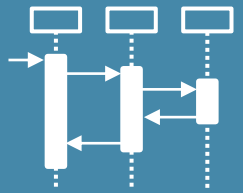
Σε αυτό το στάδιο της ανάλυσης, δεν είναι δυνατόν να προσδιοριστούν:

- Όλες οι πιθανές μέθοδοι/λειτουργίες (methods/operations) μιας κλάσης
- Όλοι οι τύποι κλάσεων.
- Όλες οι συσχετίσεις (associations) μεταξύ κλάσεων.

Πρέπει να καθορίσουμε πώς και με ποια σειρά αλληλεπιδρούν τα αντικείμενα/κλάσεις ΠΡΩΤΑ!

- Να προσδιορίσουμε τις πιθανές αλληλεπιδράσεις μεταξύ των αντικειμένων που εντοπίστηκαν ΠΡΩΤΑ! Και να κατανοήσουμε τα μηνύματα (*messages) τα οποία πρέπει να στείλουν σε άλλα αντικείμενα και τα μηνύματα στα οποία πρέπει να απαντήσουν προκειμένου να εκτελεστεί μια συγκεκριμένη διεργασία του συστήματος.*

Πρέπει να δημιουργήσουμε το Δυναμικό Μοντέλο (dynamic model) του συστήματος



Γιατί χρειαζόμαστε το Dynamic Modelling

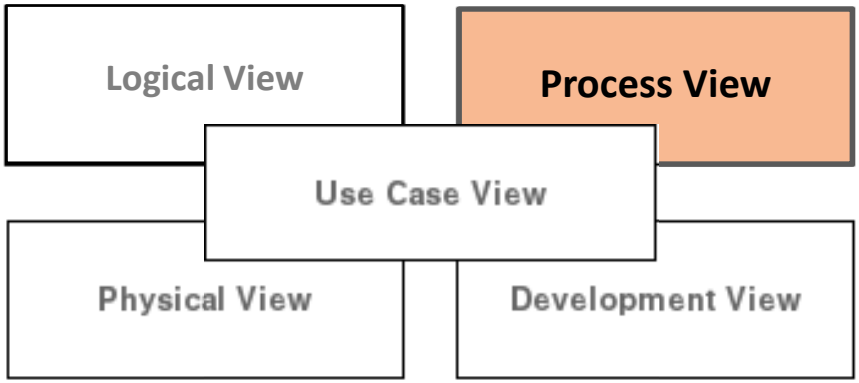
Τι προσπαθούμε να δείξουμε?

- Σε αυτή την φάση γνωρίζουμε για ποιο λόγο θα χρησιμοποιηθεί το σύστημα (**use case modelling and use case diagrams**)

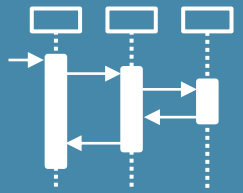
- Έχουμε μοντελοποιήσει μια προκαταρκτική, στατική άποψη του συστήματος **(object modelling and conceptual class diagram**)

- Πρέπει τώρα να εξετάσουμε πώς θα **συμπεριφέρεται το σύστημα** (**dynamic modelling and object sequence diagrams**)
  - πώς θα αλληλεπιδρούν οι κλάσεις μεταξύ τους προκειμένου να επιτευχθεί αυτό που καθορίστηκε στις περιπτώσεις χρήσης (use cases)

System Analysis
learning check...



- **Δυναμικό μοντέλο και διαγράμματα (Dynamic Model and Diagrams)**
  - Αυτό το νέο επίπεδο μοντελοποίησης εξετάζει πώς συμπεριφέρεται το σύστημα
  - Μοντελοποιεί τις αλληλεπιδράσεις μεταξύ του χρήστη και των κλάσεων (classes) του συστήματος και μεταξύ των ίδιων των κλάσεων που υλοποιούν μια συγκεκριμένη περίπτωση χρήσης (use case).
  - **Interaction diagrams, State transition diagrams and Activity diagram**

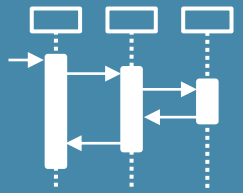


Object Sequence Diagrams (OSDs)
Σκοπός

- Περιγράφουν την αλληλεπίδραση μεταξύ διαφορετικών κλάσεων με την πάροδο του χρόνου
  - η αλληλεπίδραση αυτή πραγματοποιείται μέσω μηνυμάτων που ανταλλάσσονται μεταξύ των κλάσεων
  - τα μηνύματα είναι συχνά το όνομα της μεθόδου που καλείται σε μια κλάση

Emphasis on time ordering
Έμφαση στην χρονική σειρά

- Δείχνουν τη σειρά των μηνυμάτων (messages) που ανταλλάσσονται και συνεπώς τις μεθόδους που καλούνται για κάθε αντικείμενο/κλάση
- Δείχνουν χρονική ακολουθία- κάτι που δεν είναι εύκολο να απεικονιστεί σε άλλα διαγράμματα

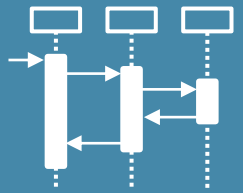


Object Sequence Diagrams (OSDs)
σκοπός

- Χρησιμοποιείται κατά τη διάρκεια της ανάλυσης και του σχεδιασμού για να επεξηγήσει το πως:
  - Αλληλεπιδρούν οι κλάσεις μεταξύ τους (How classes interact)
  - Ποιες μέθοδοι ενεργοποιούνται (What methods are initiated)
  - Την ακολουθία μεταξύ αυτής της αλληλεπίδρασης και των μηνυμάτων που στέλνονται από την μια κλάση στην άλλη (Sequence of activities)

- **Τα διαγράμματα ακολουθίας αντικειμένων (Object sequence diagrams) συνδέουν το διάγραμμα κλάσεων (class diagram) και τα διαγράμματα περιπτώσεων χρήσης (use case diagrams)**

  - Προκειμένου να περιγραφεί ο τρόπος με τον οποίο επιτυγχάνεται μια περίπτωση χρήσης (ένα use case) μέσω της αλληλεπίδρασης των αντικειμένων εντός του συστήματος.



Constructing OSDs
Από που αρχίζουμε?

Analysis Stage:
Requirements gathering (functional requirements)

Start with — Iterative / incremental process — **Start with**

Use-Case Diagrams (UCD)

Conceptual Class Diagram

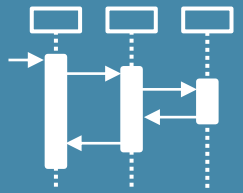
Μας δίνει τα Use-Cases και τα scenarios

Μας δίνει τα Classes ή τα instances of classes (Objects)

Object Sequence Diagrams(OSD)

Με βάση το σενάριο μιας Περίπτωσης Χρήσης (use case scenario), δείχνει τις αλληλεπιδράσεις (μηνύματα που αποστέλλονται και, ως εκ τούτου, τις μεθόδους που ενεργοποιούνται) μεταξύ αντικειμένων με την πάροδο του χρόνου).

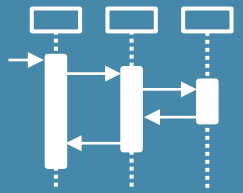
Ένα ή περισσότερα OSD για κάθε περίπτωση χρήσης (use case) στο Use case diagram



Γιατί οι περιπτώσεις χρήσης (use cases) αποτελούν το σημείο εκκίνησης για τα OSDs;

- Κάθε περίπτωση χρήσης (use case) ορίζει την αλληλεπίδραση μεταξύ ενός ή περισσοτέρων actors και του συστήματος, αλλά αντιστοιχεί και σε αλληλεπιδράσεις μεταξύ αντικειμένων/κλάσεων εντός του συστήματος.
  - *Έτσι, οι περιπτώσεις χρήσης μπορούν να χρησιμοποιηθούν ως σημείο έναρξης για τη μοντελοποίηση του τρόπου με τον οποίο τα αντικείμενα αλληλεπιδρούν εντός του συστήματος.*

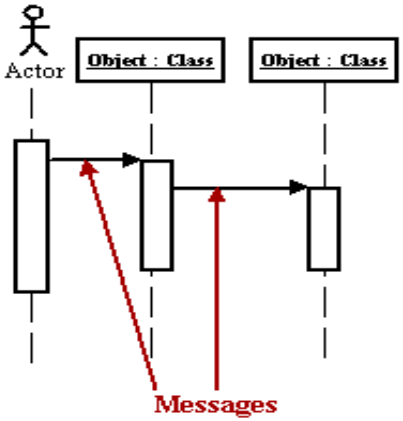
- **Σκεφτείτε ότι ένας χρήστης (actor) πατάει ένα κουμπί για να αναζητήσει προϊόντα (use case).**
  - Αυτό πυροδοτεί μια σειρά αλληλεπιδράσεων εντός του προγράμματος, οι οποίες σε ένα αντικειμενοστραφές σύστημα θα υλοποιηθούν ως συνεργασία αντικειμένων με τη μορφή μεταβίβασης μηνυμάτων μεταξύ τους.
  - Αυτές οι αλληλεπιδράσεις περιγράφονται στην περιγραφή του σεναρίου περίπτωσης χρήσης.
  - Στη συνέχεια, εξετάζουμε το διάγραμμα κλάσεων για να δούμε πώς συσχετίζονται αυτά τα αντικείμενα/κλάσεις



Object Sequence Diagrams (OSD)
key constructs

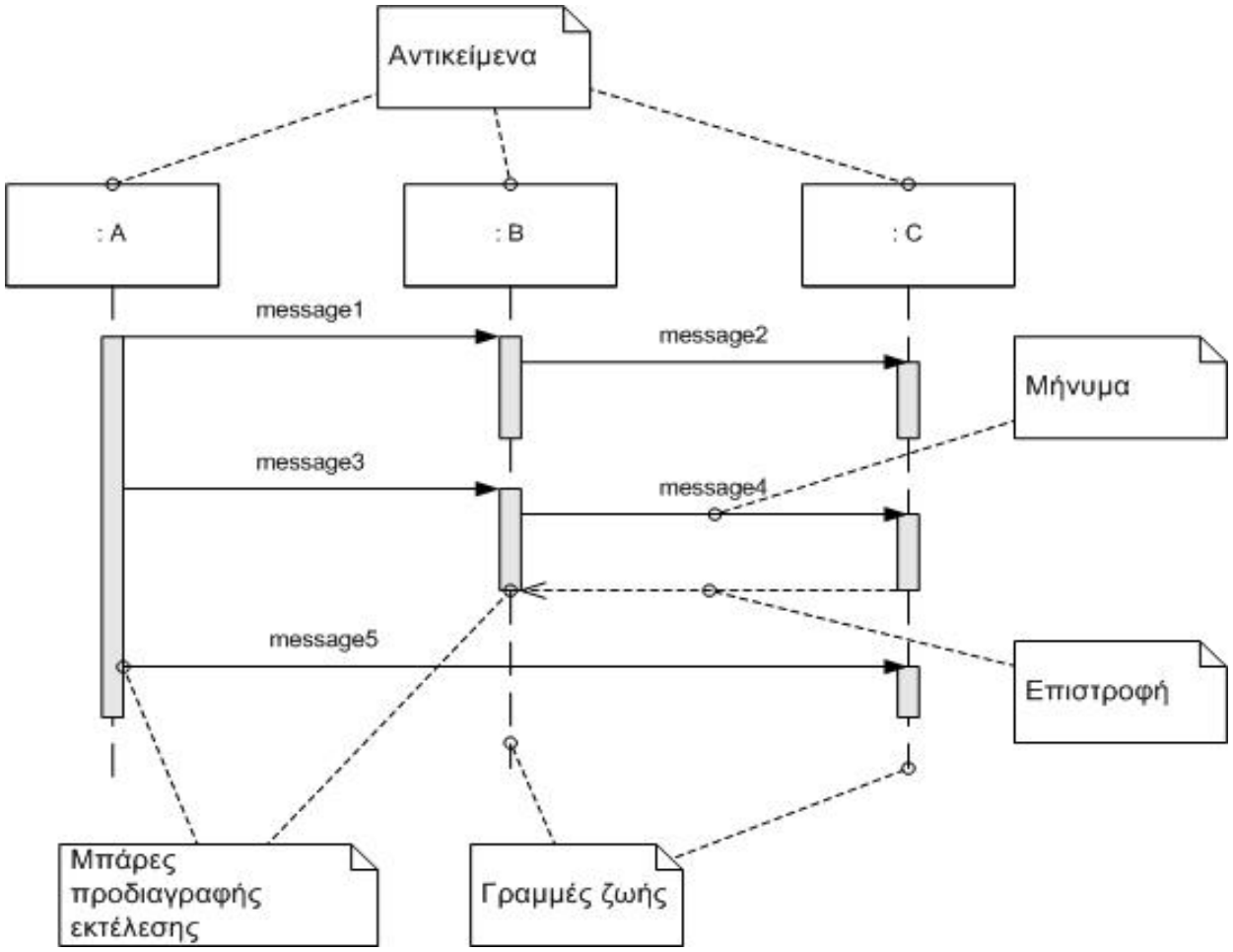
- Actors
  - προσδιορίζονται από το διάγραμμα περίπτωσης χρήσης (Use case diagram)
  - μόνο εκείνοι που ενεργοποιούν (initiate/invoke/trigger) την περίπτωση χρήσης (use case) που περιγράφουμε με ένα OSD. Άρα οι Primary (πρωτεύων) actors.

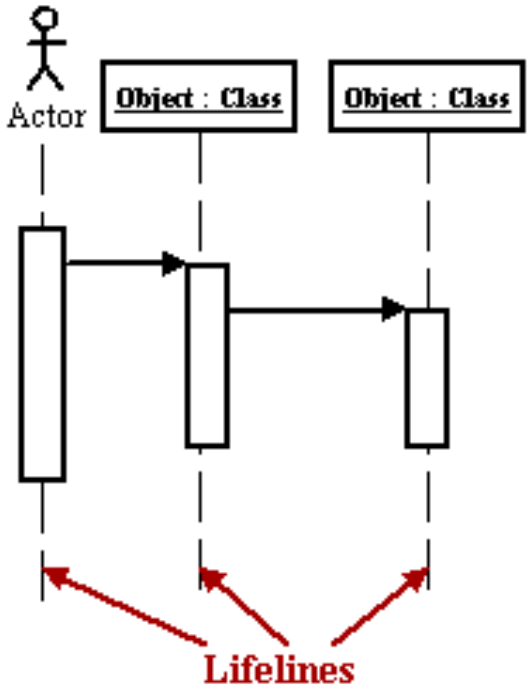
- Συμμετέχοντες - Participants (objects)
  - Αντικείμενα/Κλάσεις που σχετίζονται με την Περίπτωση Χρήσης (use case) και αλληλεπιδρούν στο διάγραμμα ακολουθίας (OSD)
  - Αναγνωρίζονται από το διάγραμμα κλάσεων (Class diagram). Πρέπει να είναι κλάσεις που υπάρχουν στο διάγραμμα κλάσεων.
  - Αν χρειαστεί να χρησιμοποιήσετε άλλες κλάσεις στο OSD τότε θα πρέπει να προσθέσετε αυτές τις καινούριες κλάσεις πίσω στο class diagram.



- Messages
  - Δείχνουν την επικοινωνία μεταξύ των συμμετεχόντων κλάσεων
  - Είναι είτε κλήσεις μεθόδων ή μηνύματα επιστροφής
  - Αναγνωρίζονται από την συμπεριφορά των κλάσεων - τις μεθόδους των κλάσεων

Object Sequence Diagrams (OSD)
key constructs

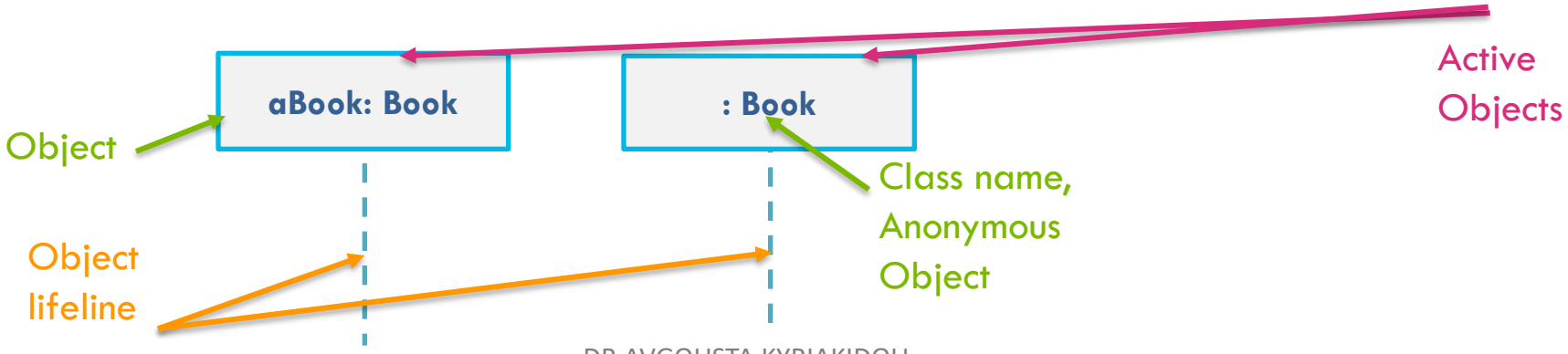




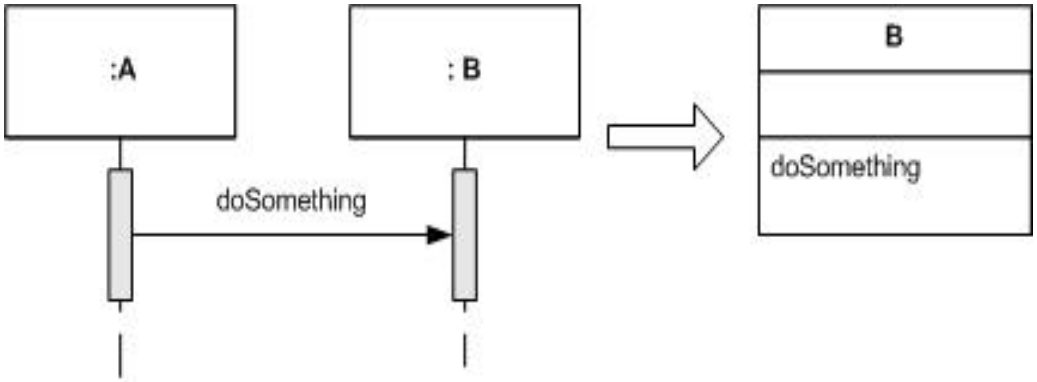
OSD - UML Notation
Objects

- Μοντελοποιούμε κλάσεις και όχι αντικείμενα. Αντιπροσωπεύονται απο:
  - Identifier – κουτιά τα οποία έχουνε μέσα το class name
    - Το όνομα της κλάσης προαιρετικά μπορεί να περιλαμβάνει και το όνομα αντικειμένου

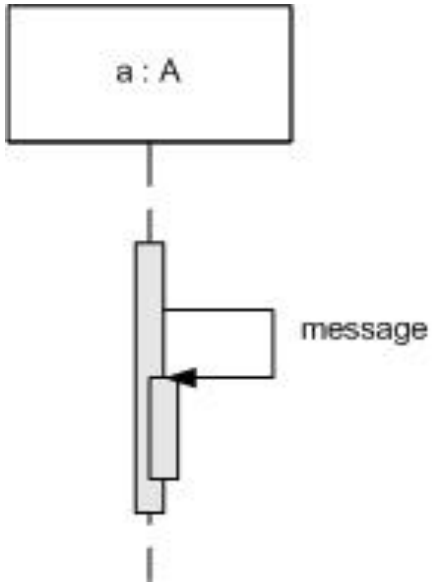
      Name Syntax= <objectName>:<className>
    - Το όνομα του αντικειμένου είναι απαραίτητο μόνο εάν διευκρινίζει καλύτερα το διάγραμμα αλλιώς αποφεύγεται.
  - Γραμμή ζωής - Life-line
    - αναπαριστά τη διάρκεια ζωής των αντικειμένων και σχεδιάζεται ως κατακόρυφη διακεκομμένη γραμμή κάτω από τα αντικείμενα.



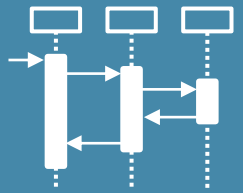
OSD - UML Notation
Messages, *Cntd.*



Η αποστολή του μηνύματος doSomething από την κλαση A στην κλάση B αντιστοιχεί στην «κλήση» της δημόσια (public) λειτουργίας/μεθόδου doSomething() της κλάσης B, όπως έχει προσδιοριστεί στο class diagram.

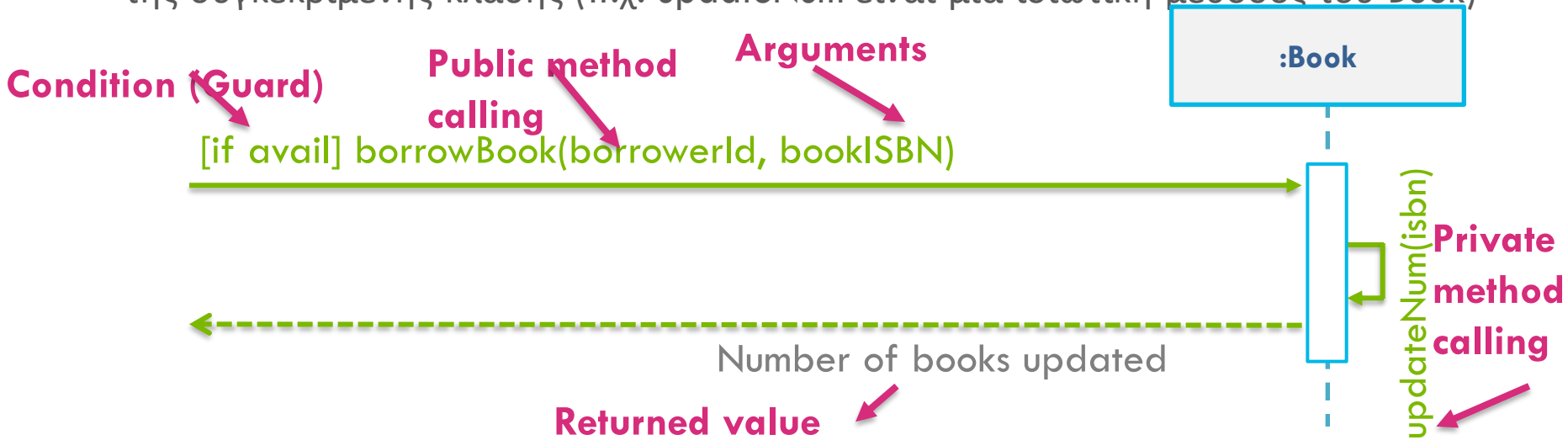


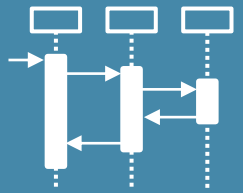
Μήνυμα από την κλάση στον εαυτό της για να ενεργοποιηθεί μια private μέθοδος της κλάσης αυτής. Η κλάση καλεί εσωτερική συμπεριφορά της



OSD - UML Notation
Messages, *Cntd.*

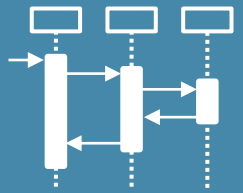
- **Τύποι μηνυμάτων:**
  - Κλήση μεθόδου (Method calling)
  - Μήνυμα επιστροφής (Return message)
- Το μήνυμα στο βέλος υποδηλώνει μια δημόσια μέθοδο (Public method) της κλάσης στο οποίο δείχνει το βέλος (π.χ. borrow book είναι μια δημόσια μέθοδος του Book).
- Ένα μήνυμα που πηγαίνει πίσω στην ίδια την κλάση υποδηλώνει μια ιδιωτική μέθοδο της συγκεκριμένης κλάσης (π.χ. updateNum είναι μια ιδιωτική μέθοδος του Book)





OSD Constructs
Που κοιτάζουμε για να τα βρούμε?

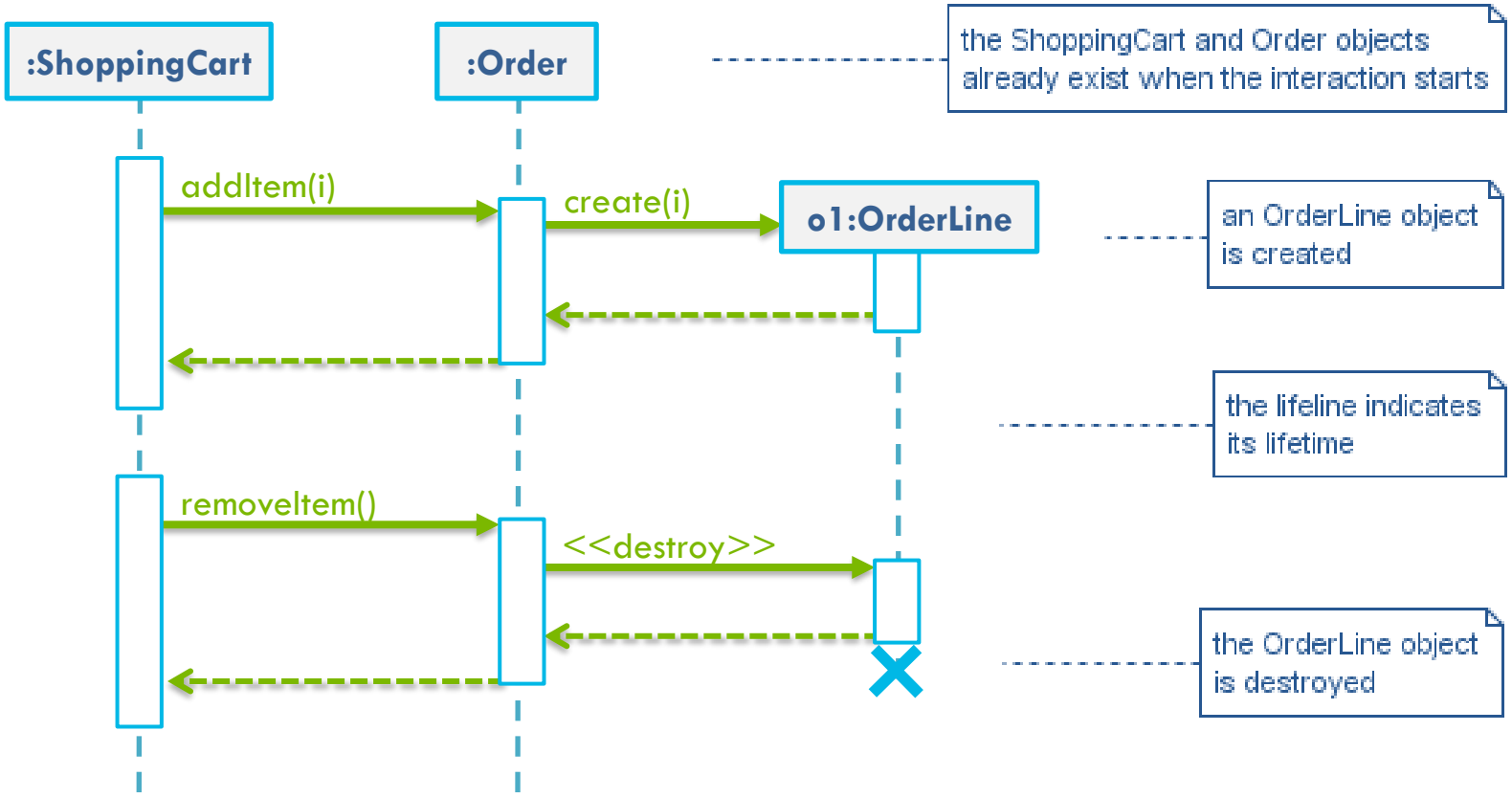
- Το OSD κατασκευάζεται από πληροφορίες που παίρνουμε από το διάγραμμα περιπτώσεων χρήσης και το διάγραμμα κλάσεων
  - Ένα ξεχωριστό OSD για κάθε Use Case στο use case diagram
- Actors (from Use Case diagram)
  - Μόνο οι πρωτεύων – αυτοί που ενεργοποιούν ένα use case
- Participants (objects/classes) (from Conceptual Class Diagram - CCD)
  - πρέπει να είναι κλάσεις που προσδιορίζονται στο CCD
  - Αν μια καινούρια κλάση προσδιοριστεί τοτε πάμε πίσω στο CCD και το αλλάζουμε για να συμπεριλάβουμε την καινούρια κλάση.
    i.e. Έτσι προχωράμε από το διάγραμμα κλάσεων ανάλυσης στο διάγραμμα κλάσεων σχεδιασμού (from Conceptual Class Diagram to Design Class Diagram
- Μηνύματα (κλήσεις μεθόδων/επιστροφή - από τη συμπεριφορά των κλάσεων)
- Ακολουθία γεγονότων (από το use case σενάριο)

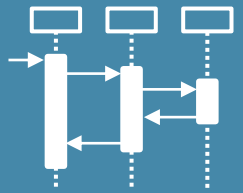


Object Lifetime in OSDs

Δημιουργία και διαγραφή Αντικειμένων
Creation and Destruction

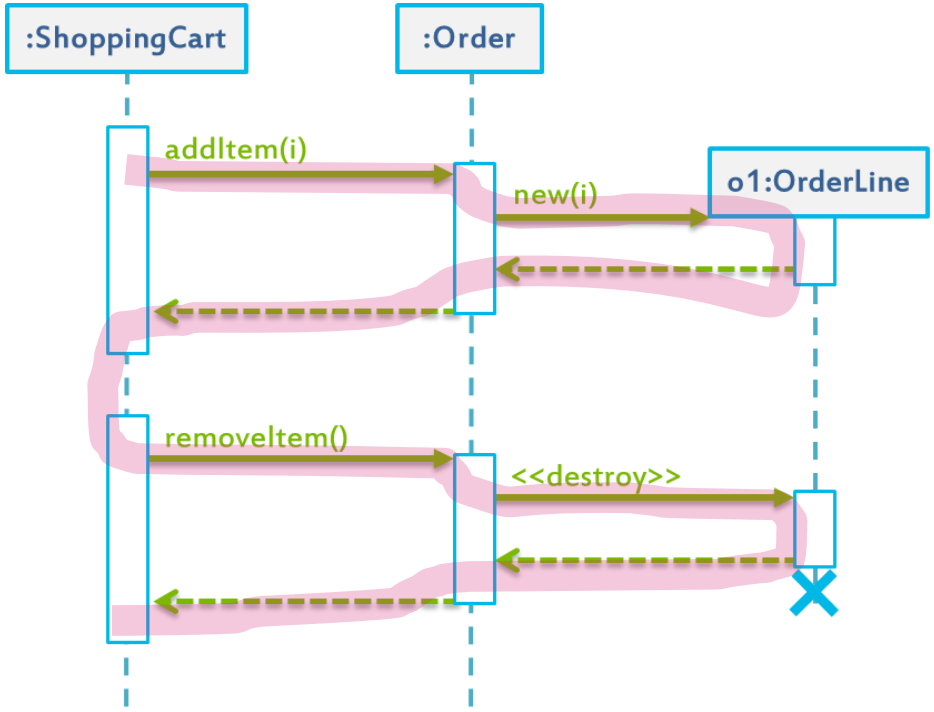
Place order example:

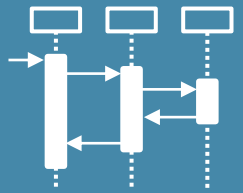




Reading Object Sequence Diagrams
things to note...

- Το διάγραμμα διαβάζεται από πάνω προς τα κάτω
- Ένα μήνυμα μεταξύ δύο κλάσεων συνεπάγεται μια συσχέτιση (association) μεταξύ τους στο διάγραμμα κλάσεων
- Ο ιδιοκτήτης της συμπεριφοράς είναι ο παραλήπτης του μηνύματος.
  message



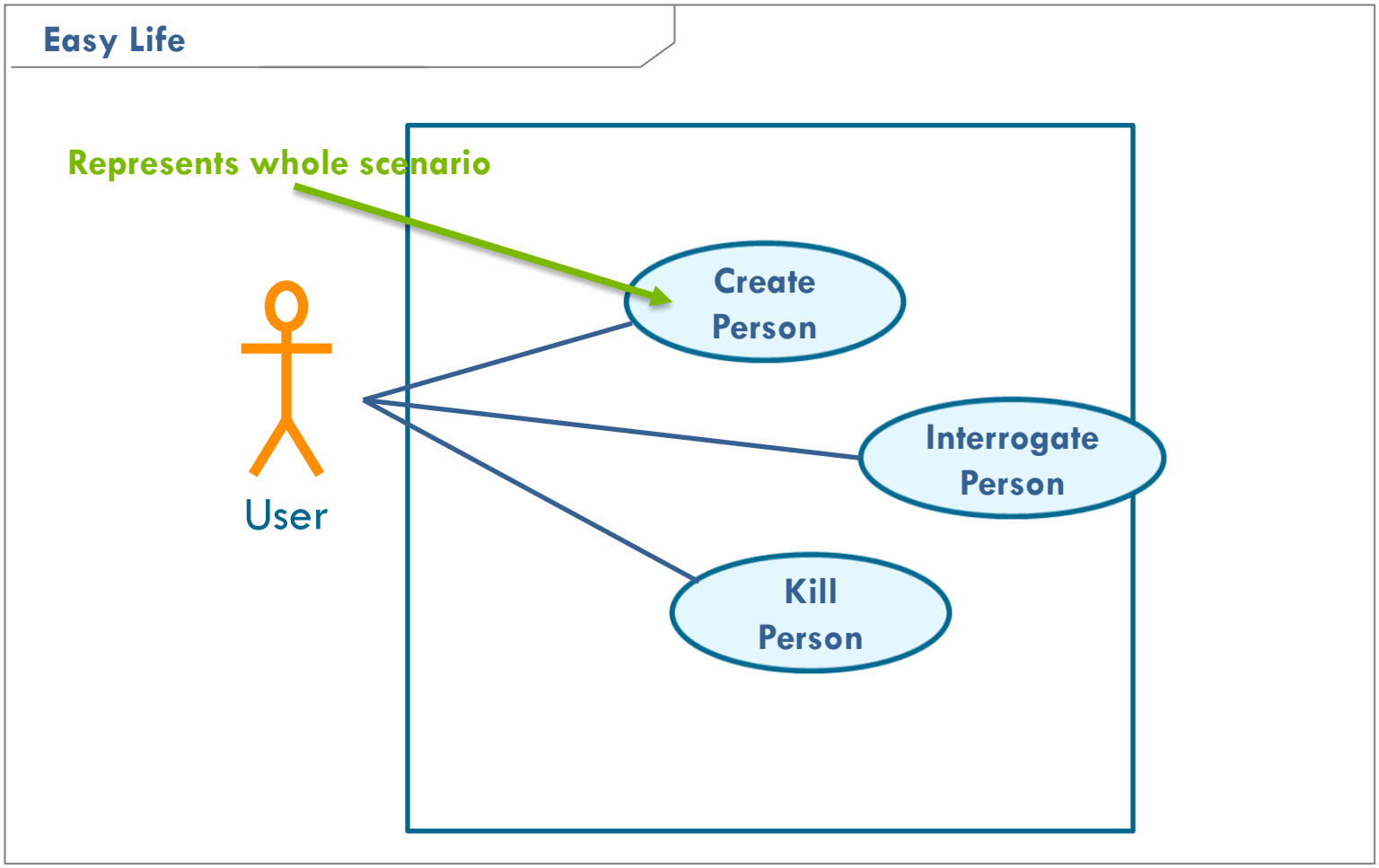


Example 1
An Easy Life

- The Scenario
  - Create a game where personas are created to live a virtual existence in Second Life.
  - It will allow us to **create** a virtual person, and **interrogate** them about their existence (name, age and gender) .
  - The application should allow users to **kill** a person (if their hearts let them ☺).

Tip: **Nouns** help us identify **classes; Verbs** help us to determine **methods** (behaviour)

Example 1- An Easy Life
Use Case Diagram

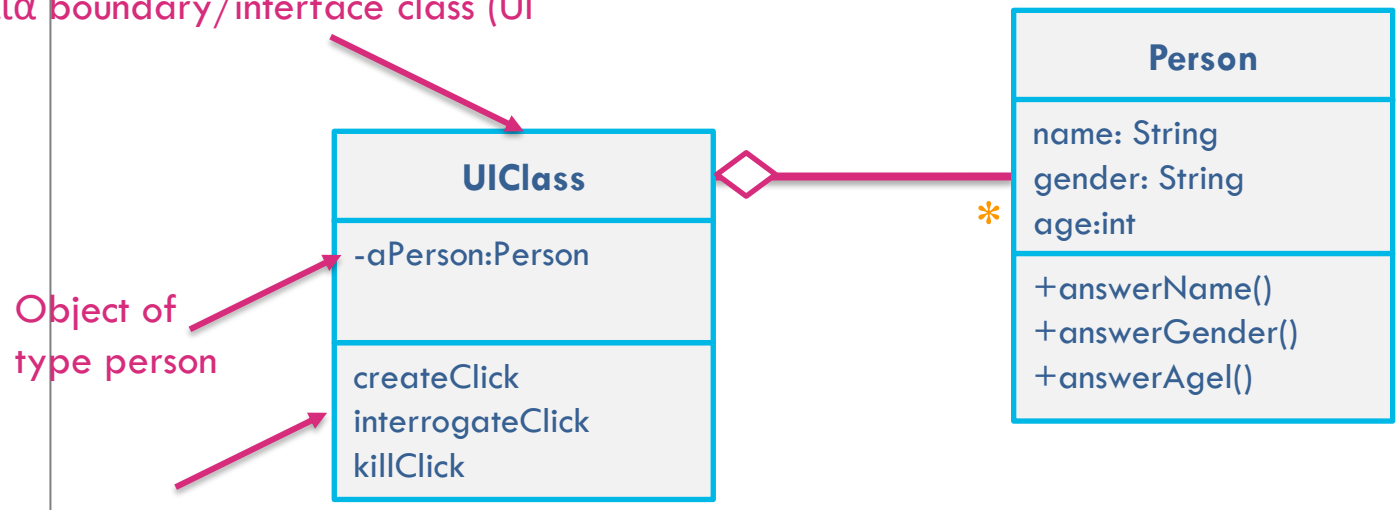


Example 1- An Easy Life
Conceptual Class Diagram

Easy Life

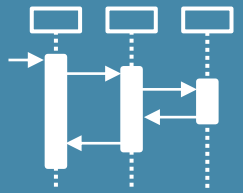
Δεν είναι αυστηρά μέρος του Conceptual Class diagram, αλλά προς το παρόν χρειαζόμαστε μια κλάση για τη δημιουργία αντικειμένων τύπου Person και αυτή η κλάση μπορεί να είναι μια boundary/interface class (UI class)

Οι κλάσεις UI ή boundary classes είναι ένας νέος τύπος κλάσης που αναγνωρίζουμε σε αυτό το σημείο καθώς προχωράμε στο σχεδιασμό.
Ένας user/actor αλληλεπιδρά πάντα με τη διεπαφή χρήστη του συστήματος και όχι απευθείας με τις κλάσεις (entity classes).

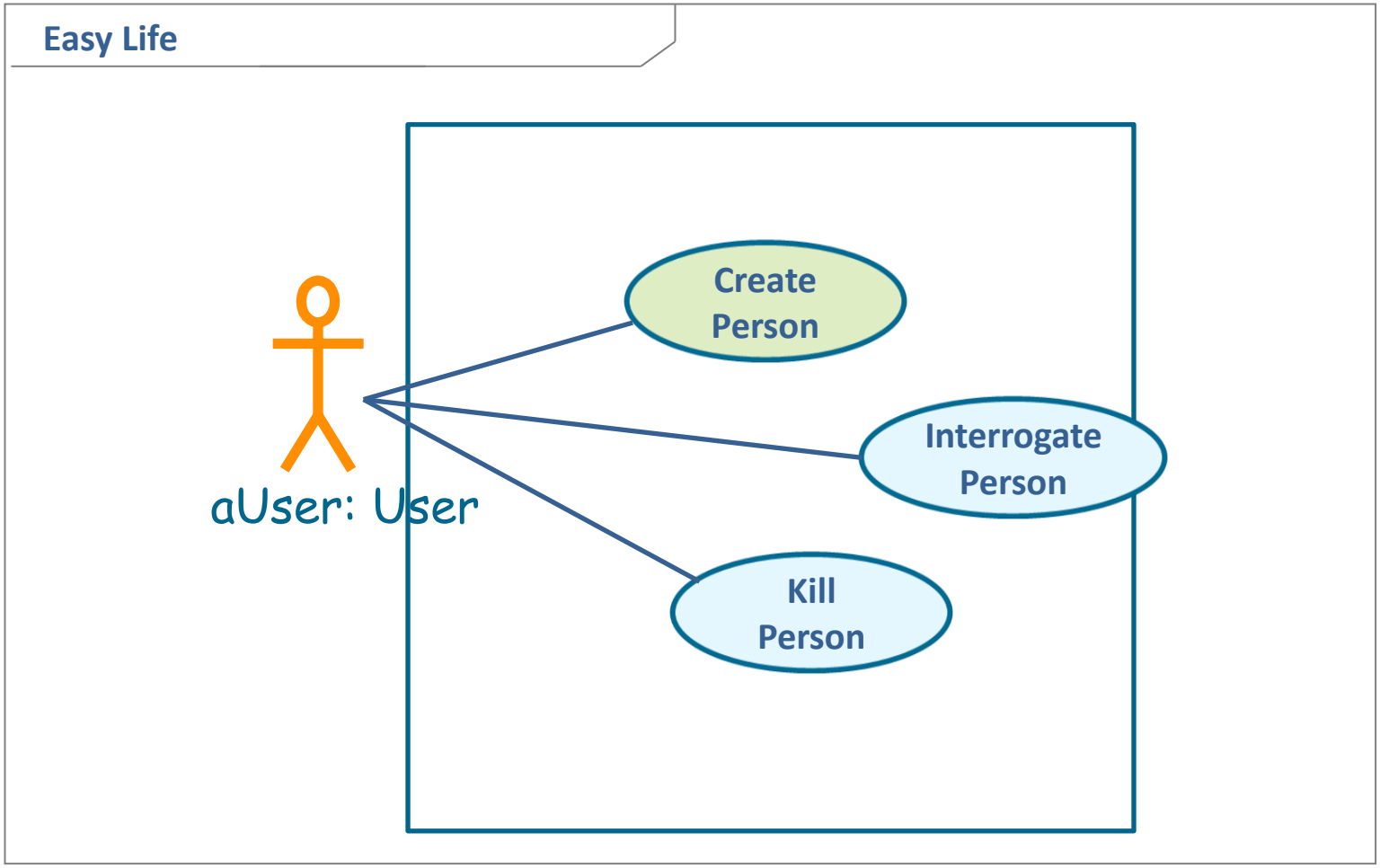


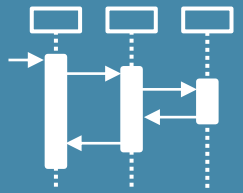
Object of type person

Οι κλάσεις διεπαφής (boundary classes) ανταποκρίνονται σε click methods

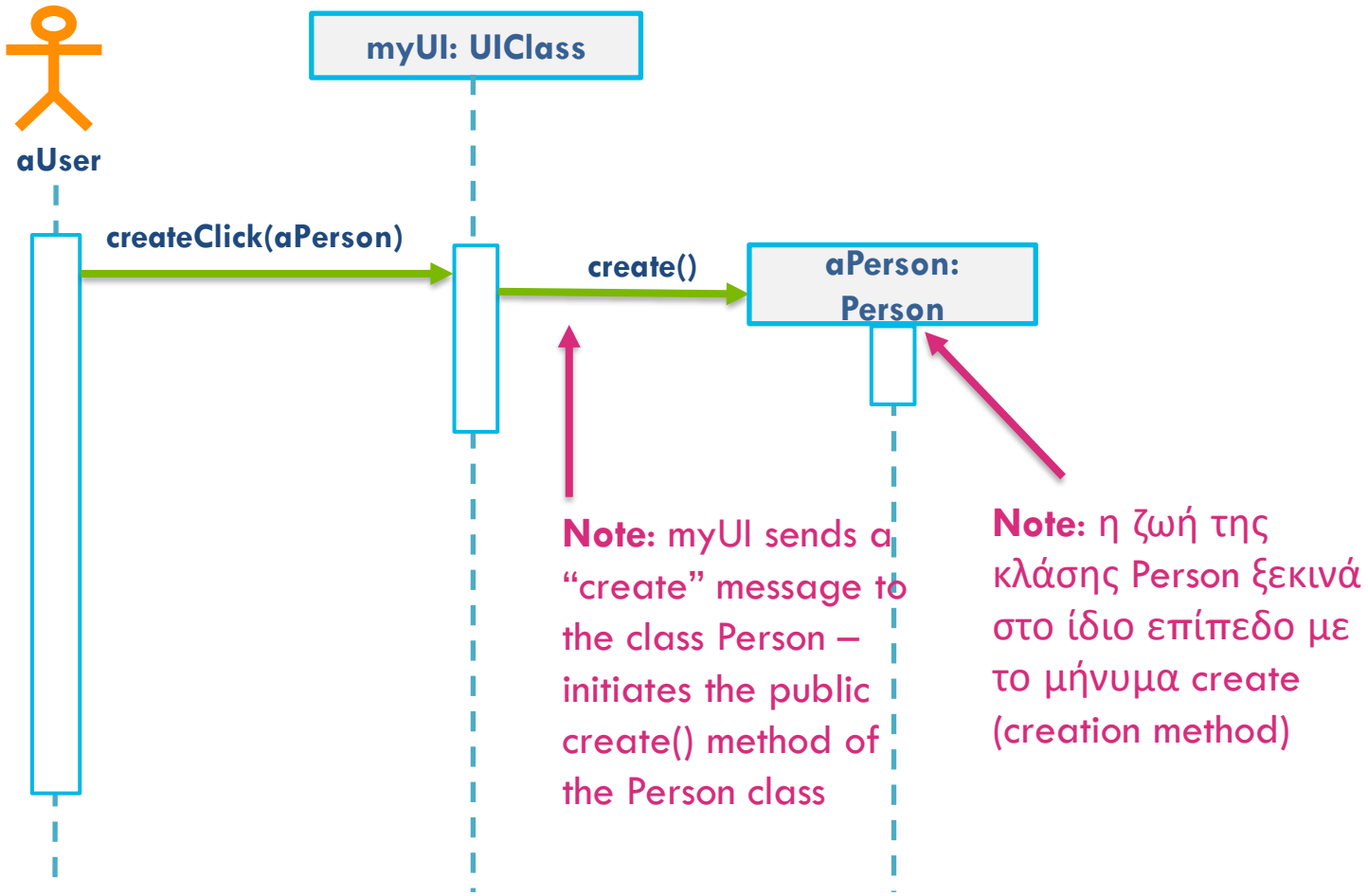


Example 1- An Easy Life
Use Case Diagram

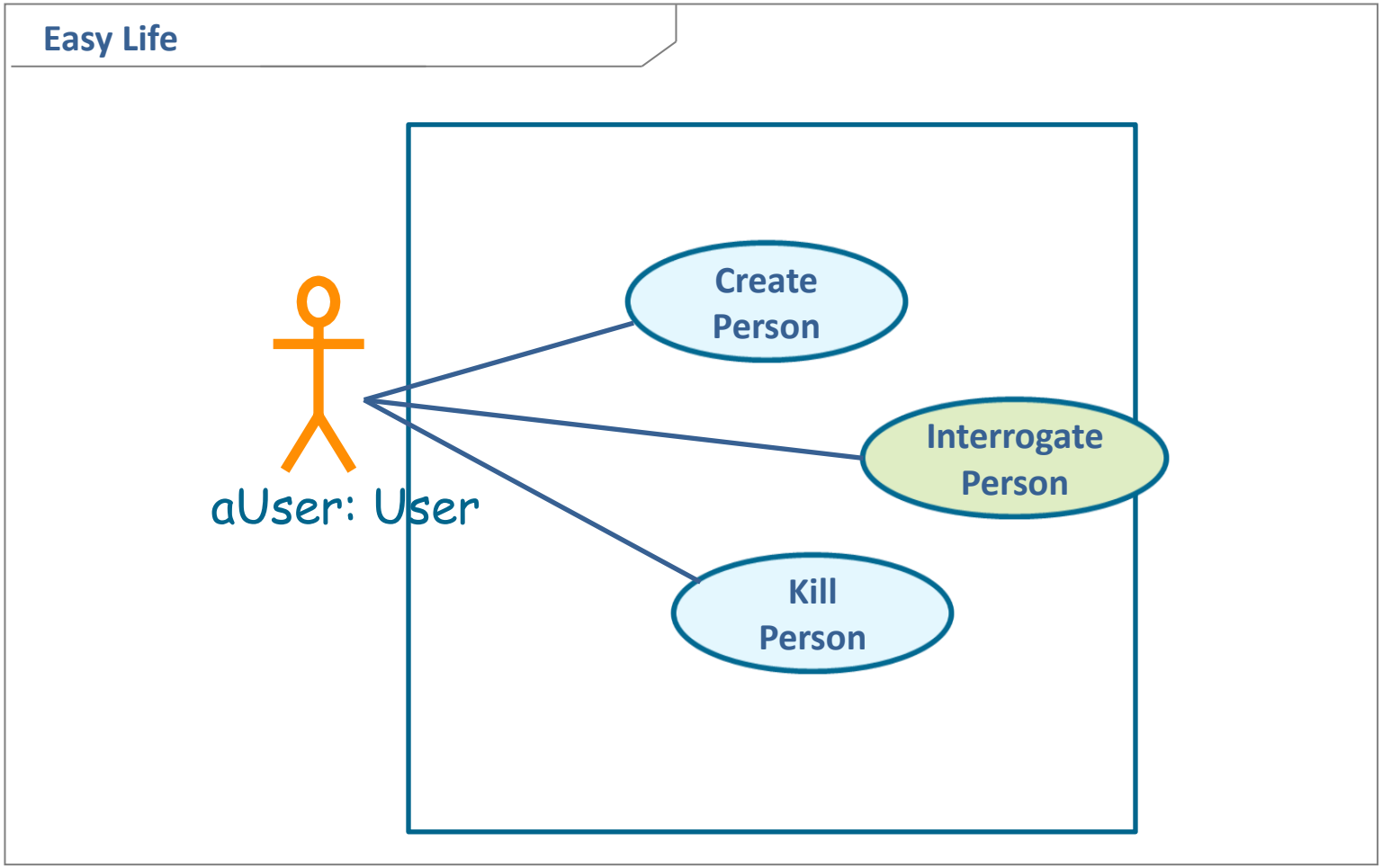




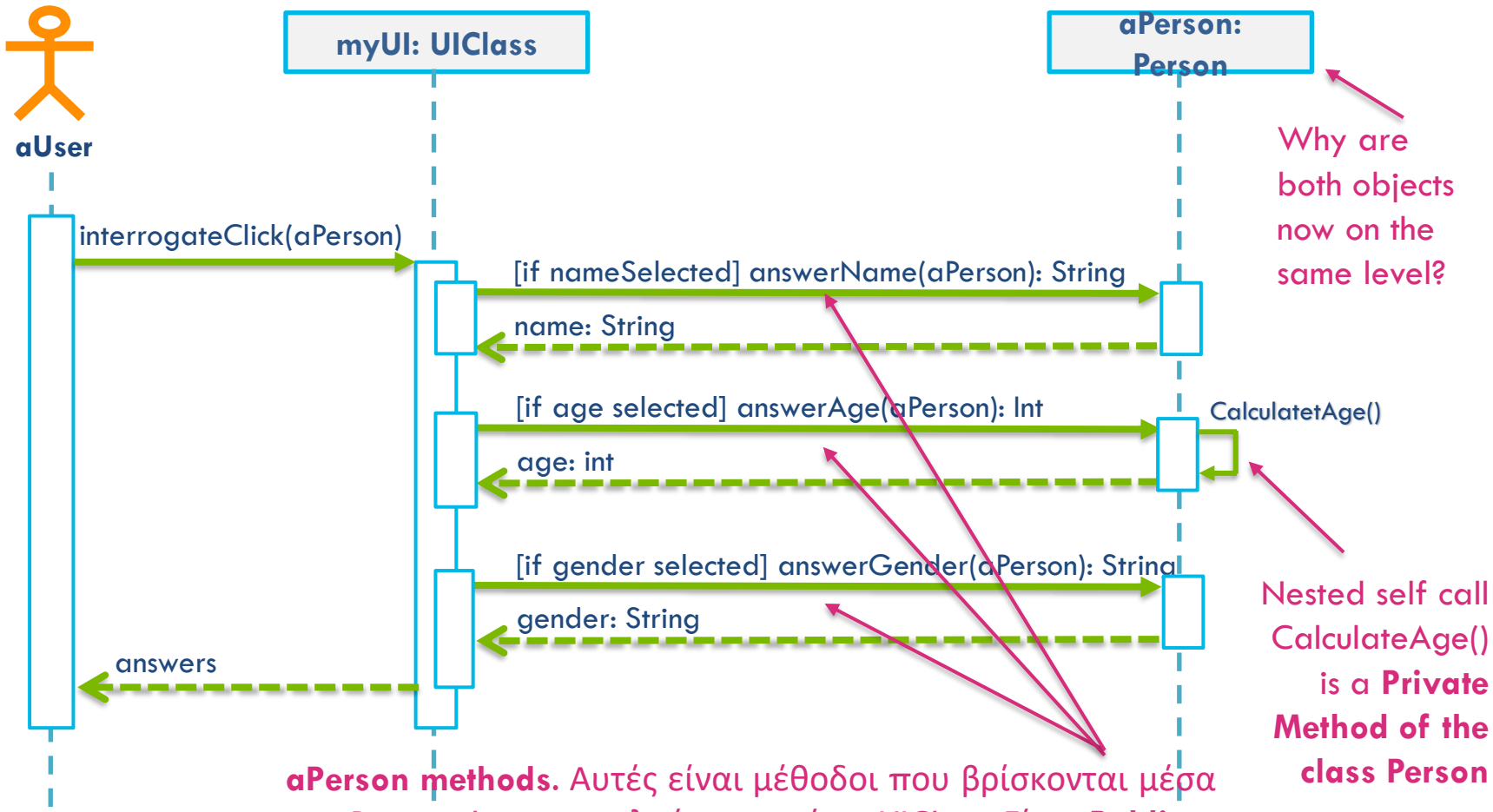
Example 1- An Easy Life
Create Person Sequence Diagram



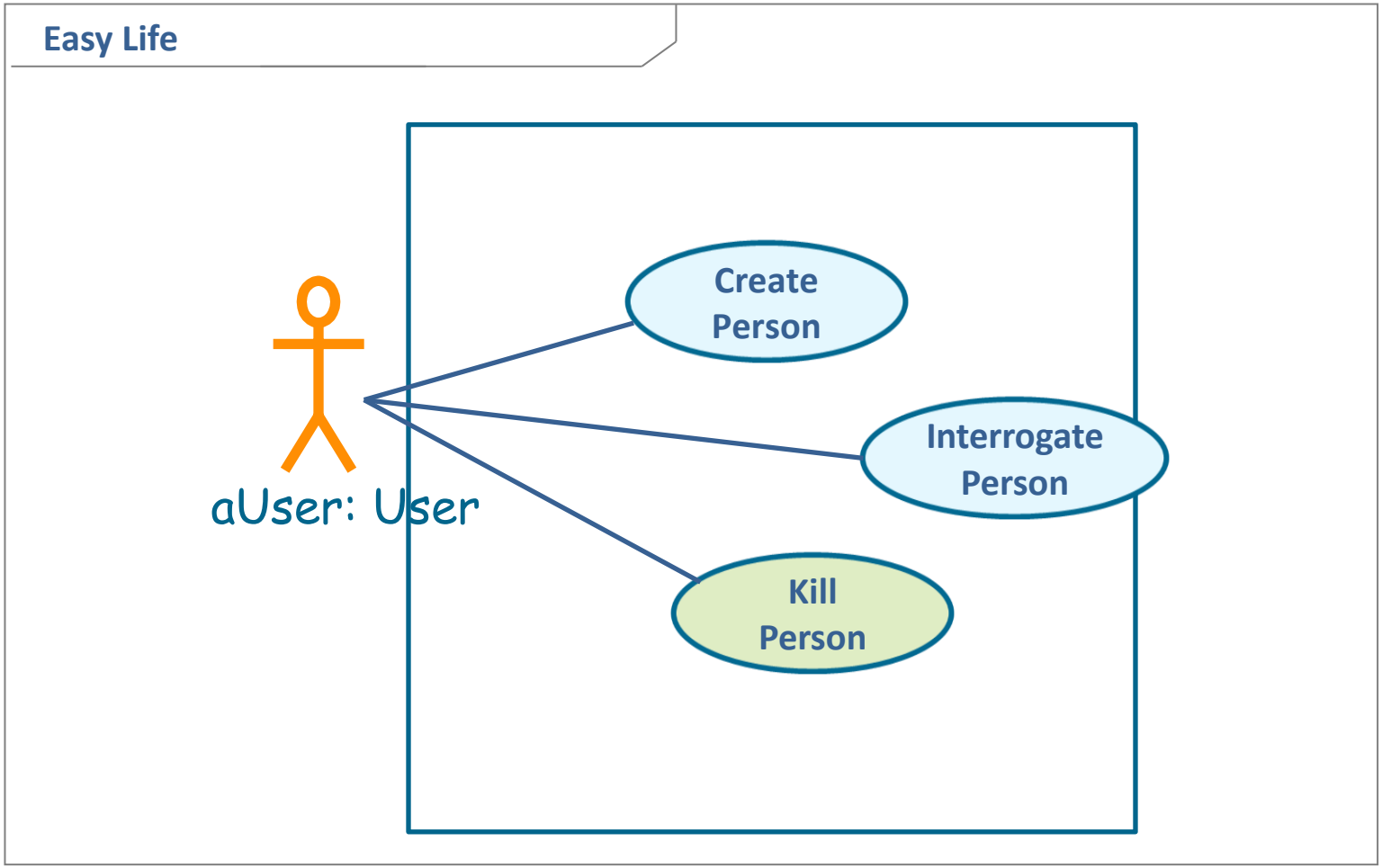
Example 1- An Easy Life
Use Case Diagram

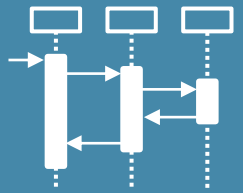


Example 1- An Easy Life
Interrogate Person Sequence Diagram

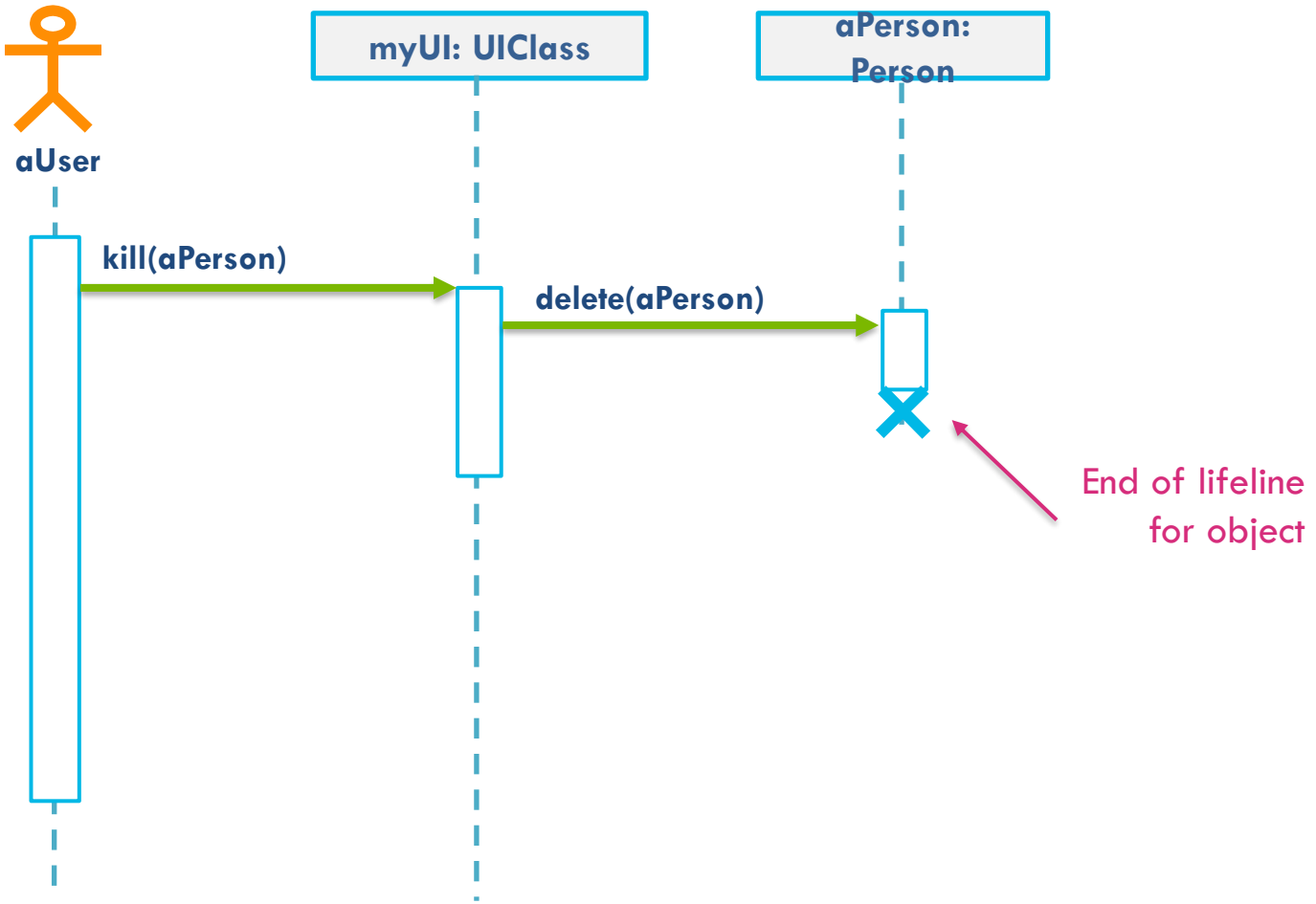


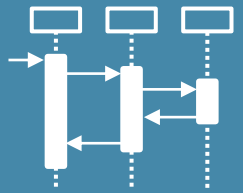
Example 1- An Easy Life
Use Case Diagram





Example 1- An Easy Life
Kill Person Sequence Diagram





Object Sequence Diagrams
annotating OSDs

Interrogate Person Use case scenario

User Clicks Interrogate Button

If question name selected

Answer Name

End If return name

If Question age Selected

Answer age

Person calculates Age

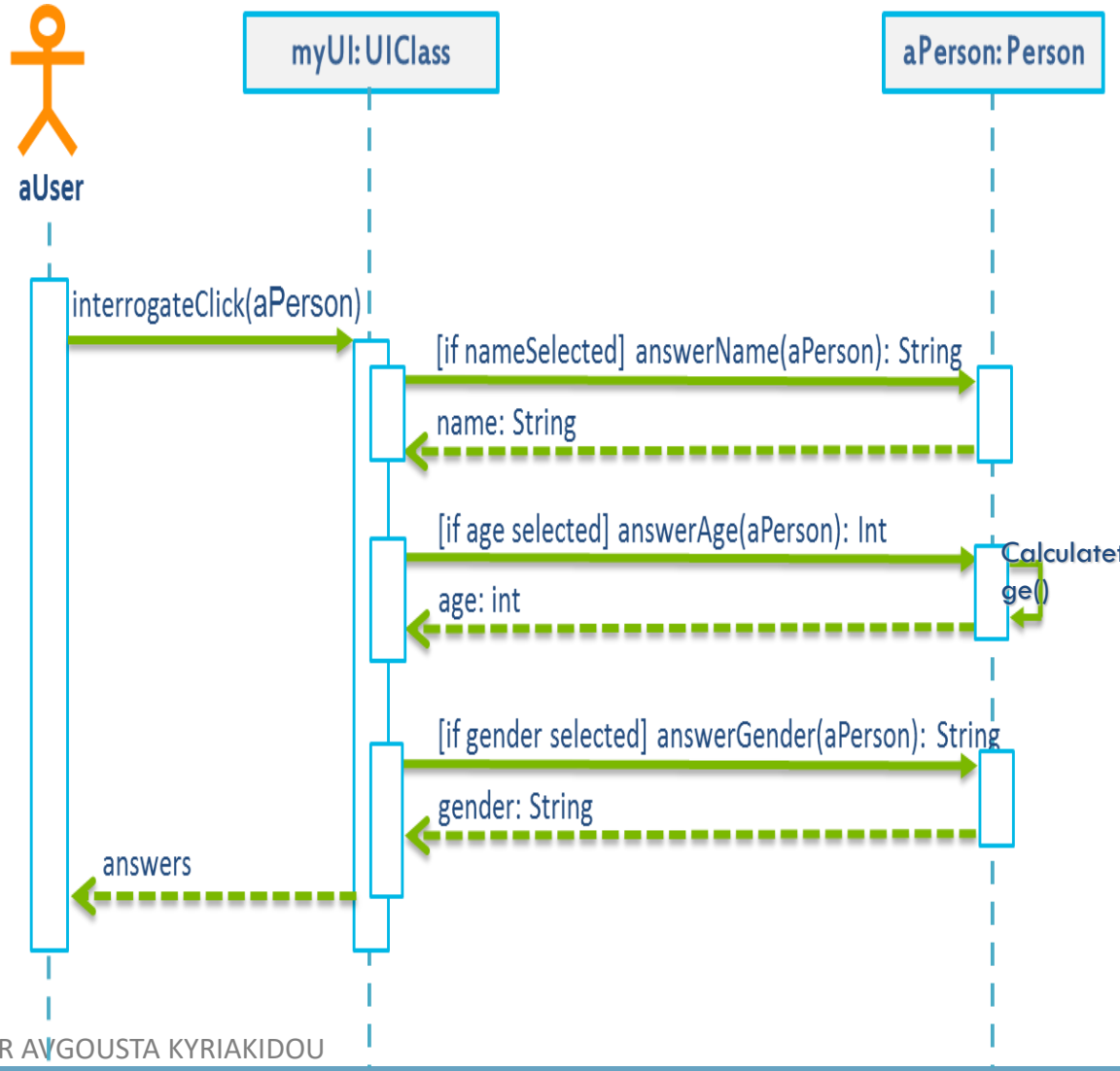
End If return age

If Question gender selected

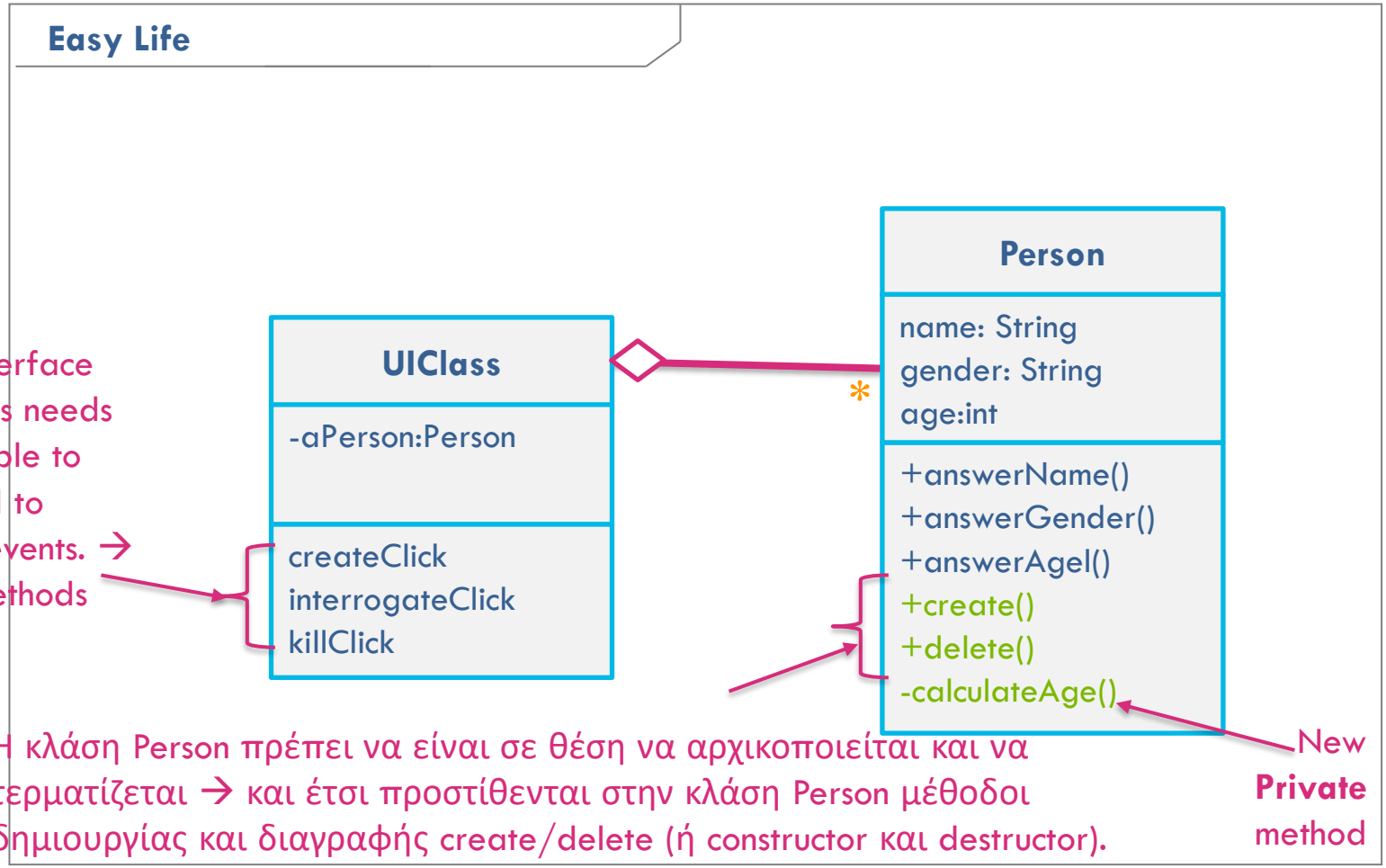
Answer gender

End If return gender

Interrogation answers returned



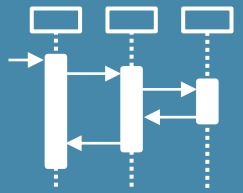
Example 1- An Easy Life
Updating Conceptual Class Diagram



User Interface (UI) class needs to be able to respond to "click" events. → Click methods

New **Private** method

Η κλάση Person πρέπει να είναι σε θέση να αρχικοποιείται και να τερματίζεται → και έτσι προστίθενται στην κλάση Person μέθοδοι δημιουργίας και διαγραφής create/delete (ή constructor και destructor). Αυτές είναι public methods οι οποίες αναγνωρίστηκαν από το OSD



OSD - UML Notation

even more notation..

- Probes

  για να αποφύγετε τα μεγάλα, περίπλοκα διαγράμματα ακολουθίας, χωρίστε τα και χρησιμοποιήστε probes για να συνδέσετε τα διαγράμματα μεταξύ τους

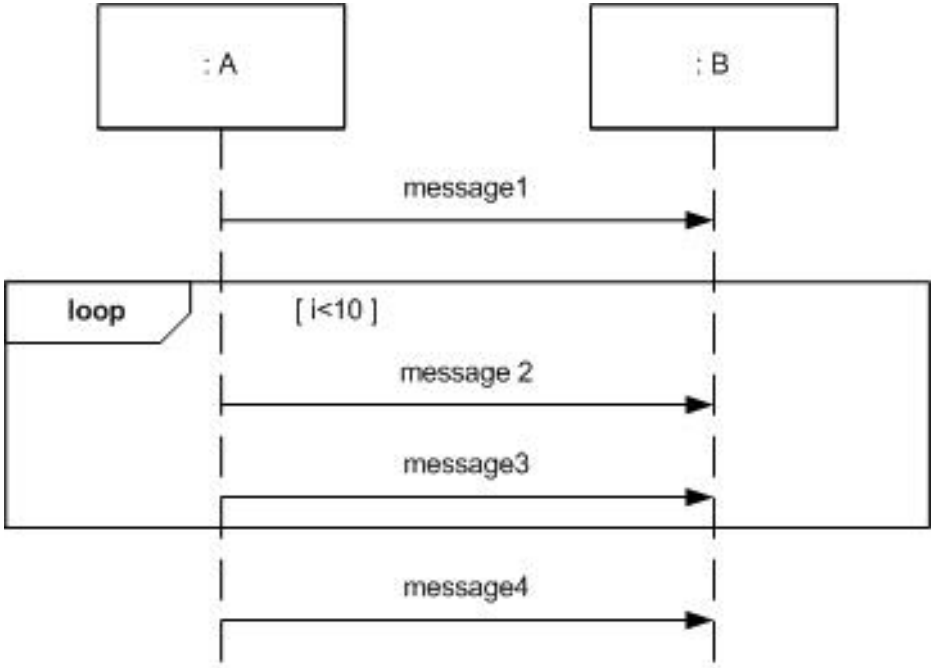
  - to model <<extends>> (to link to another <<extends>> use case)
  - to model <<includes>> (to link to another <<includes>> use case)

- Stereotypes

  εικονίδια που χρησιμοποιούνται για την περιγραφή κοινών τύπων αντικειμένων και actors σε λογισμικά UML

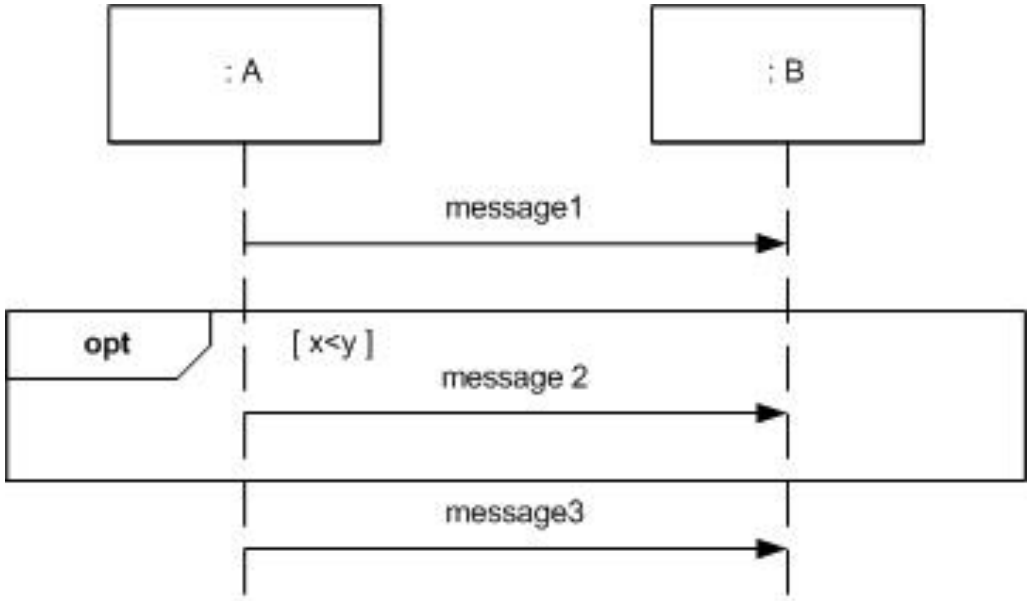
Object Sequence Diagram
Επανάληψη

- Για την επανάληψη χρησιμοποιείται πλαίσιο (frame) με την ένδειξη loop
- Η επανάληψη συνοδεύεται με την αντίστοιχη συνθήκη



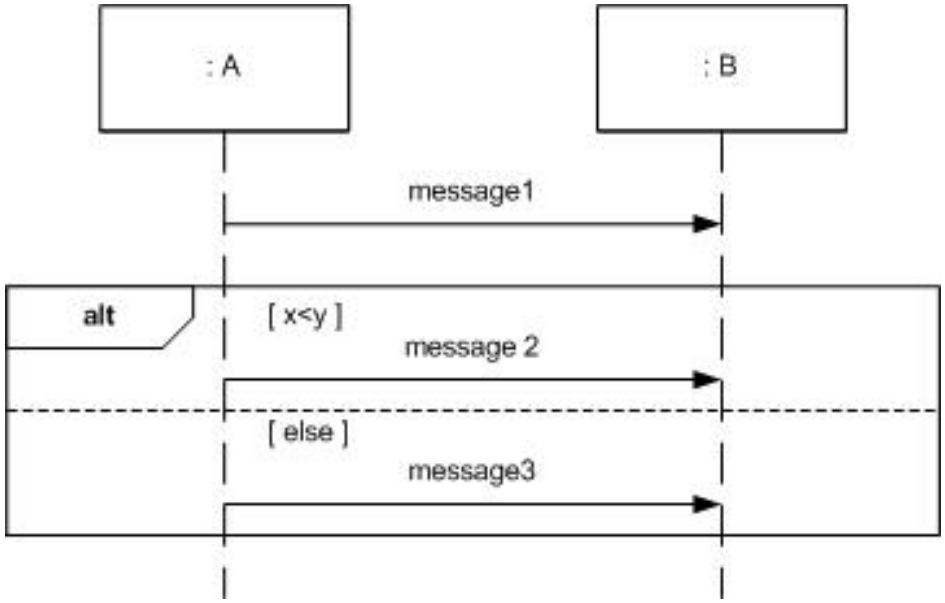
Object Sequence diagrams
Συνθήκη If

- Για την αποστολή μηνυμάτων υπό συνθήκη (if) χρησιμοποιείται το πλαίσιο με την ένδειξη opt.
- Συνοδεύεται και από την αντίστοιχη συνθήκη

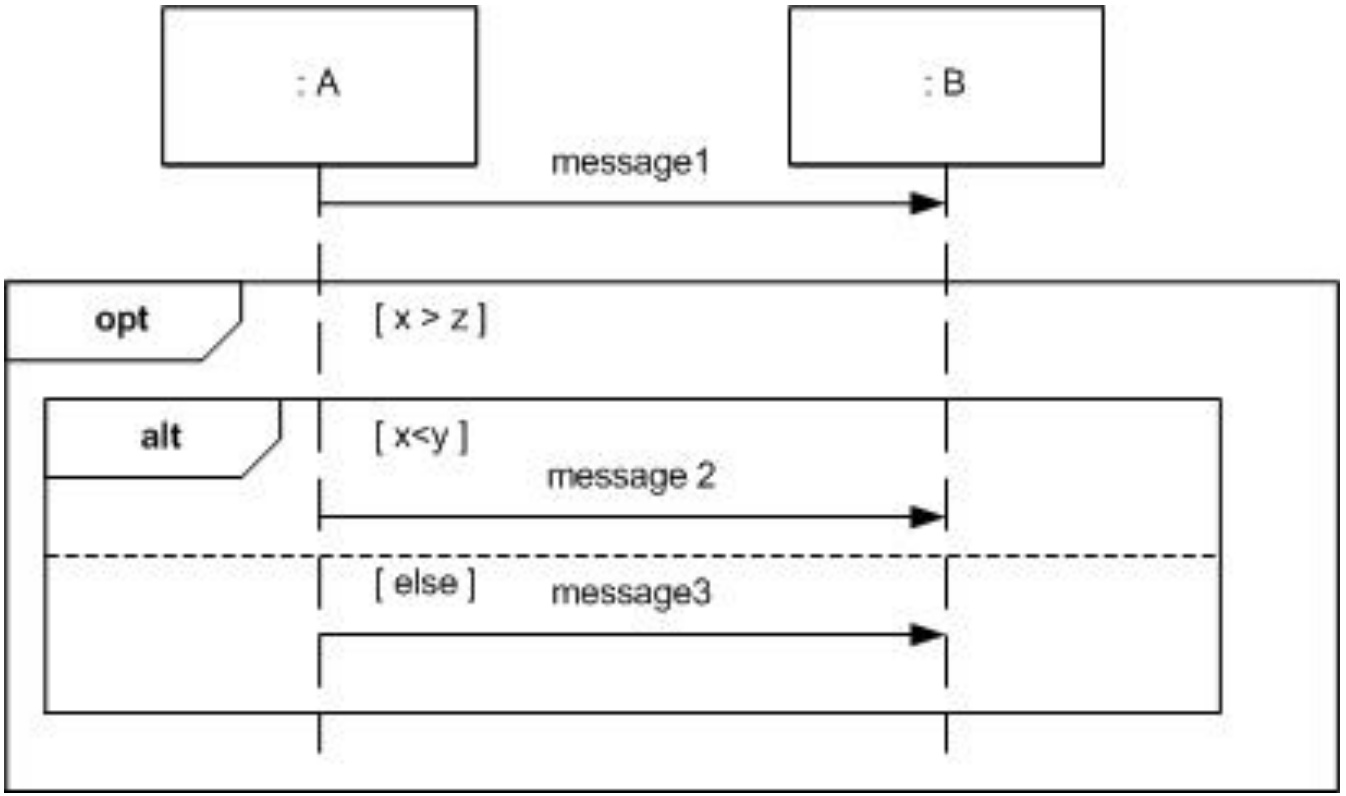


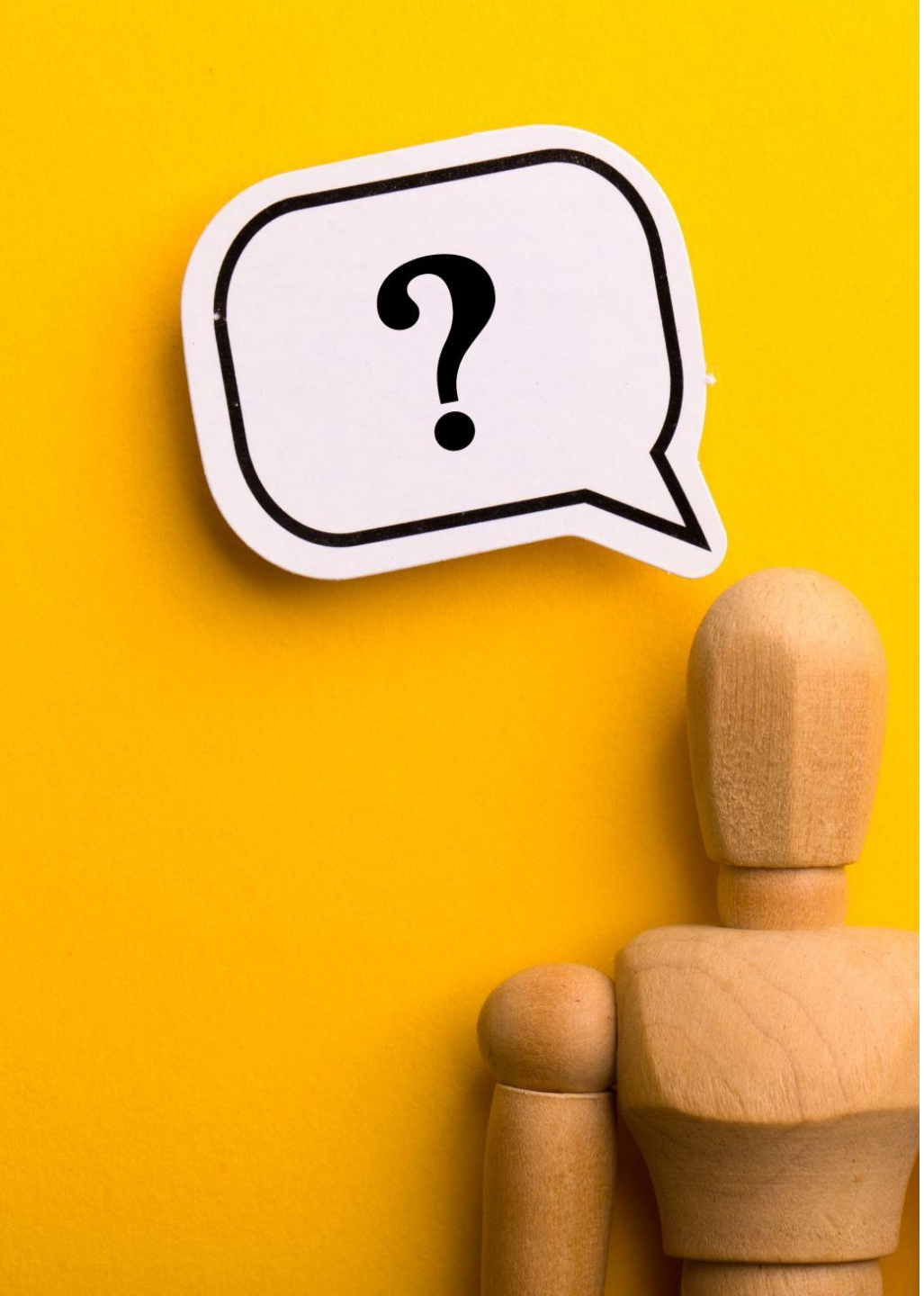
Object Sequence diagrams
Συνθήκη If- else

- Για την αποστολή μηνυμάτων με αμοιβαίο αποκλεισμό (if – else) χρησιμοποιείται το πλαίσιο με την ένδειξη alt
- Η διακεκομμένη γραμμή υποδεικνύει τον αμοιβαίο αποκλεισμό

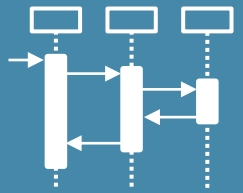


Object Sequence diagrams
περικλειόμενα πλαίσια



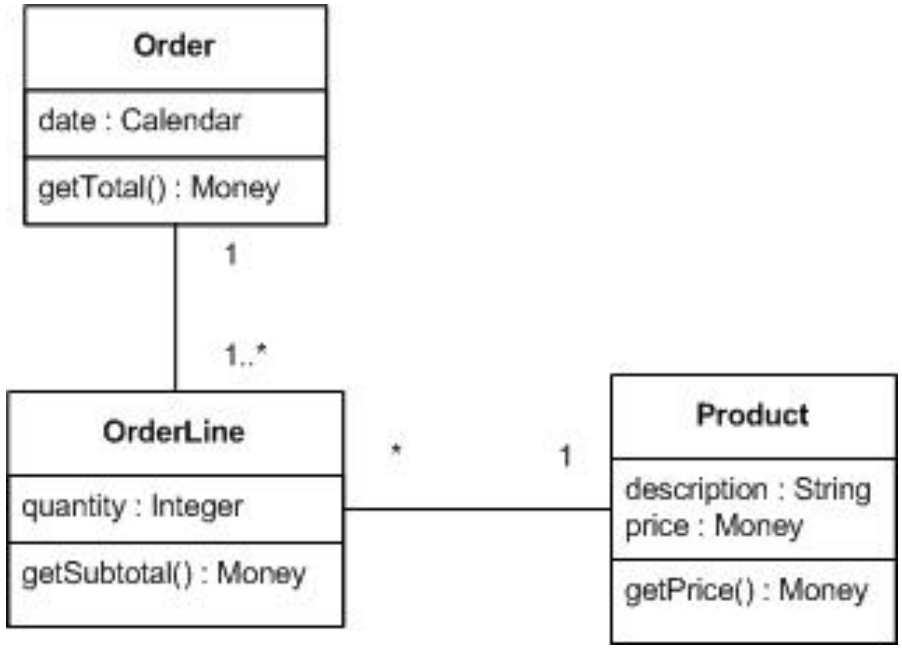


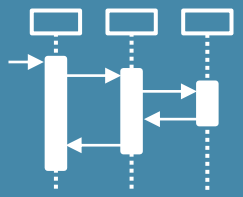
BREAKOUT SESSION



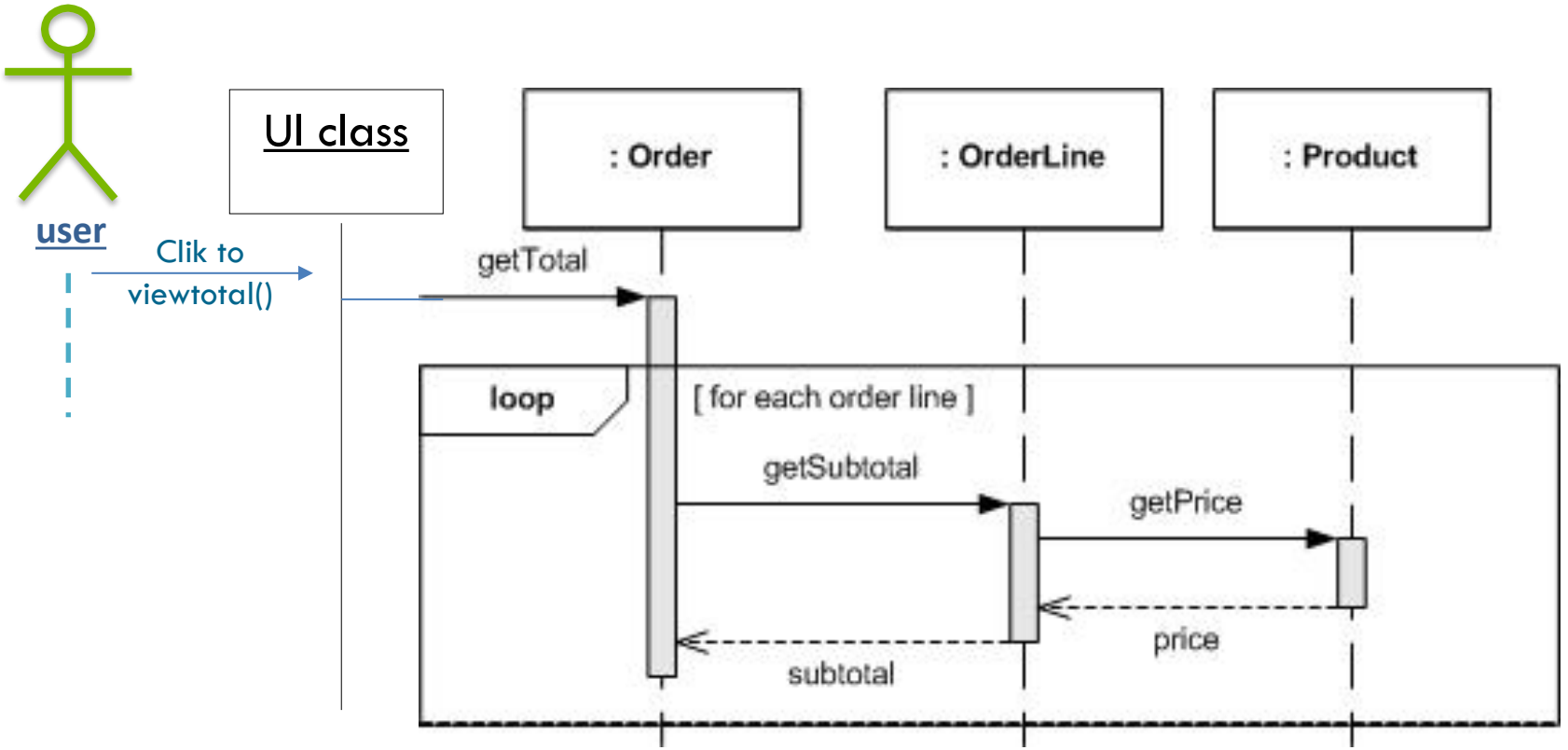
Παράδειγμα – σύστημα παραγγελίας
Ενημέρωση του διαγράμματος κλάσεων

- Ζητείται ένα Object sequence diagram που θα εμφανίζει την επικοινωνία των αντικειμένων για τον υπολογισμό της συνολικής αξίας της παραγγελίας
- Η ανταλλαγή μηνυμάτων σε ένα object sequence diagram θα πρέπει να ενημερώνει και τις λειτουργίες (μεθόδους) του αντίστοιχου διαγράμματος κλάσεων

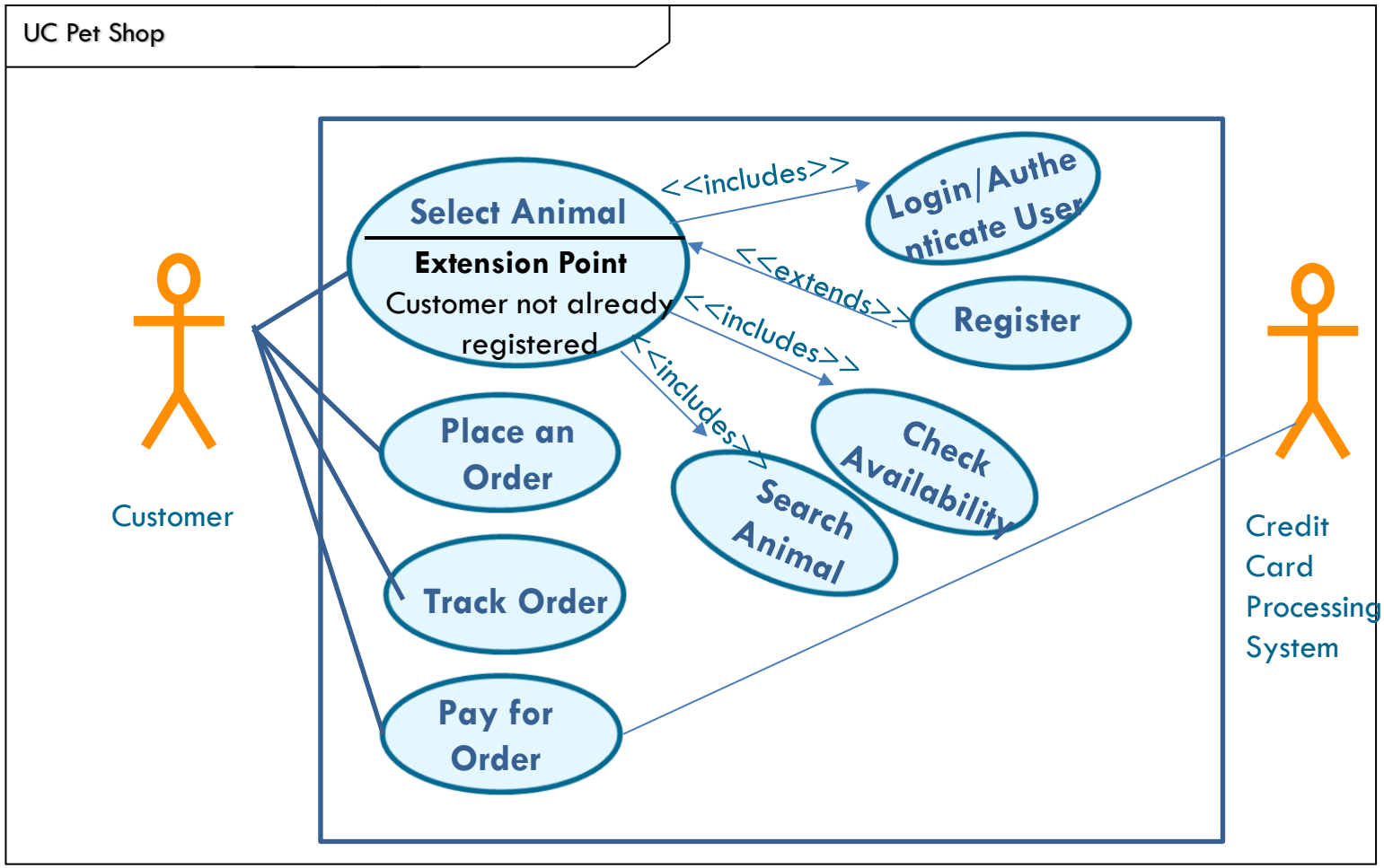




Παράδειγμα – σύστημα παραγγελίας
Object sequence diagram



Example 3- Online Pet Shop
Use Case Diagram



Use case name	Select an animal	
Actors	Customer	
Description	This describes how a customer can select a particular animal from the website	
Typical course of events	**Actor Action** **Step 1:**This use case is initiated by a customer Step 3: Customer provides user name and password. **Step 5:** Customer can search the catalogue by name. **Step 7:** Customer selects an animal from the list. Step 9: Confirmation of availability received.	**System response** Step 2: Invoke <<includes>> use case Authenticate User. Step 4: Login ok. Invoke <<includes>> use case Search Animal. Customer is presented with a catalogue with a selection of search criteria. **Step 6:** Customer is presented with a search result (list of animals based on the name search.) **Step 8:** Invoke <<includes>> check availability. If available, sent confirmation.
Alternate courses	Step 2: If the customer is not already registered, then Invoke <<extends>> use case, Register. Step 8 If the customer chooses an animal that is not available, then a notification is sent to the customer.	
Precondition	This use case is initiated by customers	
Post condition	List of animals	

Object sequence example

Select an Animal Use case

1) **Customer:** the actor from the use case diagram initiating the use-case Choose animal.

2) **OrderingWebpage:** **Boundary class** (an actor interacts with the system through a UI.

3) Classes and Objects

Customer = all the objects of the class Customers

Instance: Animal = this is the single chosen object of the class Animal.

Search results = we need this class since we aren't supposed to display the entire contents of the catalogue