DIS501 ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ

Lecture 8: Δυναμικό Μοντέλο: Διαγράμματα ακολουθίας αντικειμένων (Object Sequence Diagrams)

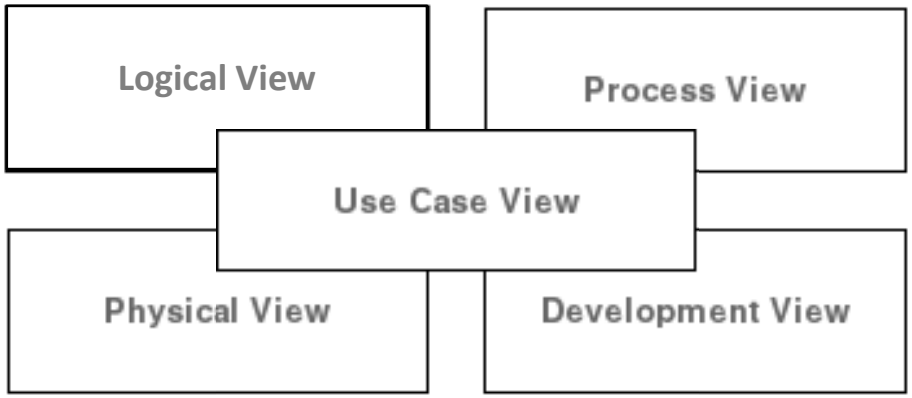
ΒΥ: ΔΡ ΑΥΓΟΥΣΤΑ ΚΥΡΙΑΚΙΔΟΥ ΖΑΧΑΡΟΥΔΙΟΥ

ΜΑΘΗΣΙΑΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

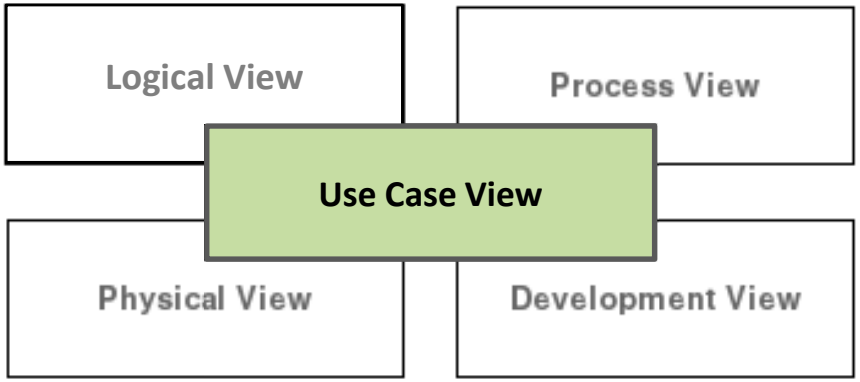
- Εξηγήστε το σκοπό της δυναμικής μοντελοποίησης (dynamic modelling) και των σχετικών διαγραμμάτων
- Ορισμός των διαγραμμάτων ακολουθίας αντικειμένων και συμβολισμός με UML
- Δημιουργία διαγραμμάτων ακολουθίας αντικειμένων (Object Sequence Diagrams)
- Παροχή κατευθυντήριων γραμμών για την αποτελεσματική δυναμική μοντελοποίηση

System Analysis
learning check...

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

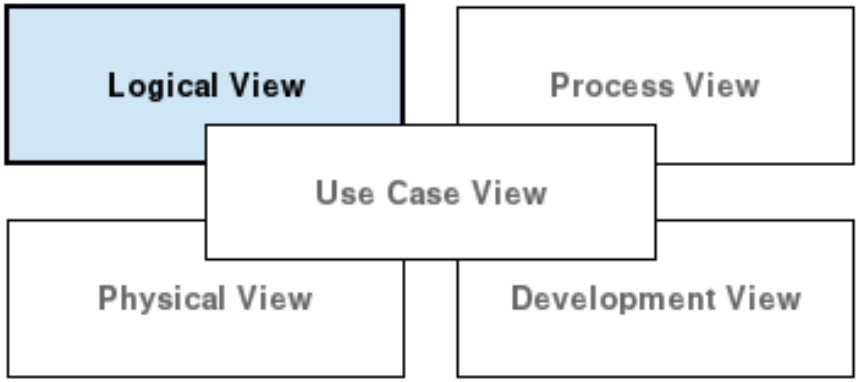


System Analysis
learning check...

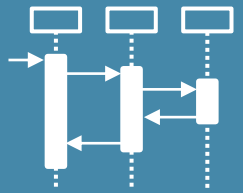


- **Διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams)**
  - Οι λειτουργικές απαιτήσεις (functional requirements) απεικονίζονται ως περιπτώσεις χρήσης (use cases)
  - Περιγραφή του συστήματος με γνώμονα τον χρήστη
  - Απεικονίζει τον τρόπο με τον οποίο οι τελικοί χρήστες αλληλοεπιδρούν με το σύστημα

System Analysis
learning check...



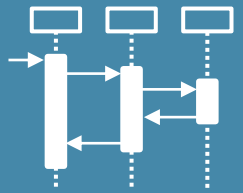
- **Διάγραμμα Κλάσεων Ανάλυσης (Conceptual Class Diagram)**
  - Ένα αφαιρετικό μοντέλο που αποτυπώνει τις σημαντικές κλάσεις και τις σχέσεις τους, όπως αυτές υπάρχουν μέσα στο πρόβλημα που έχουμε να επιλύσουμε.
  - Μια "στατική" εικόνα ('static' picture) των κύριων αντικειμένων/κλάσεων (Objects/classes)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

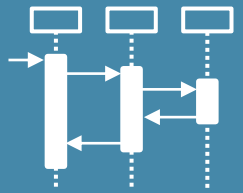
Σε αυτό το στάδιο της ανάλυσης, δεν είναι δυνατόν να προσδιοριστούν:

- 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

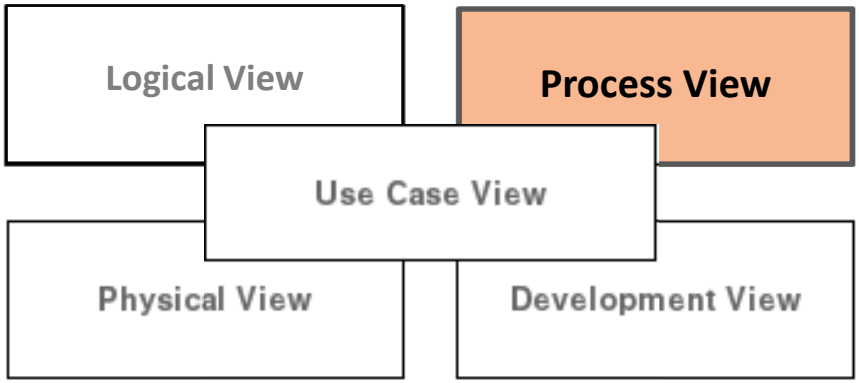
Τι προσπαθούμε να δείξουμε?

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

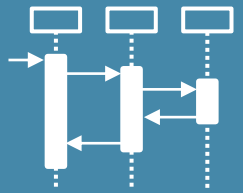
- Έχουμε μοντελοποιήσει μια προκαταρκτική, στατική άποψη του συστήματος **(object modelling and conceptual class diagram**)

- 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**



Μοντελοποίηση δυναμικής συμπεριφοράς με UML

Η UML προσφέρει διάφορους τύπους διαγραμμάτων για την αποτύπωση αυτής της δυναμικής συμπεριφοράς.

- Διαγράμματα αλληλεπίδρασης - Interaction Diagrams

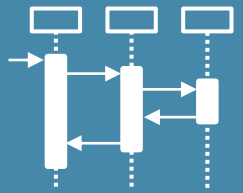
  Απεικονίζουν τον τρόπο με τον οποίο ένα σύνολο αντικειμένων/κλάσεων εμπλέκεται σε ένα συγκεκριμένο σενάριο περίπτωσης χρήσης (use case scenario) και τα μηνύματα που μπορούν να μεταβιβαστούν μεταξύ τους.

- Διαγράμματα κατάστασης - State transition Diagrams

  Τα διαγράμματα καταστάσεων αποτελούνται από καταστάσεις, μεταβάσεις, γεγονότα και δραστηριότητες (states, transitions, events, and activities) ενός αντικειμένου. Παρουσιάζουν μια πιο λεπτομερή άποψη ενός αντικειμένου/κλάσης (Object/Class).

- Διαγράμματα δραστηριοτήτων - Activity Diagrams

  Παρουσιάζουν τη ροή ελέγχου ενός συστήματος και δείχνουν τα βήματα τα οποία χρειάζεται να γίνουν για να υλοποιηθεί μια περίπτωση χρήσης (use case) του συστήματος



Modelling Dynamic Behaviour
with UML

- Διαγράμματα αλληλεπίδρασης - Interaction Diagrams
  - **Διαγράμματα ακολουθίας αντικειμένων - Object Sequence Diagrams**

    Μοντελοποιούν τον τρόπο με τον οποίο αλληλοεπιδρά ο Actor με τις κλάσεις του συστήματος και οι κλάσεις μεταξύ τους για μια συγκεκριμένη περίπτωση χρήσης. Δείχνουν τη σειρά με την οποία γίνεται αυτή η αλληλεπίδραση, και η ανταλλαγή μηνυμάτων όταν μια συγκεκριμένη περίπτωση χρήσης (use case) του συστήματος ενεργοποιείται.

    Δίνει έμφαση στην σειρά των γεγονότων - Emphasis on time ordering.

  - **Διαγράμματα επικοινωνίας - Communication Diagrams**

    Μια επέκταση του διαγράμματος κλάσεων (class diagram) που δείχνει πως τα objects/classes αλληλεπιδρούν και επικοινωνούν μαζί και με τα μηνύματα που στέλνονται από το ένα στο άλλο.

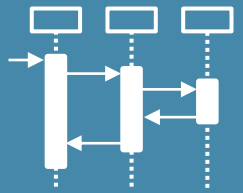
    Δίνει έμφαση στα αντικείμενα και στις μεταξύ τους σχέσεις - Emphasis on structural ordering

Dynamic Modelling
Objectives

- Μοντελοποίηση δυναμικής συμπεριφοράς
- Διαγράμματα αλληλεπίδρασης - Διαγράμματα ακολουθίας αντικειμένων (Object Sequence diagrams)
- Συμβολισμός με τη χρήση της UML
- Κατασκευή διαγραμμάτων ακολουθίας αντικειμένων (Object Sequence diagrams)

ΣΥΝΔΥΑΖΟΝΤΑΣ ΤΑ ΔΕΔΟΜΕΝΑ ΠΟΥ ΕΧΟΥΜΕ ΜΕΧΡΙ ΣΤΙΓΜΗΣ:

ΤΟ ΔΥΝΑΜΙΚΟ ΜΟΝΤΕΛΟ ΚΑΙ ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΑΚΟΛΟΥΘΙΑΣ ΑΝΤΙΚΕΙΜΕΝΩΝ (OBJECT SEQUENCE DIAGRAMS)

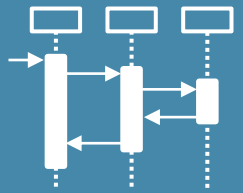


Object Sequence Diagrams (OSDs)
Σκοπός

- 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

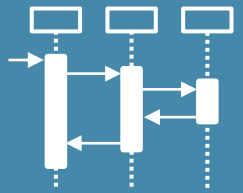


Object Sequence Diagrams (OSDs)
σκοπός

- 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 → **Use-Case Diagrams (UCD)**

Iterative / incremental process

Start with → **Conceptual Class Diagram**

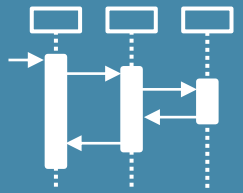
Μας δίνει τα Use-Cases και τα scenarios

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).

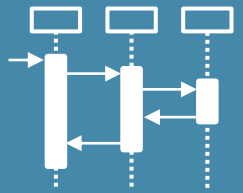
Ένα ή περισσότερα OSD για κάθε περίπτωση χρήσης (use case) στο Use case diagram



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.*

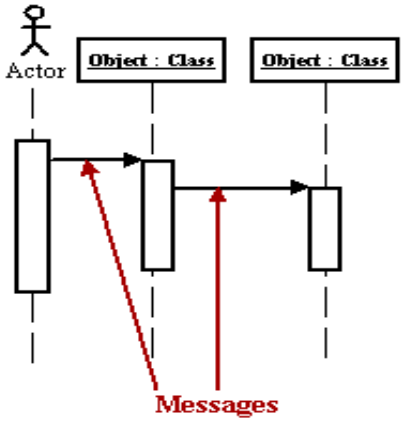
- **Σκεφτείτε ότι ένας χρήστης (actor) πατάει ένα κουμπί για να αναζητήσει προϊόντα (use case).**
  - Αυτό πυροδοτεί μια σειρά αλληλεπιδράσεων εντός του προγράμματος, οι οποίες σε ένα αντικειμενοστραφές σύστημα θα υλοποιηθούν ως συνεργασία αντικειμένων με τη μορφή μεταβίβασης μηνυμάτων μεταξύ τους.
  - 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

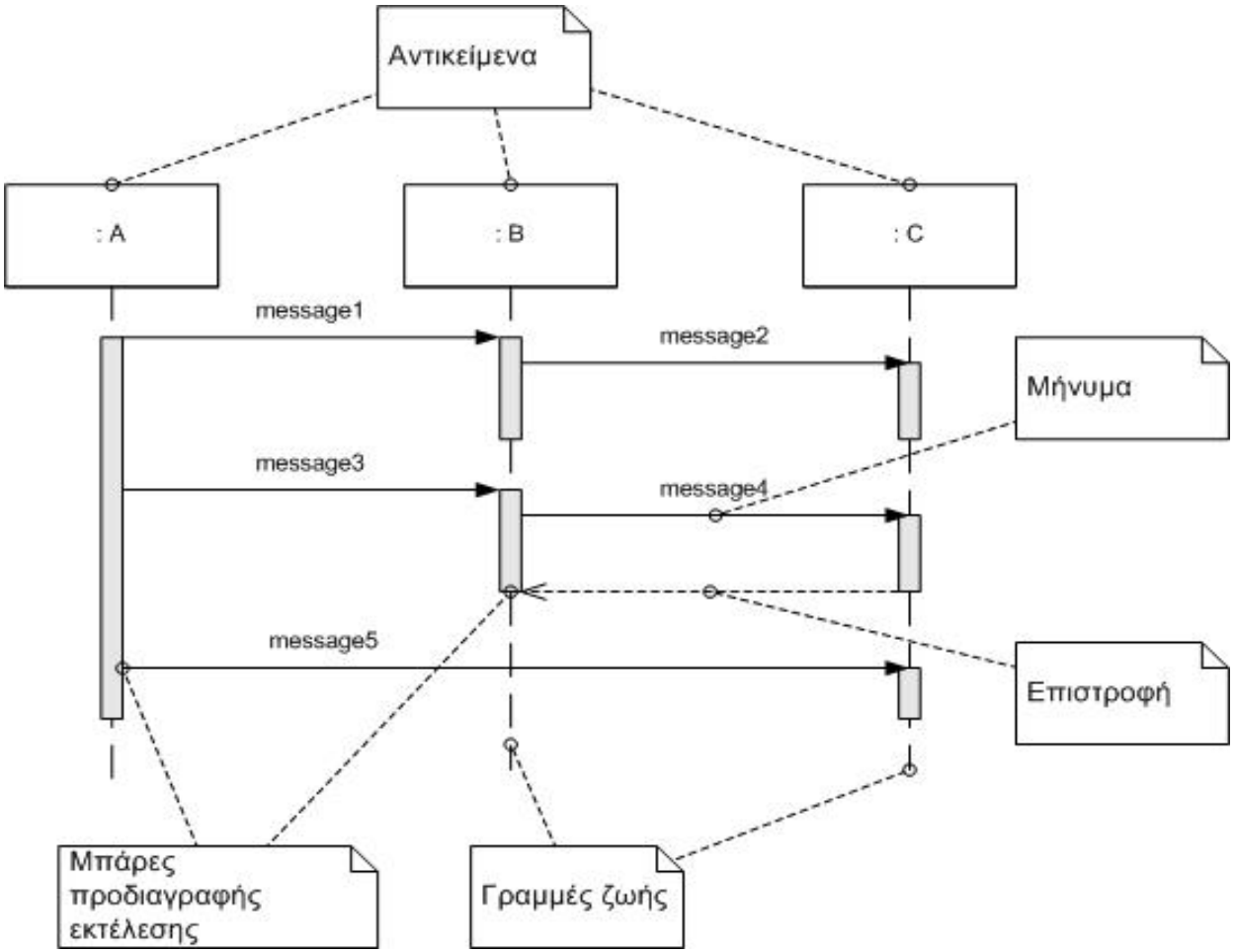
- Actors
  - προσδιορίζονται από το διάγραμμα περίπτωσης χρήσης (Use case diagram)
  - μόνο εκείνοι που ενεργοποιούν (initiate/invoke/trigger) την περίπτωση χρήσης (use case) που περιγράφουμε με ένα OSD. Άρα οι Primary (πρωτεύων) actors.

- 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

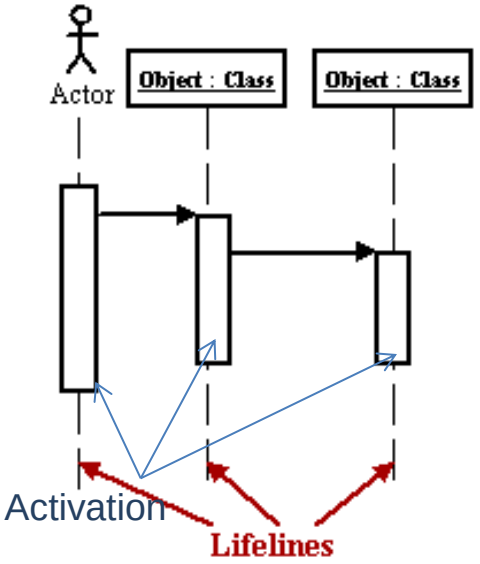
Object Sequence Diagrams (OSD)
key constructs



Η σειρά των μηνυμάτων φαίνεται από την τοποθέτηση τους στον κατακόρυφο άξονα

Object Sequence Diagrams
key constructs, *Cntd.*

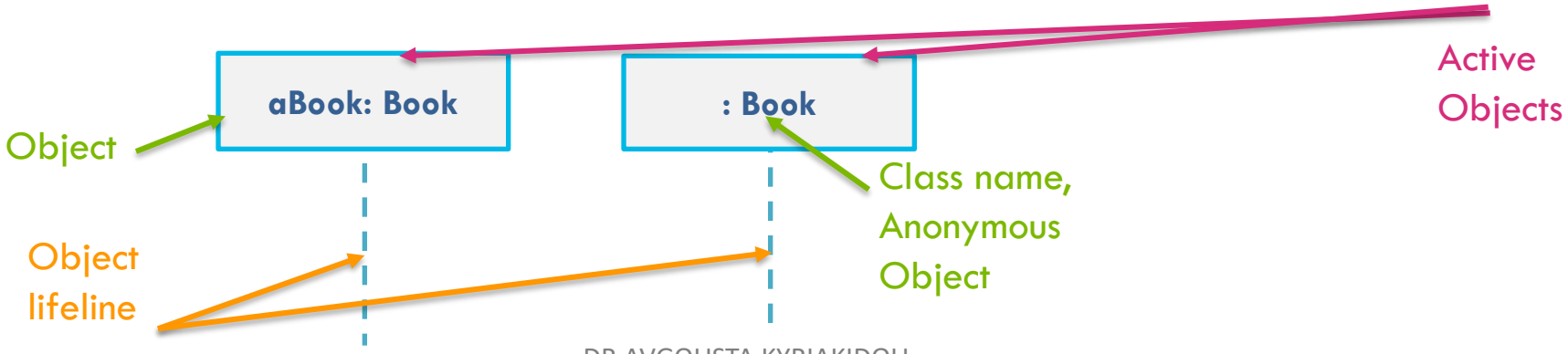
- **Χρονικοί Άξονες - Time Axes (in OSD)**
  - **Οριζόντιος - Horizontal (Object Dimension)**
    - Δείχνει τα object/classes που συνεργάζονται/ενεργούν σε μια περίπτωση χρήσης (use case)
    - Υποδεικνύει πότε δημιουργούνται τα αντικείμενα
  - **Κατακόρυφος - Vertical (Time Dimension)**
    - απεικονίζει το σχετικό χρόνο (χρονική σειρά)
    - προς τα κάτω -> προς τα εμπρός στο χρόνο
- **Μπάρες προδιαγραφής εκτέλεσης - Activation**
  - χρονική περίοδος κατά την οποία ένα αντικείμενο εκτελεί μια μέθοδο/λειτουργία
  - ένα ενεργοποιημένο αντικείμενο είτε εκτελεί μία δική του λειτουργία είτε περιμένει την επιστροφή, όταν αποστέλλει ένα μήνυμα σε κάποιο άλλο αντικείμενο.

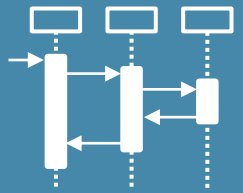


OSD - UML Notation
Objects

- 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

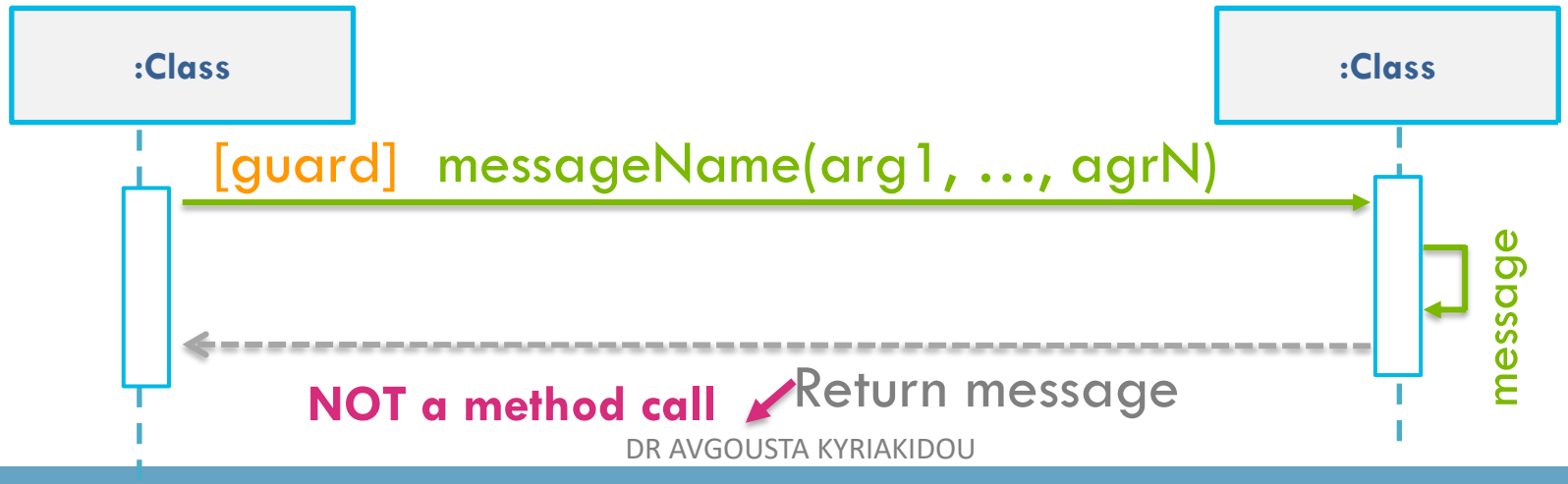
      Name Syntax= <objectName>:<className>
    - Το όνομα του αντικειμένου είναι απαραίτητο μόνο εάν διευκρινίζει καλύτερα το διάγραμμα αλλιώς αποφεύγεται.
  - Γραμμή ζωής - Life-line
    - αναπαριστά τη διάρκεια ζωής των αντικειμένων και σχεδιάζεται ως κατακόρυφη διακεκομμένη γραμμή κάτω από τα αντικείμενα.



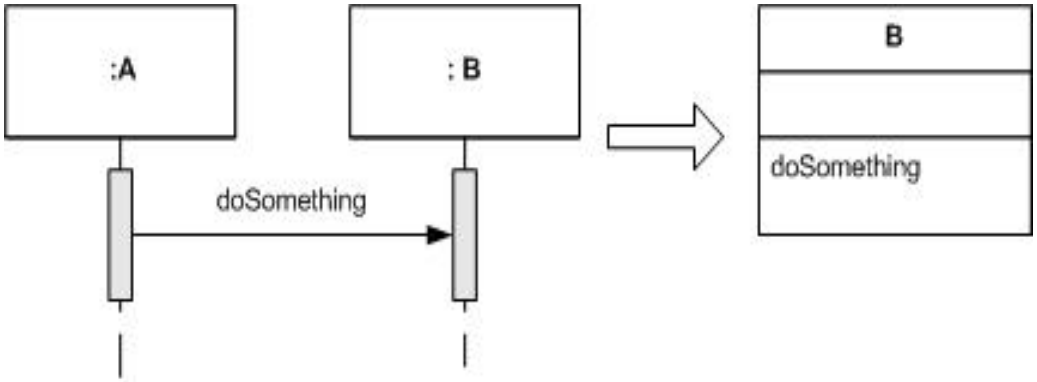


OSD - UML Notation
Messages, *Cntd.*

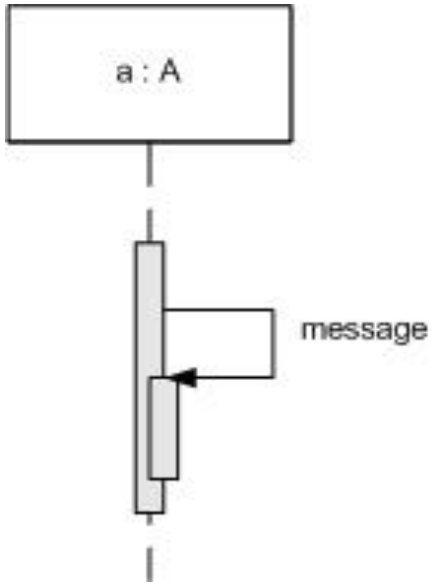
- Ένα μήνυμα (message) είναι μια **κλήση μεθόδου** και των χαρακτηριστικών (attributes) που μεταφέρονται
- **Τα μηνύματα μεταξύ των αντικειμένων αντιπροσωπεύονται από:**
  - Το όνομα του μηνύματος και των attributes που μεταφέρονται σε παρένθεση πάνω από το τόξο
  - Προαιρετικά, μπορούμε να συμπεριλάβουμε τυχόν συναφείς συνθήκες για την αποστολή του μηνύματος
  - Είναι ένα οριζόντιο τόξο από μια κλάση σε άλλη ή από τον πρωτεύοντα actor σε μια κλάση.
- Για να σχεδιάσετε μια κλάση που καλεί εσωτερική συμπεριφορά της (private method), συνδέστε το μήνυμα πίσω στην ίδια κλάση



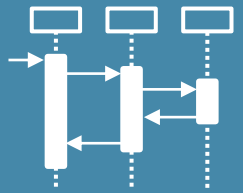
OSD - UML Notation
Messages, *Cntd.*



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

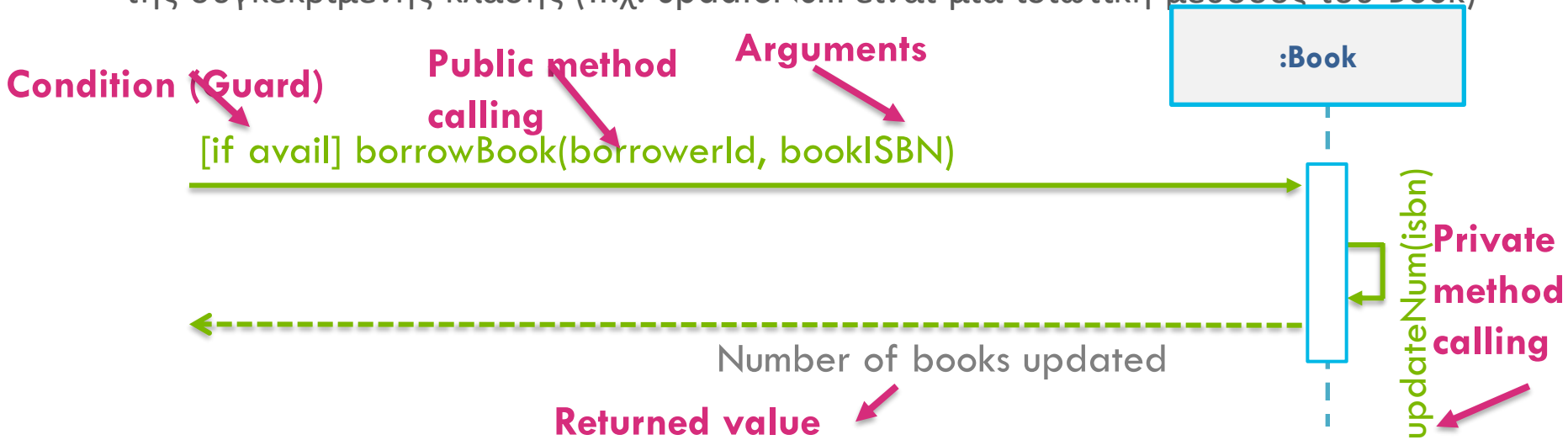


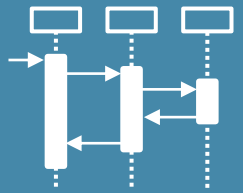
Μήνυμα από την κλάση στον εαυτό της για να ενεργοποιηθεί μια private μέθοδος της κλάσης αυτής.



OSD - UML Notation
Messages, *Cntd.*

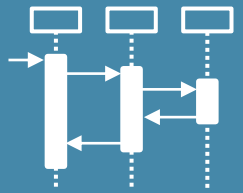
- **Τύποι μηνυμάτων:**
  - Κλήση μεθόδου (Method calling)
  - Μήνυμα επιστροφής (Return message)
- Το μήνυμα στο βέλος υποδηλώνει μια δημόσια μέθοδο (Public method) της κλάσης στο οποίο δείχνει το βέλος (π.χ. borrow book είναι μια δημόσια μέθοδος του Book).
- Ένα μήνυμα που πηγαίνει πίσω στην ίδια την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





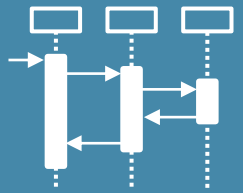
OSD - UML Notation
Activation

- Μπάρες προδιαγραφής εκτέλεσης - Activation
  - Όταν ένα **μήνυμα** αποστέλλεται σε ένα αντικείμενο, **καλεί μια μέθοδο του αντικειμένου αυτού**. Το **όνομα του μηνύματος πρέπει να** είναι το **ίδιο με τη μέθοδο που καλείται.**
  - Μόλις ληφθεί ένα μήνυμα, η λειτουργία/μέθοδος που έχει κληθεί αρχίζει να εκτελείται.
  - Το μικρό ορθογώνιο μπλοκ που τοποθετείται κατά μήκος της γραμμής ζωής **υποδεικνύει τον χρόνο που η μέθοδος που ενεργοποιείται είναι ενεργή**.



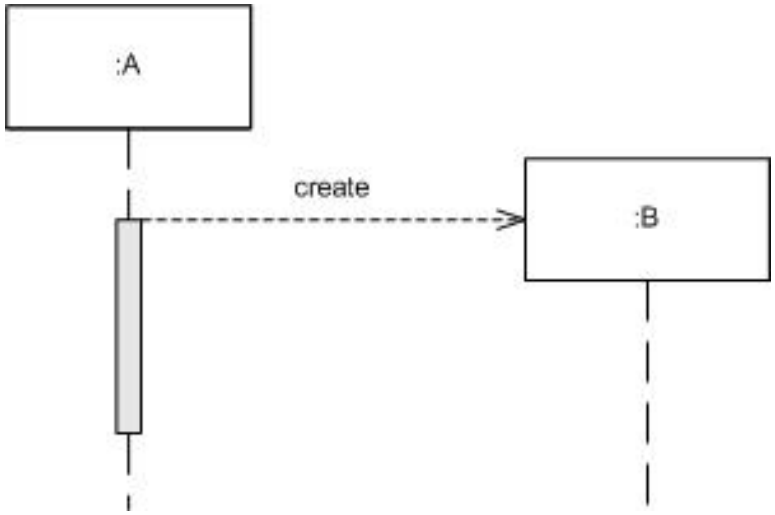
OSD Constructs
Που κοιτάζουμε για να τα βρούμε?

- 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)

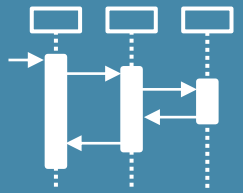


Object Lifetime in OSDs

Δημιουργία και διαγραφή Αντικειμένων
Creation and Destruction



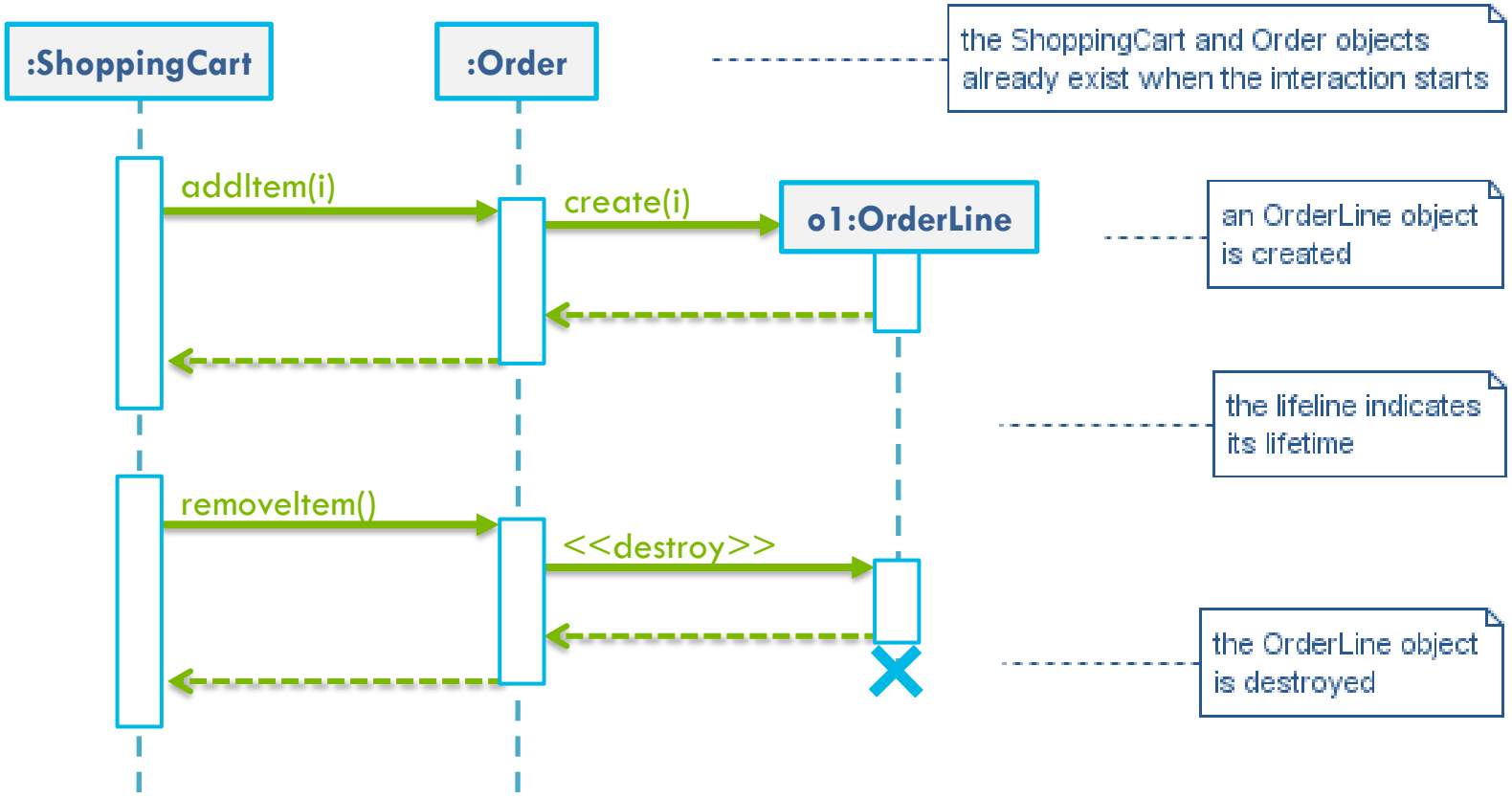
- Η δημιουργία αντικειμένων έχει διαφορετικό συμβολισμό στην αποστολή του μηνύματος
- Το αντικείμενο που δημιουργείται δεν τοποθετείται στην κορυφή του διαγράμματος αλλά στο σημείο που δημιουργείται

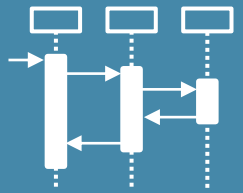


Object Lifetime in OSDs

Δημιουργία και διαγραφή Αντικειμένων
Creation and Destruction

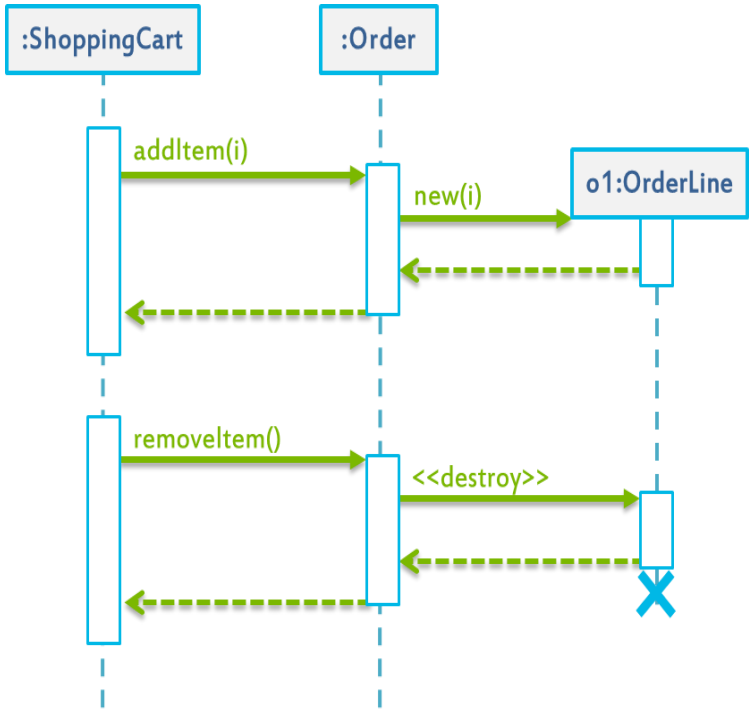
Place order example:

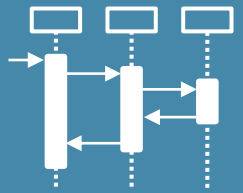




Object Lifetime in OSDs
Creation and Destruction

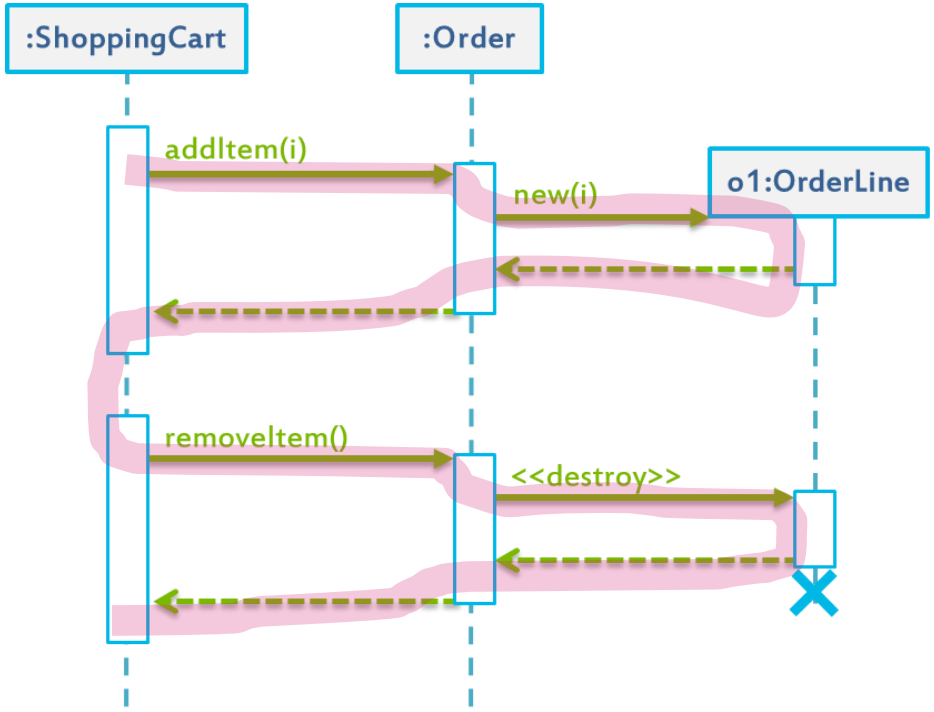
- Δημιουργία - Creation
  - η γραμμή ζωής του αντικειμένου ξεκινά μόνο μετά τη λήψη ενός μηνύματος δημιουργίας
  - βέλος με new/create από πάνω του
  - παρατηρήστε ότι τα αντικείμενα που δημιουργήθηκαν μετά την έναρξη του σεναρίου εμφανίζονται χαμηλότερα από από τις υπόλοιπες κλάσεις
- Διαγραφή - Deletion
  - εμφανίζεται με το X που κόβει τη γραμμή ζωής αντικειμένων
  - Η Java/C# δεν διαγράφει ρητά αντικείμενα- πέφτουν εκτός εμβέλειας και γίνονται garbage collected.
  - Εναλλακτικά μπορούν να μηδενιστούν (Null)

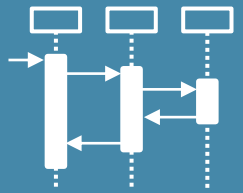




Reading Object Sequence Diagrams
things to note...

- 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

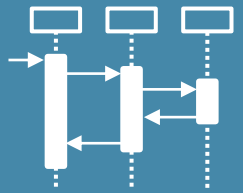




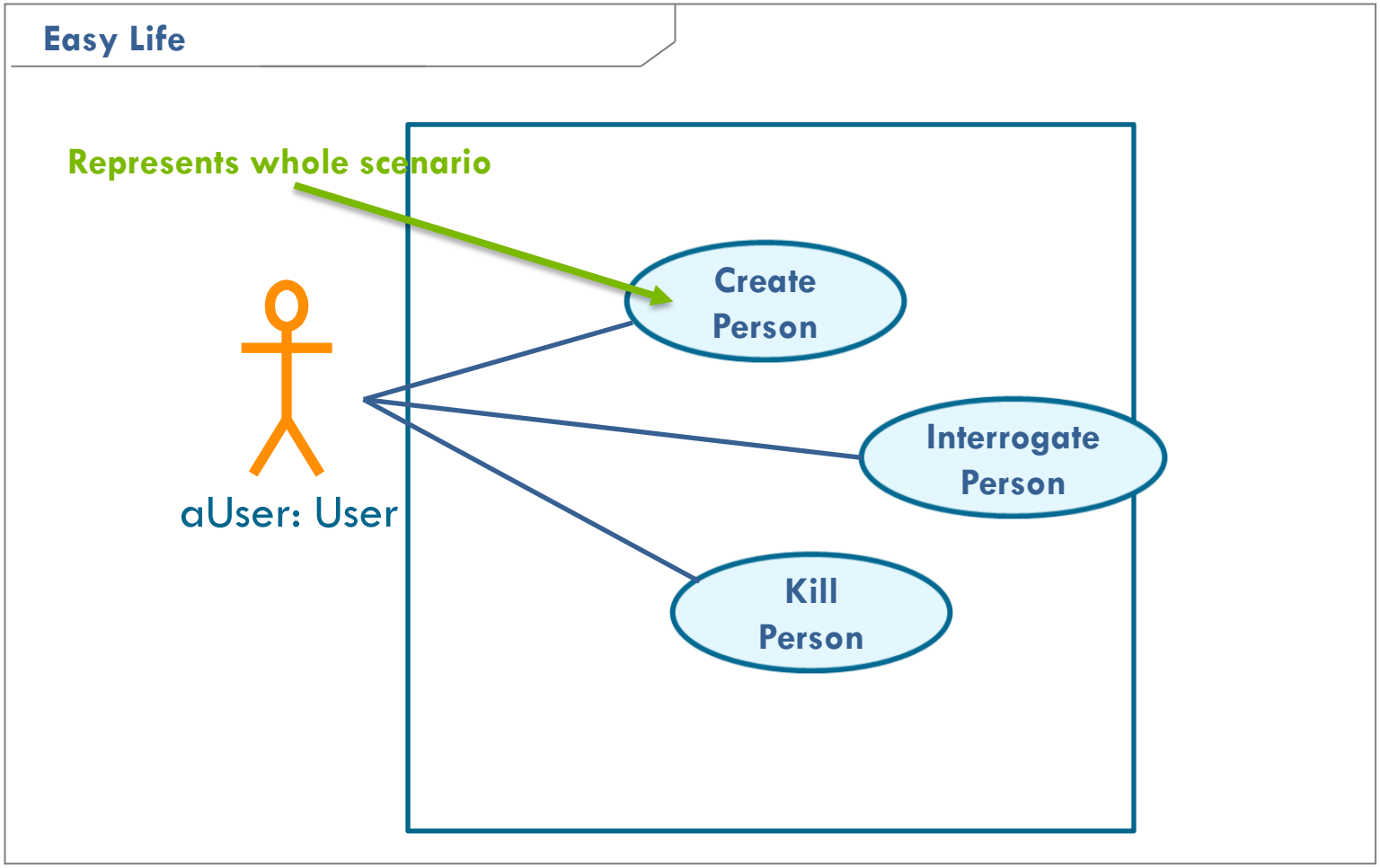
Example 1
An Easy Life

- 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

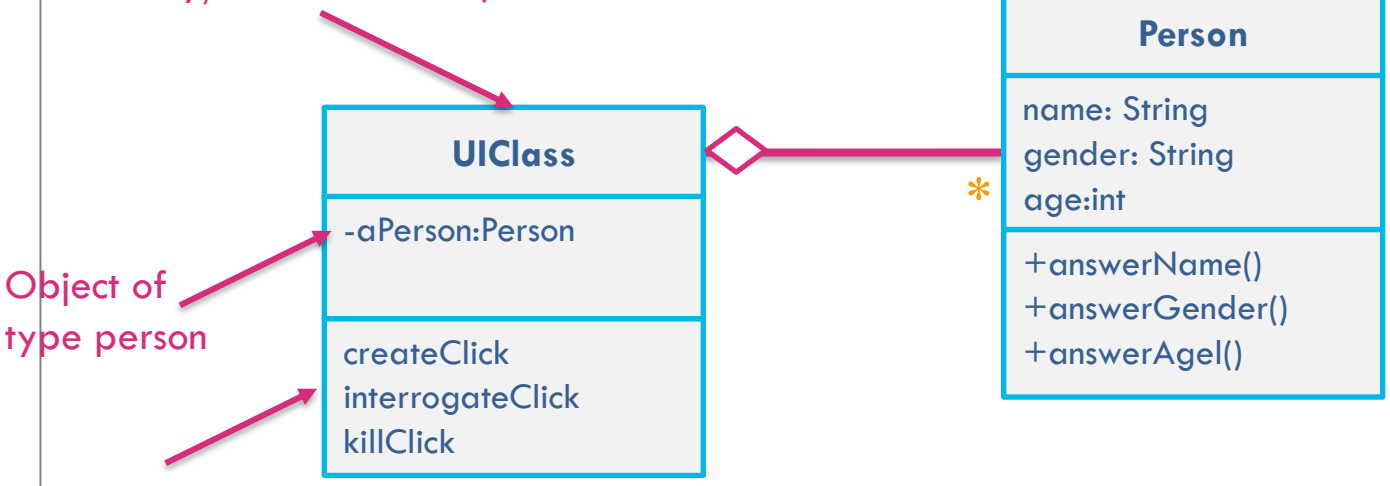


Example 1- An Easy Life
Conceptual Class Diagram

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)

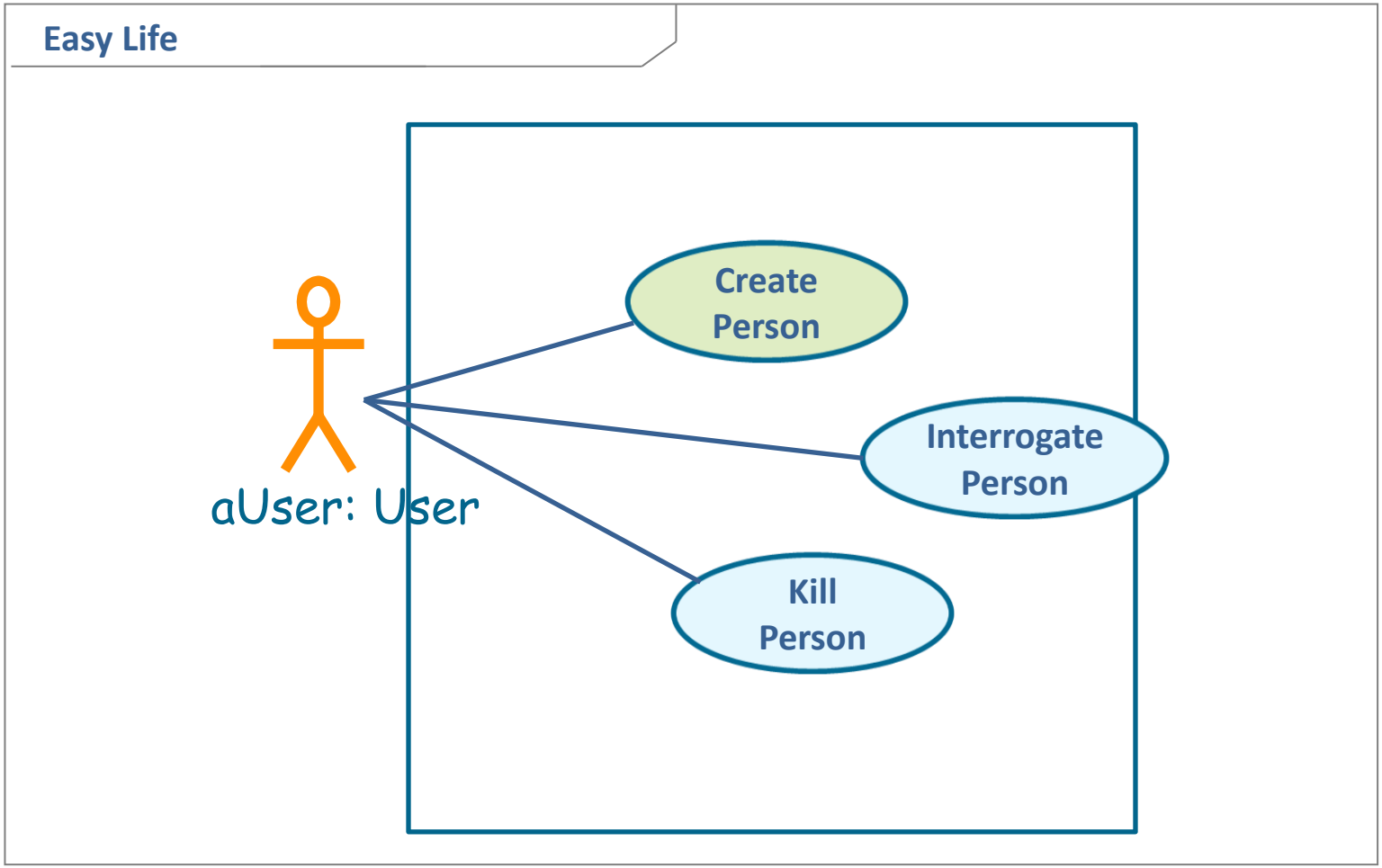
Οι κλάσεις UI ή boundary classes είναι ένας νέος τύπος κλάσης που αναγνωρίζουμε σε αυτό το σημείο καθώς προχωράμε στο σχεδιασμό.
Ένας user/actor αλληλεπιδρά πάντα με τη διεπαφή χρήστη του συστήματος και όχι απευθείας με τις κλάσεις (entity classes).

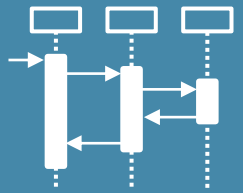


Object of type person

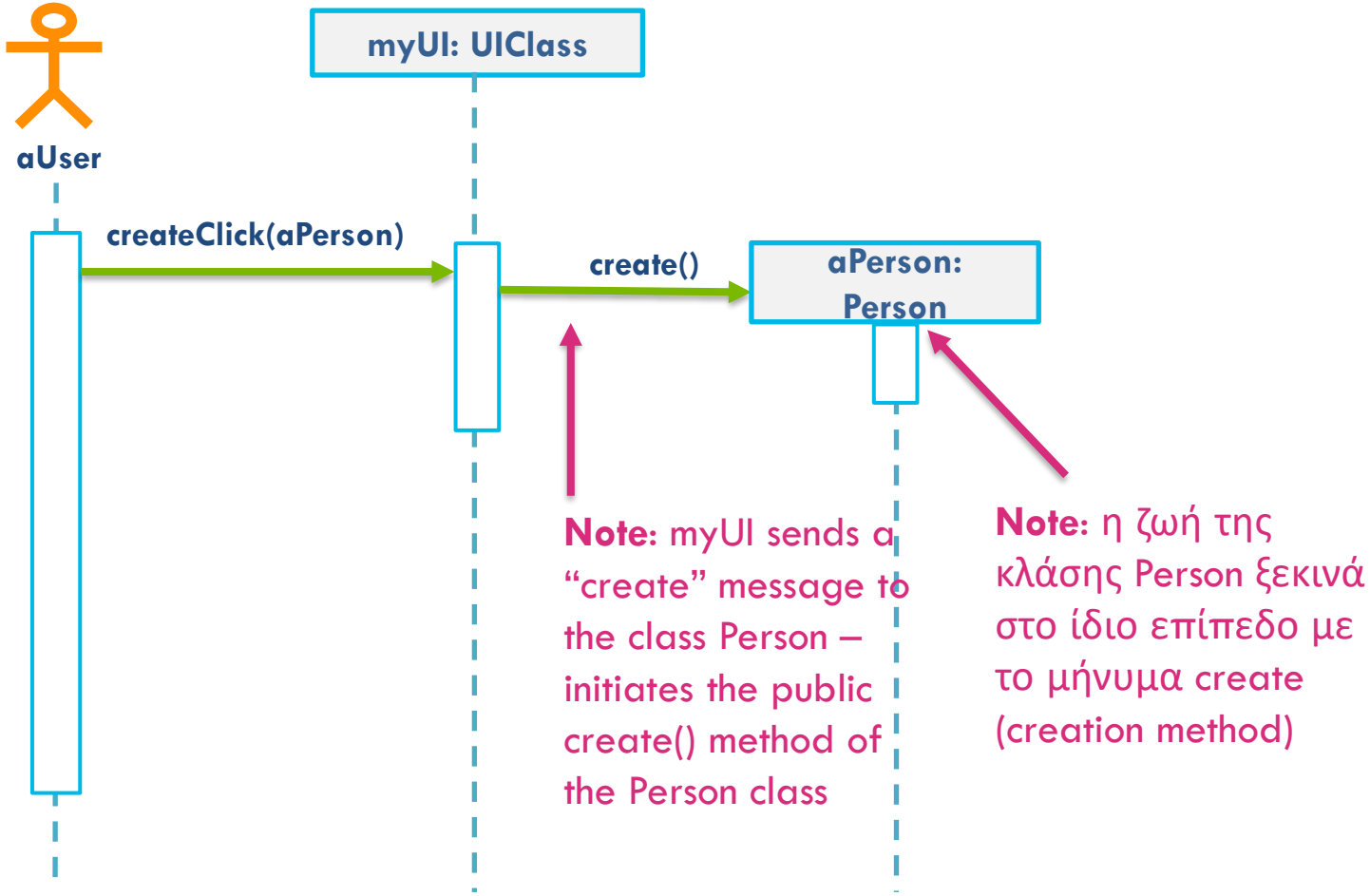
Οι κλάσεις διεπαφής (boundary classes) ανταποκρίνονται σε click methods

Example 1- An Easy Life
Use Case Diagram

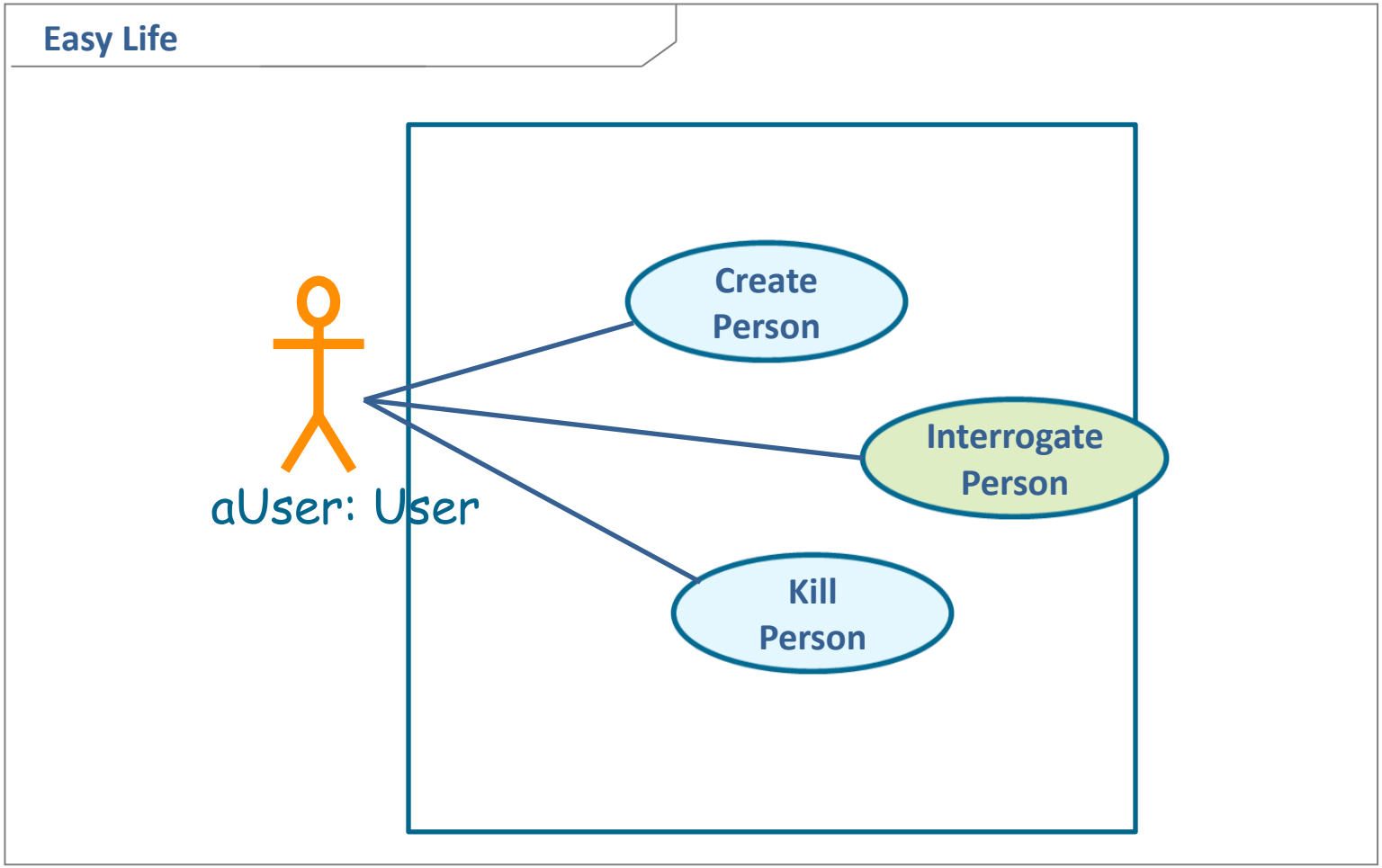




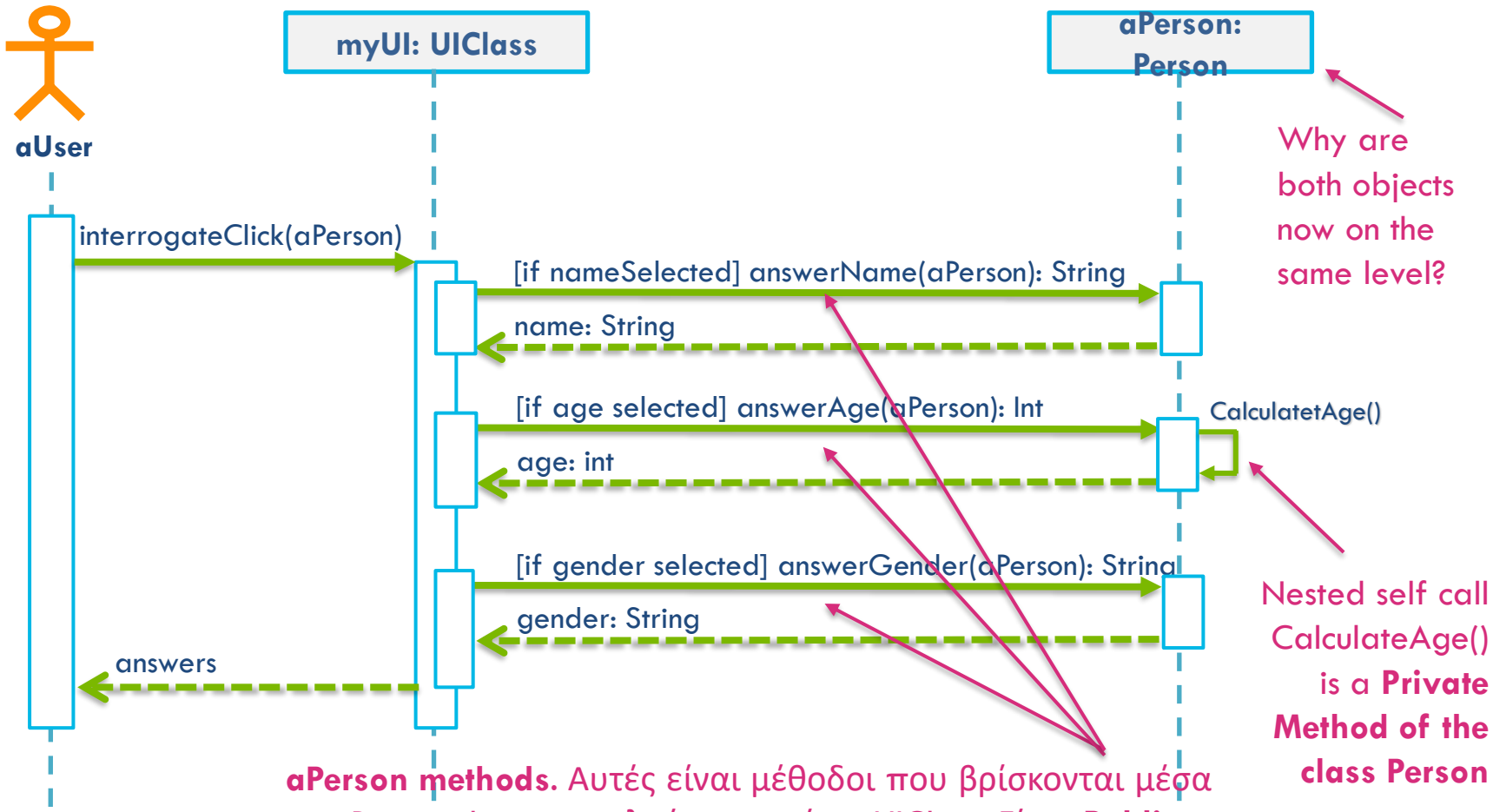
Example 1- An Easy Life
Create Person Sequence Diagram



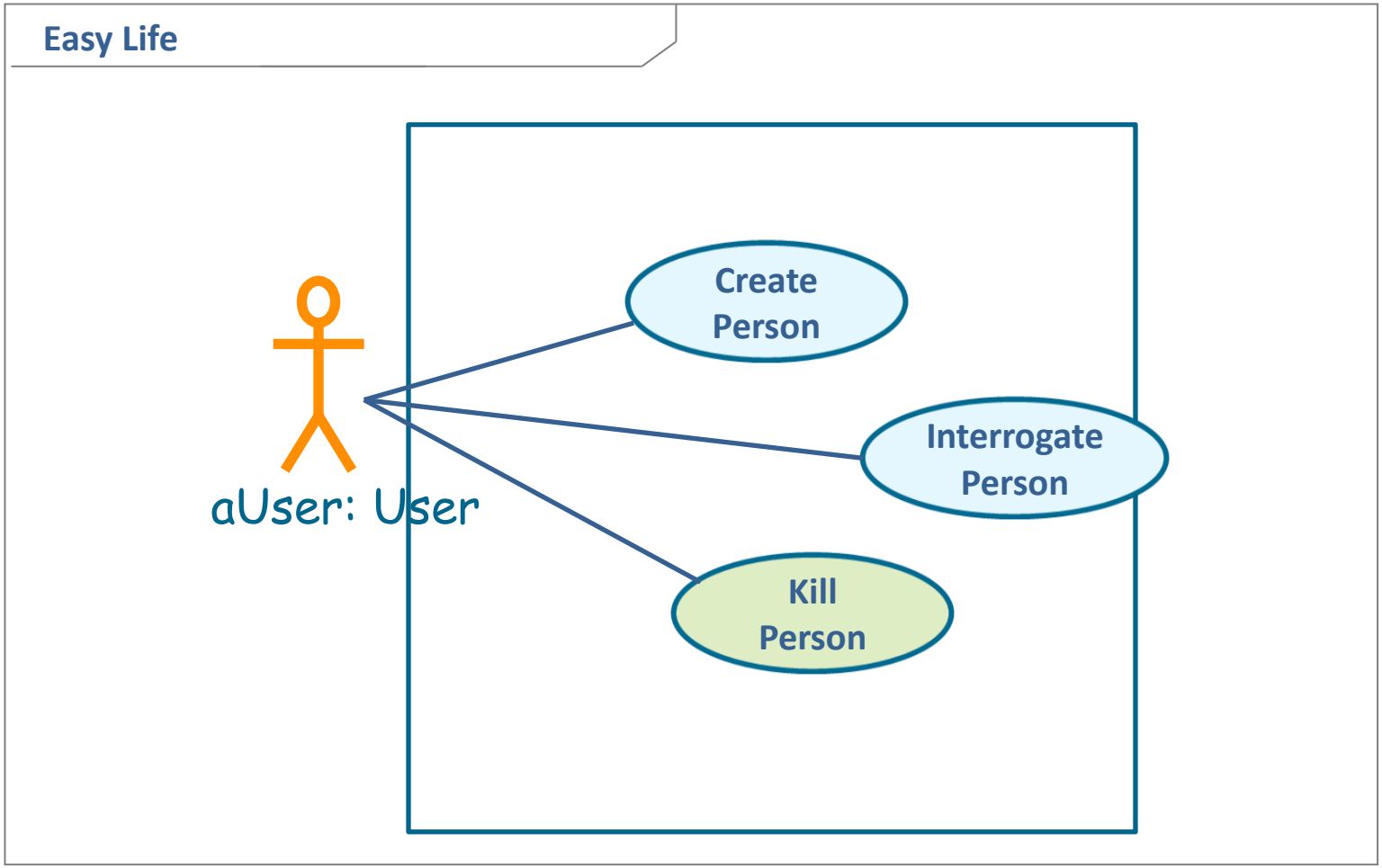
Example 1- An Easy Life
Use Case Diagram

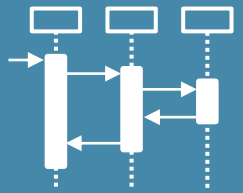


Example 1- An Easy Life
Interrogate Person Sequence Diagram

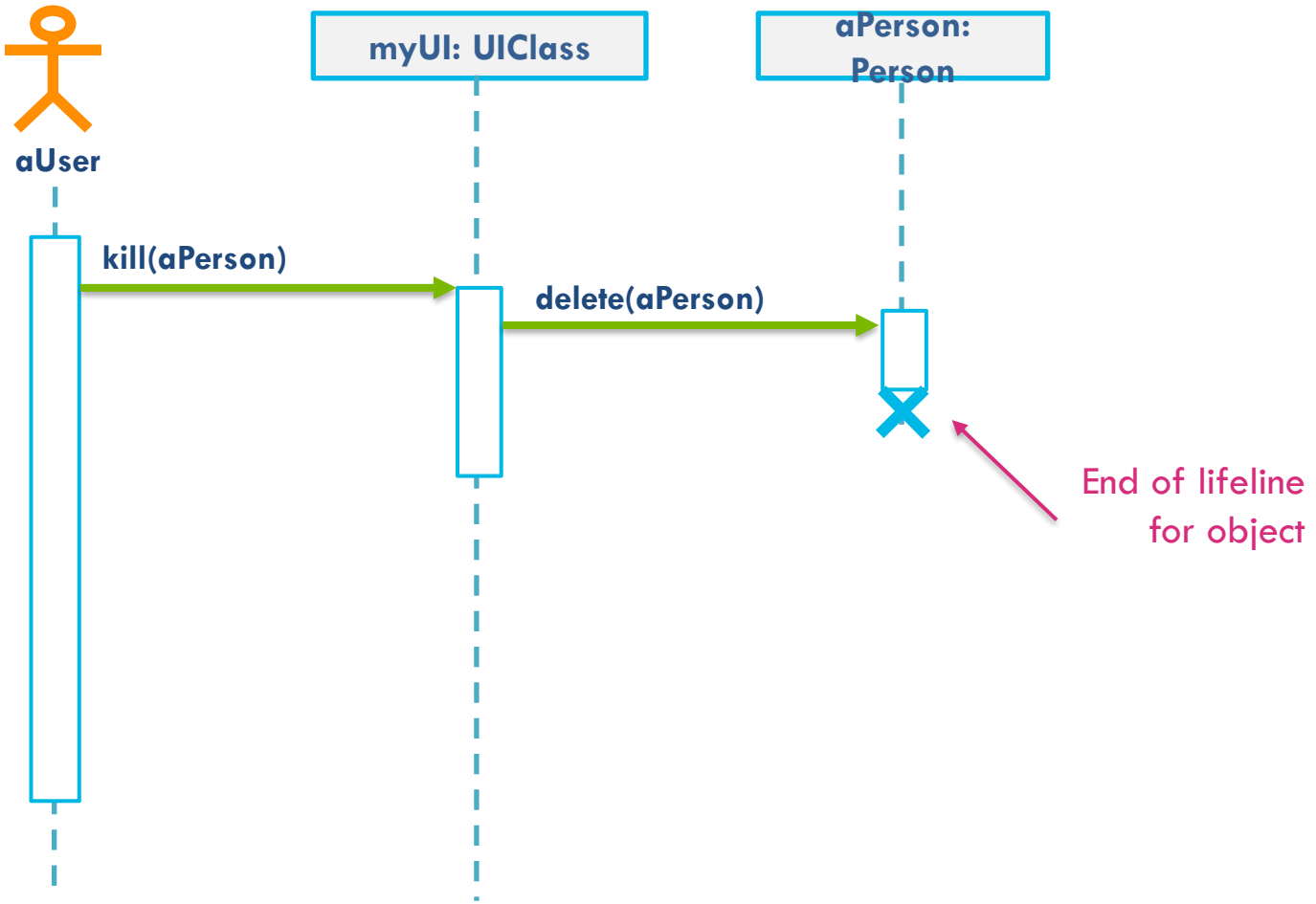


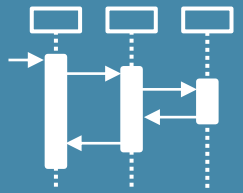
Example 1- An Easy Life
Use Case Diagram





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

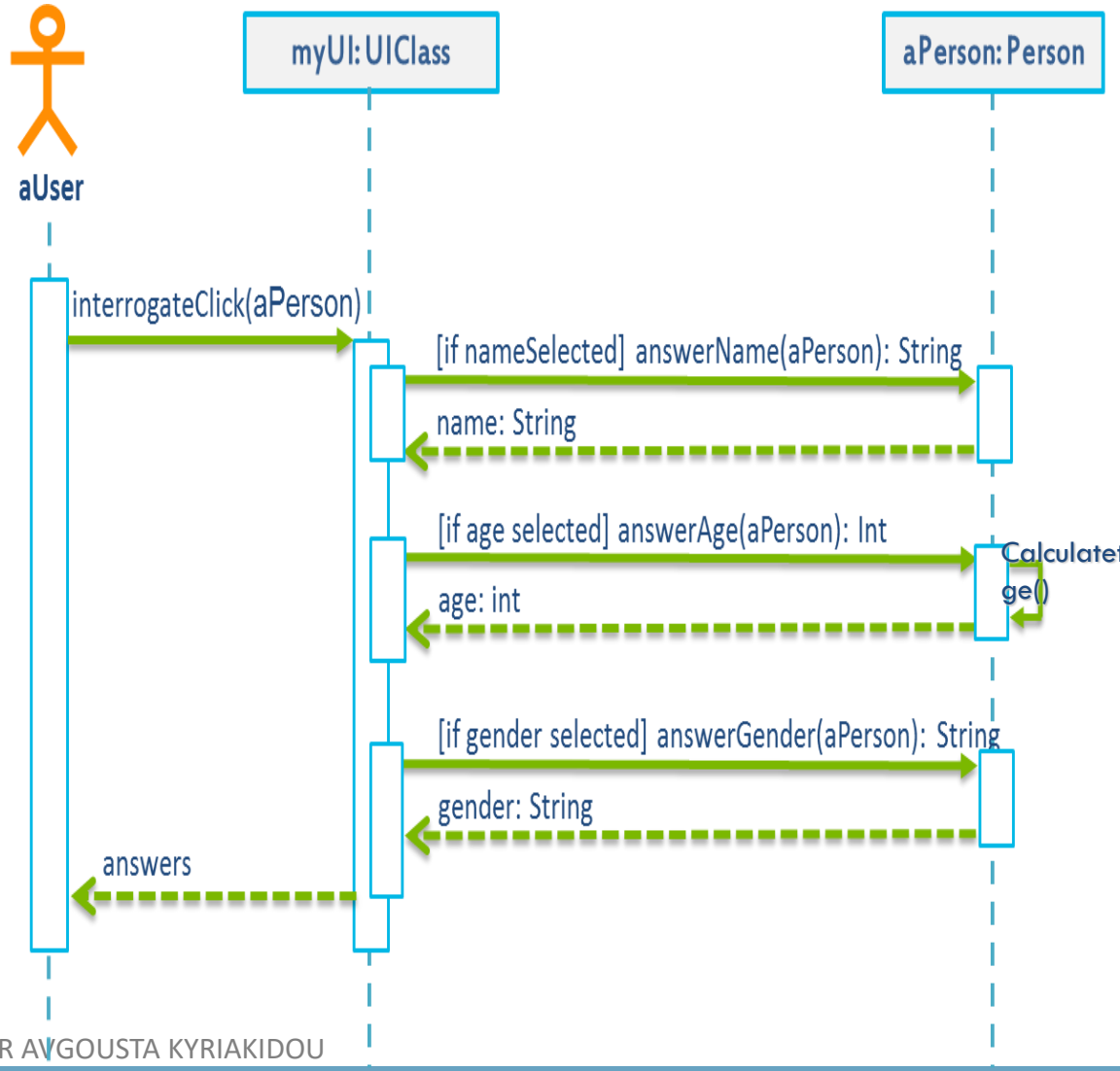
End If return age

If Question gender selected

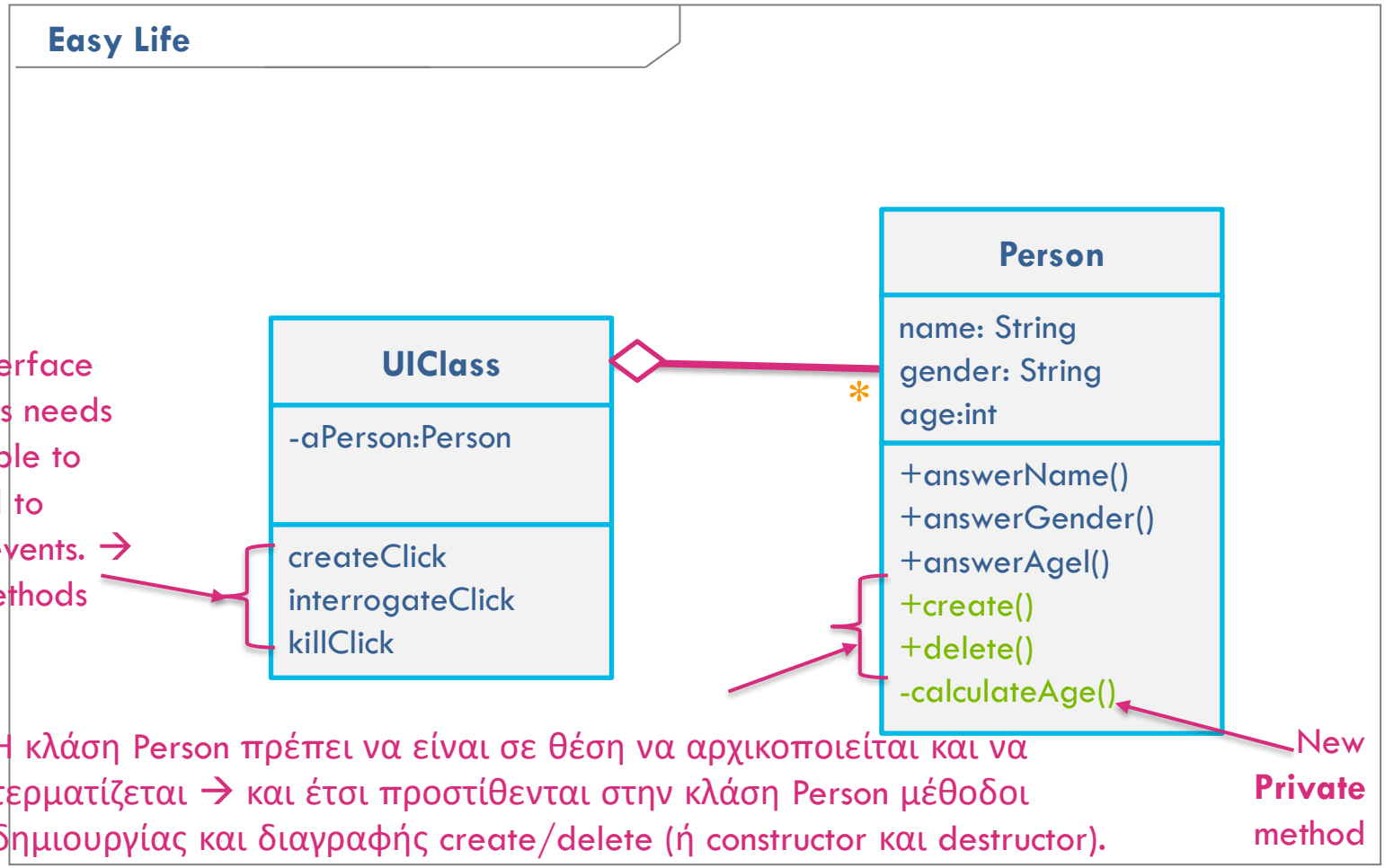
Answer gender

End If return gender

Interrogation answers returned



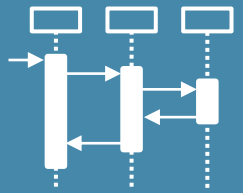
Example 1- An Easy Life
Updating Conceptual Class Diagram



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

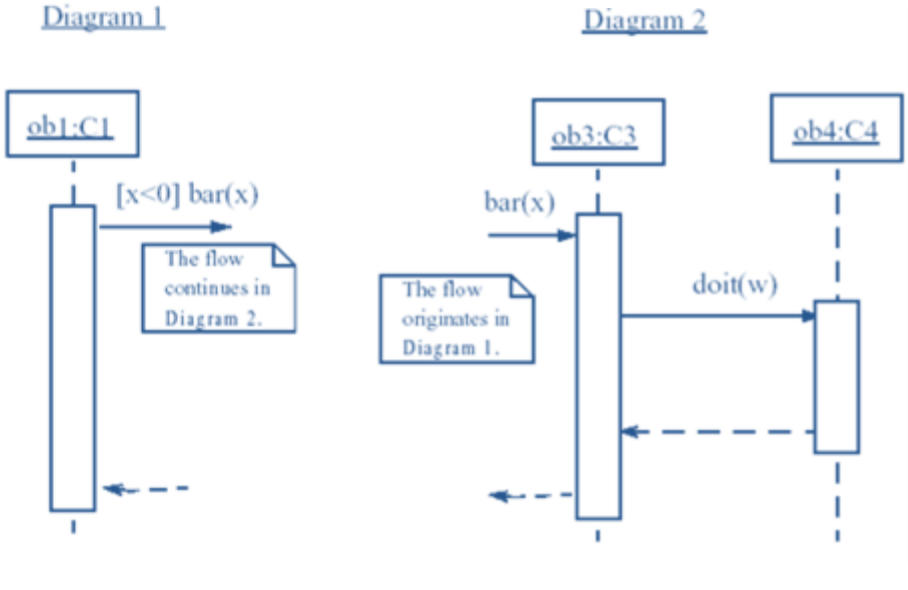
  - to model <<extends>> (to link to another <<extends>> use case)
  - to model <<includes>> (to link to another <<includes>> use case)

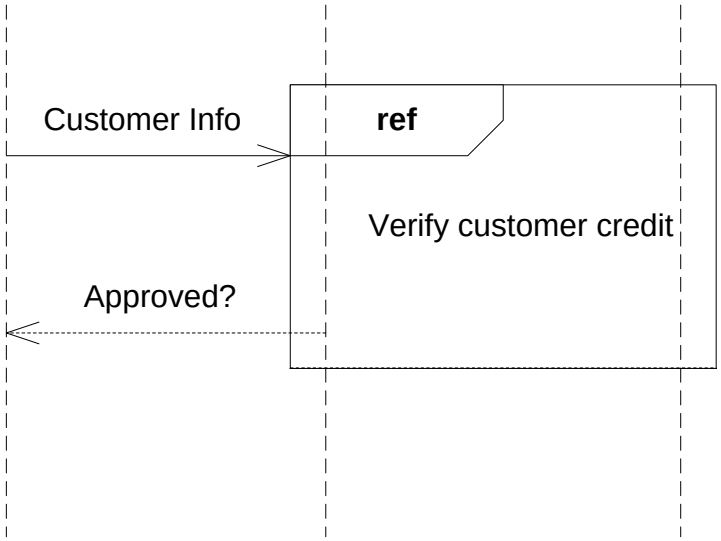
- Stereotypes

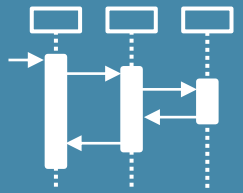
  εικονίδια που χρησιμοποιούνται για την περιγραφή κοινών τύπων αντικειμένων και actors σε λογισμικά UML

Linking Sequence Diagrams

- Αν ένα Διάγραμμα Ακολουθίας Αντικειμένων (OSD) γίνει πολύ μεγάλο ή χρειάζεται να παραπέμψει σε άλλο διάγραμμα, υποδείξτε τη σύνδεση με άλλο διάγραμμα είτε:an unfinished arrow and comment
  - με **ένα ημιτελές βέλος και σχόλιο**, είτε
  - με ένα **πλαίσιο "ref" που να ονομάζει το άλλο διάγραμμα**.

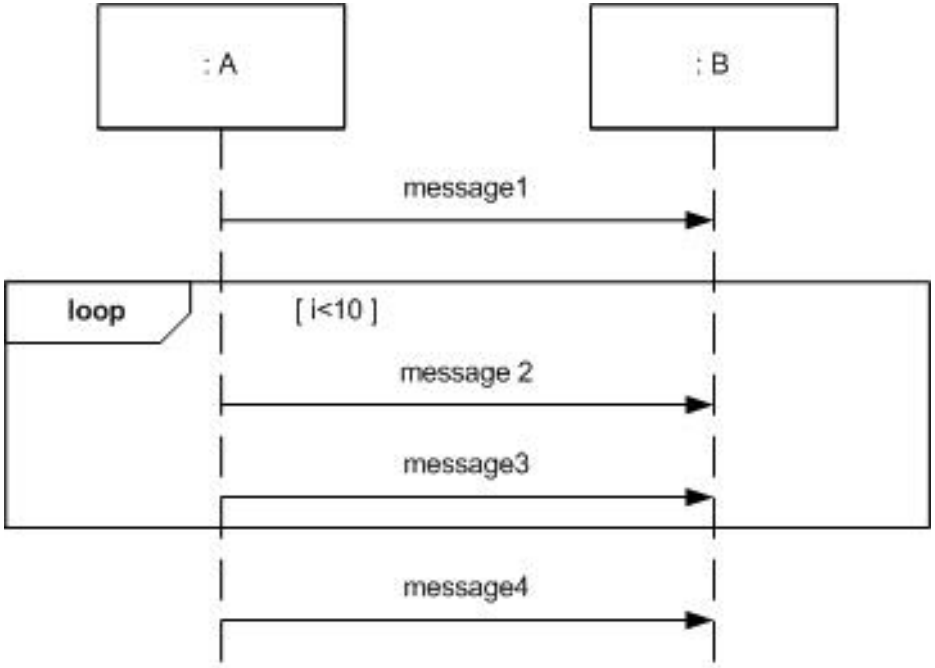






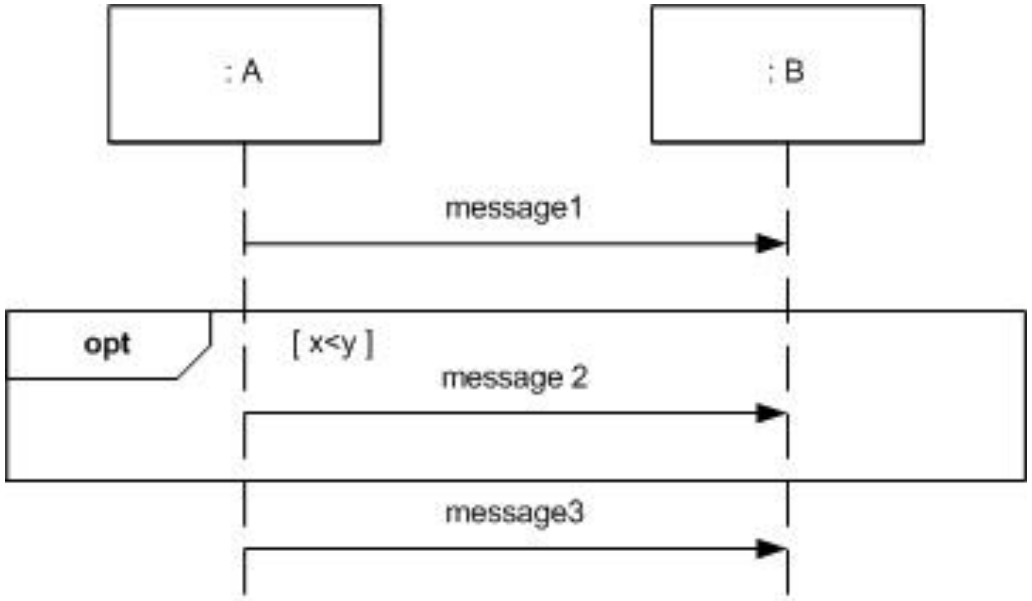
Object Sequence Diagram
Επανάληψη

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



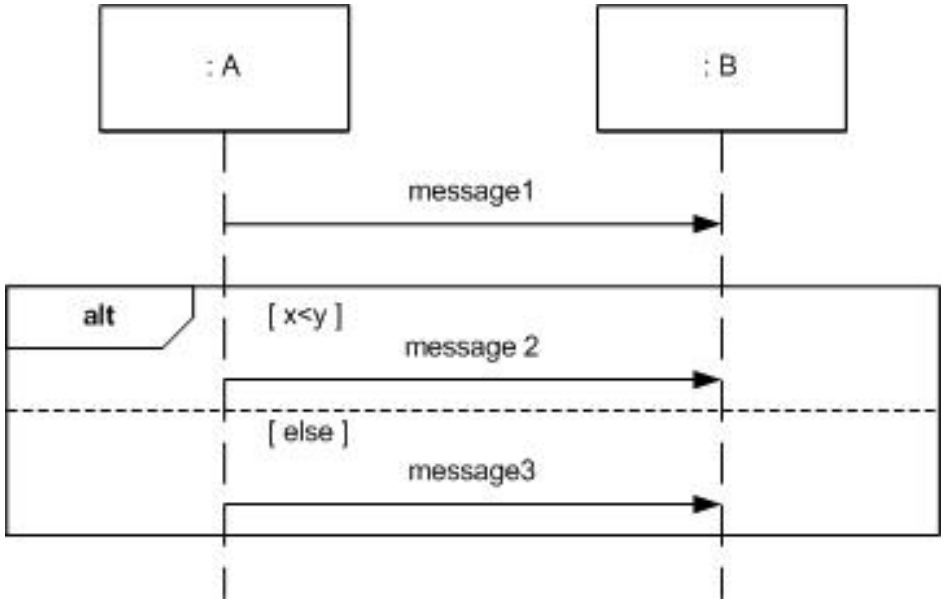
Object Sequence diagrams
Συνθήκη If

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

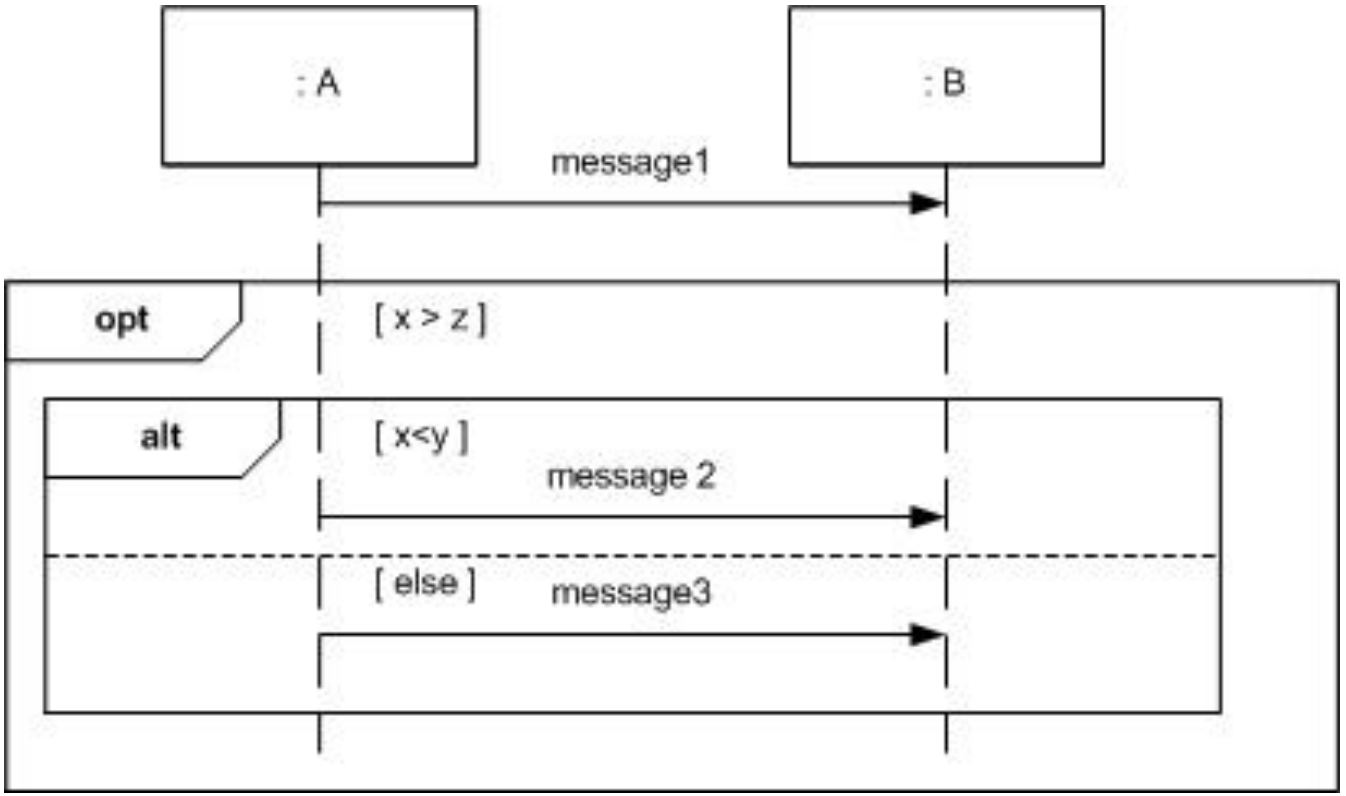


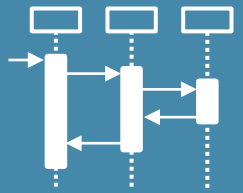
Object Sequence diagrams
Συνθήκη If- else

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



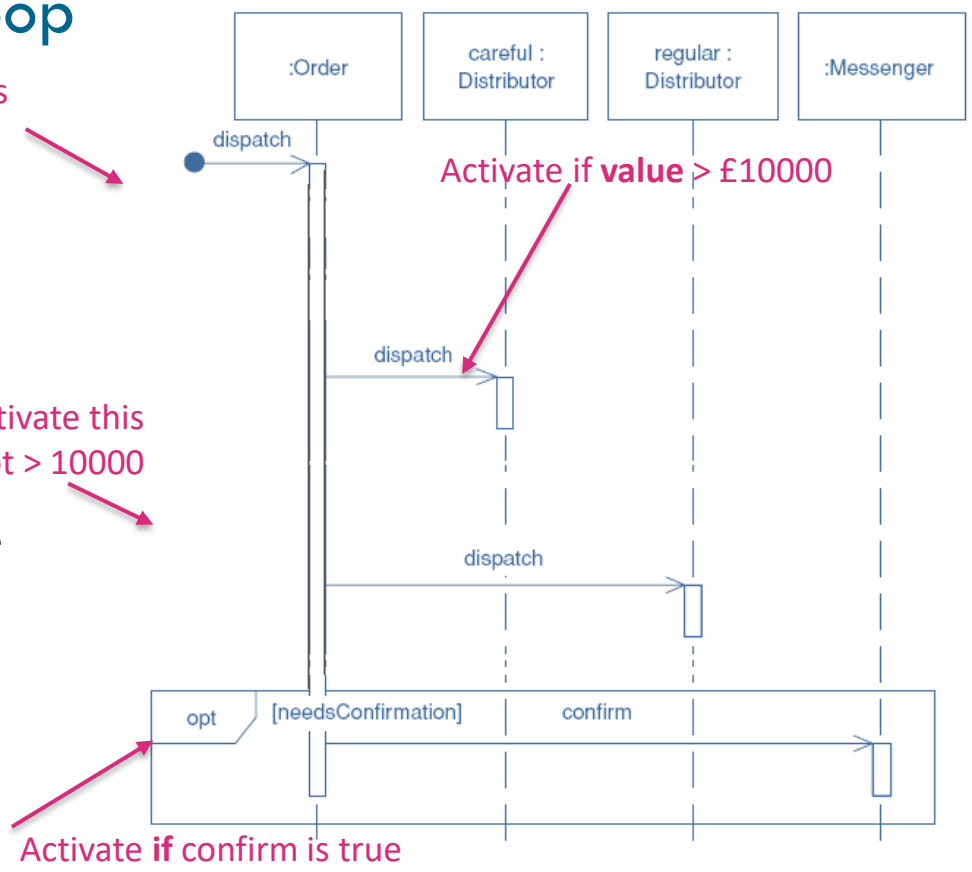
Object Sequence diagrams
περικλειόμενα πλαίσια





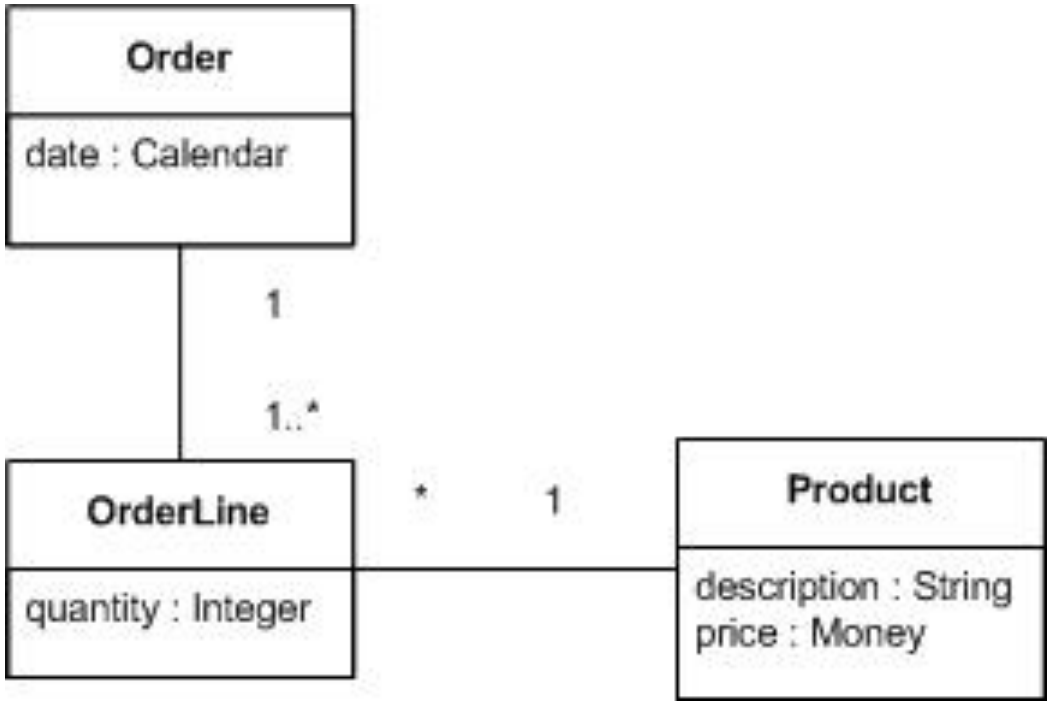
OSD - UML Notation
Indicating Selection and Loops

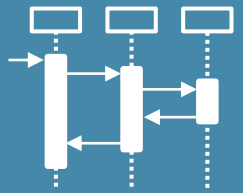
- Frame around part of a sequence diagram to indicate selection or loop
  - if
    (opt) [condition]
  - if/else
    (alt) [condition], separated by horizontal dashed line
  - loop
    (loop) [condition or items to loop over]



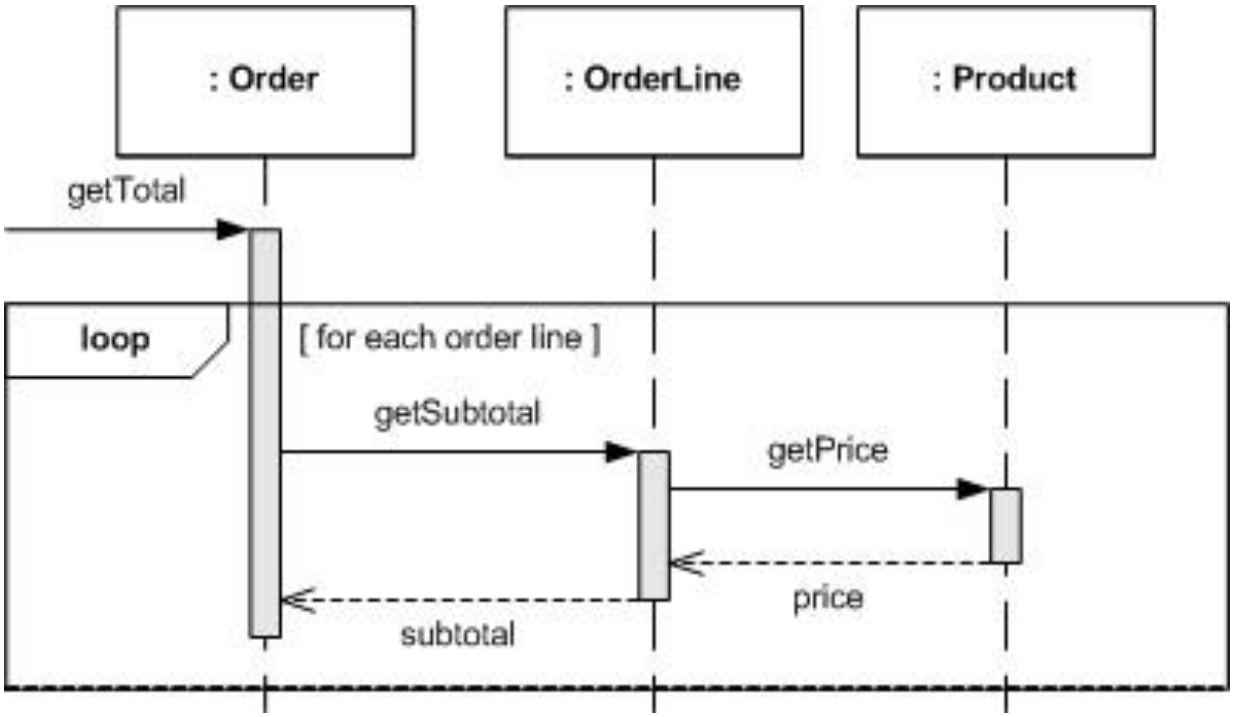
Παράδειγμα – σύστημα παραγγελίας

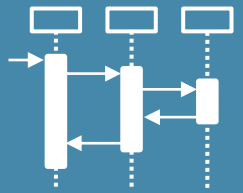
Ζητείται ένα Object sequence diagram που θα εμφανίζει την επικοινωνία των αντικειμένων για τον υπολογισμό της συνολικής αξίας της παραγγελίας





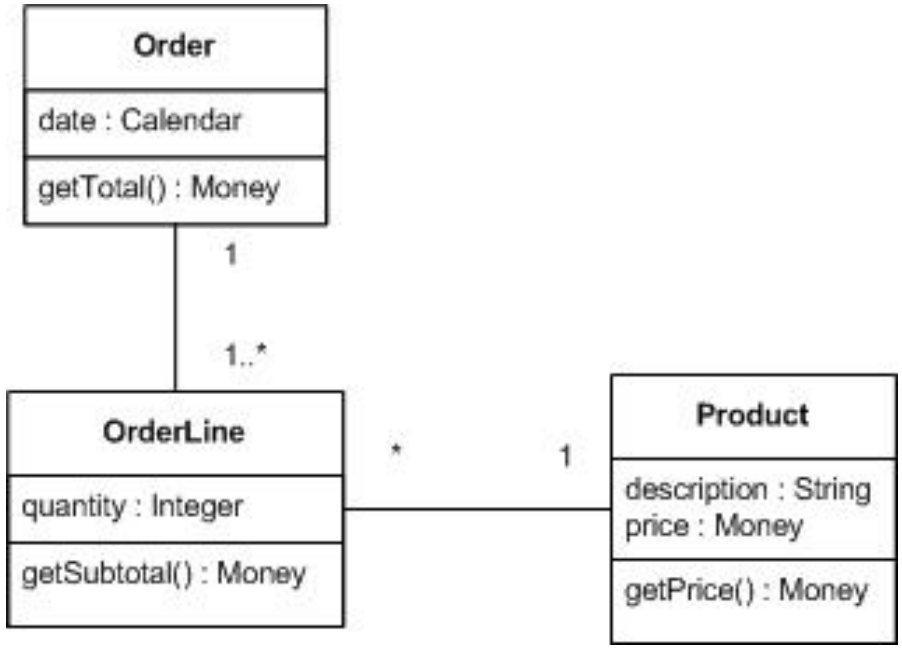
Παράδειγμα – σύστημα παραγγελίας
Object sequence diagram



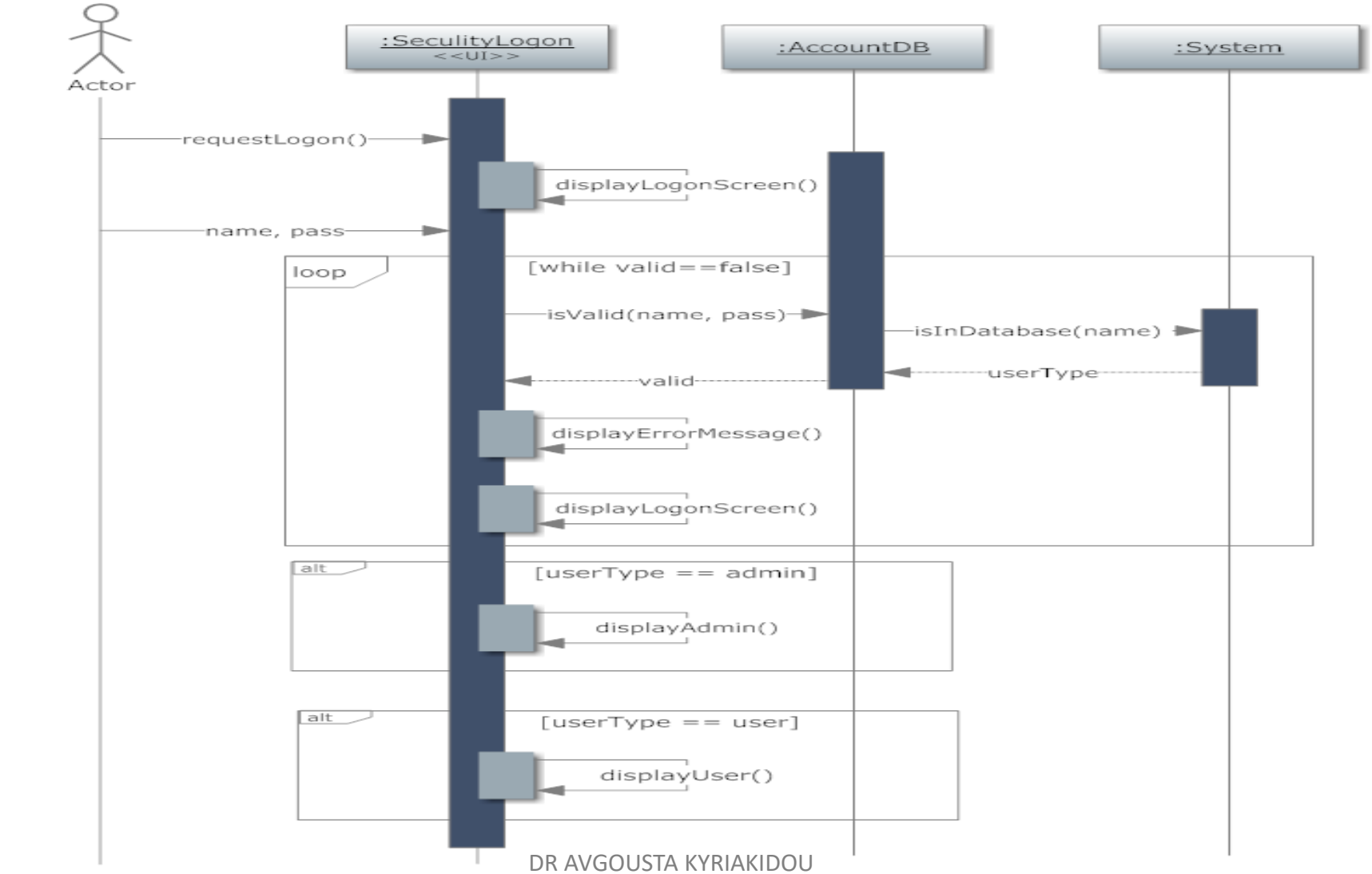


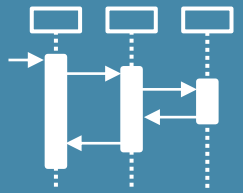
Παράδειγμα – σύστημα παραγγελίας
Ενημέρωση του διαγράμματος κλάσεων

- Η ανταλλαγή μηνυμάτων σε ένα object sequence diagram θα πρέπει να ενημερώνει και τις λειτουργίες (μεθόδους) του αντίστοιχου διαγράμματος κλάσεων



Example 2
Login Scenario

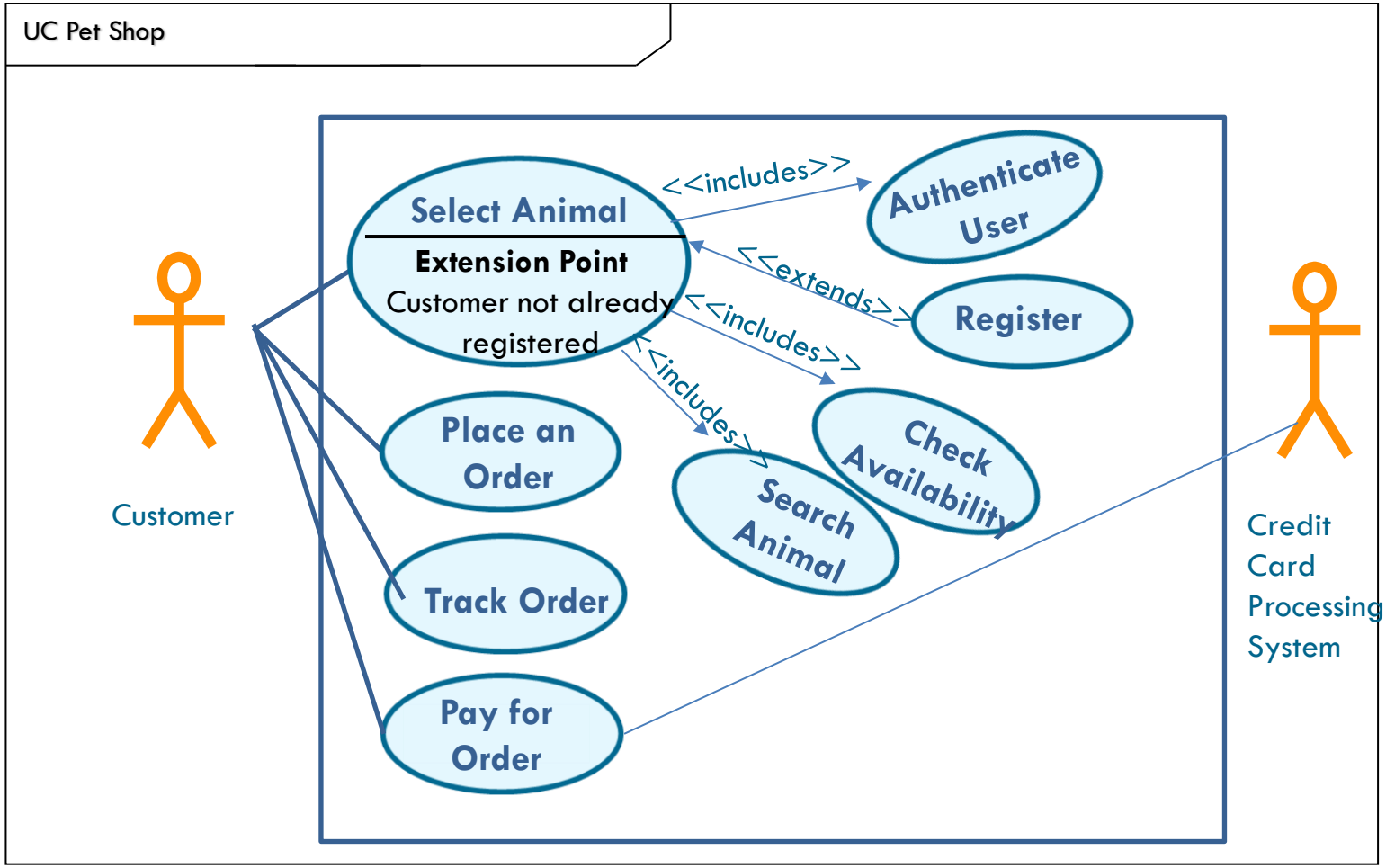




Object Sequence Diagrams
Γιατί να μην γράψουμε κώδικα και ασχολούμαστε με τα OSDs?

- Τα διαγράμματα ακολουθίας (object sequence diagrams) μπορούν να είναι κάπως κοντά στο επίπεδο κώδικα. Γιατί λοιπόν να μην κωδικοποιήσετε τον αλγόριθμο αντί να τον σχεδιάσετε ως διάγραμμα ακολουθίας;
  - ένα καλό διάγραμμα ακολουθίας εξακολουθεί να είναι πιο υψηλού επιπέδου (δεν σχεδιάζεται όλος ο κώδικας στο διάγραμμα)
  - τα διαγράμματα ακολουθίας μπορούν μετέπειτα να υλοποιηθούν σε πολλές διαφορετικές γλώσσες
  - Και οι μη προγραμματιστές μπορούν να κάνουν διαγράμματα ακολουθίας
  - μπορούν να δουν πολλά αντικείμενα/κλάσεις ταυτόχρονα στην ίδια σελίδα (οπτικό εύρος ζώνης)

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.

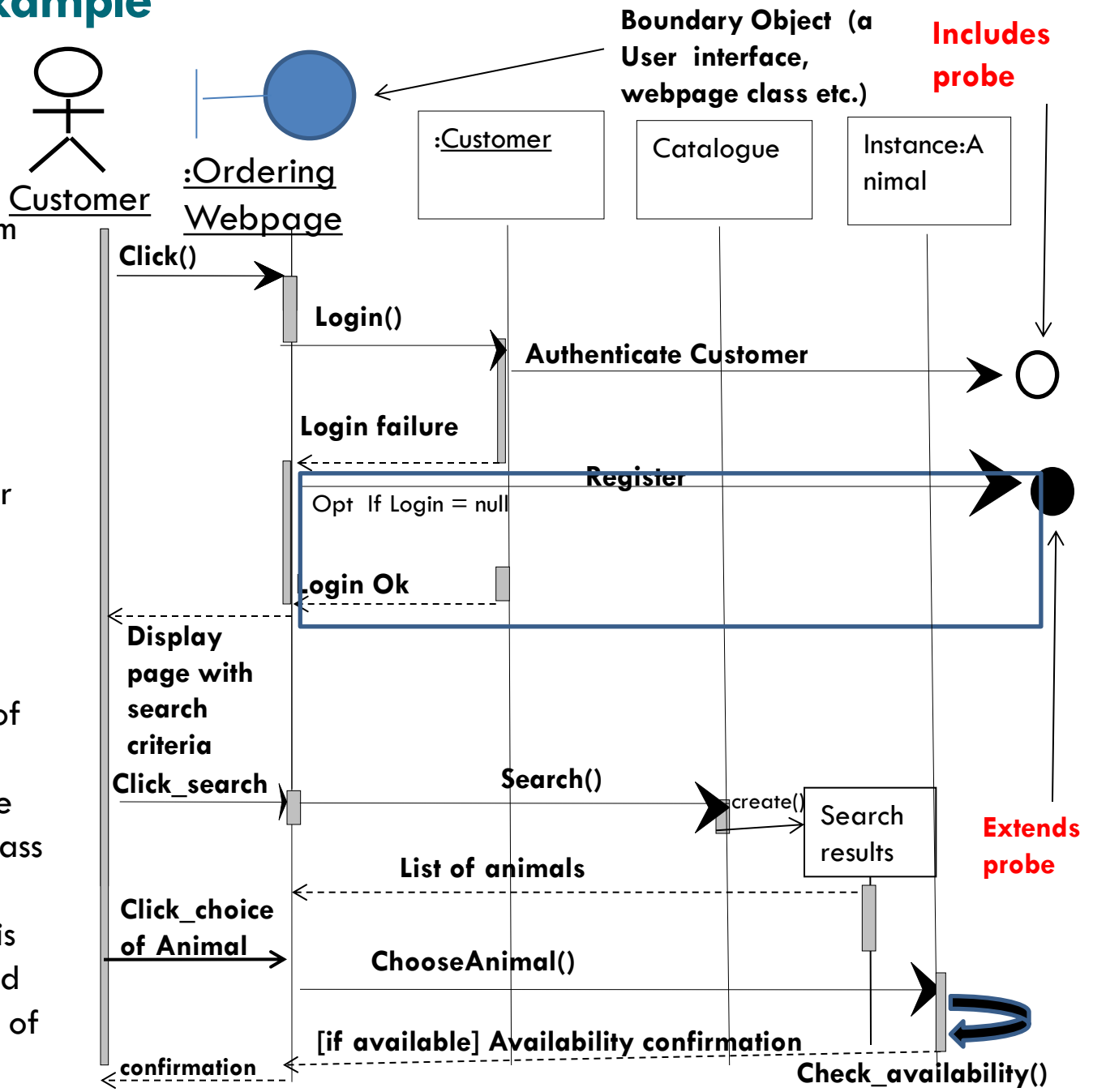
2) **OrderingWebpage:** **Boundary class** (an actor interacts with the system through a UI.

3) Classes and Objects

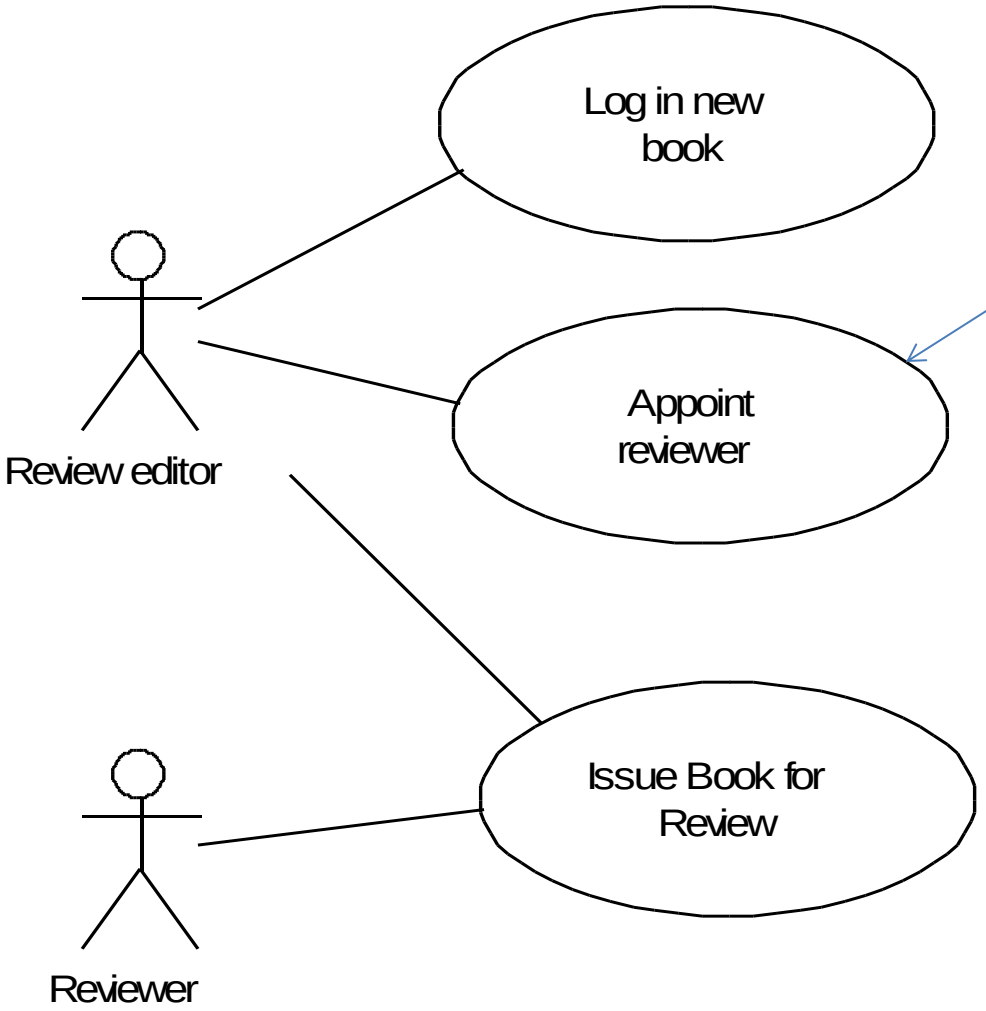
Customer = all the objects of the class Customers

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



Online System for book reviews



We will create the Object Sequence Diagram for this use case

Appoint Reviewer use case

Description

Search book

Review subject keywords

Match with reviewers

If Reviewers found

Create review

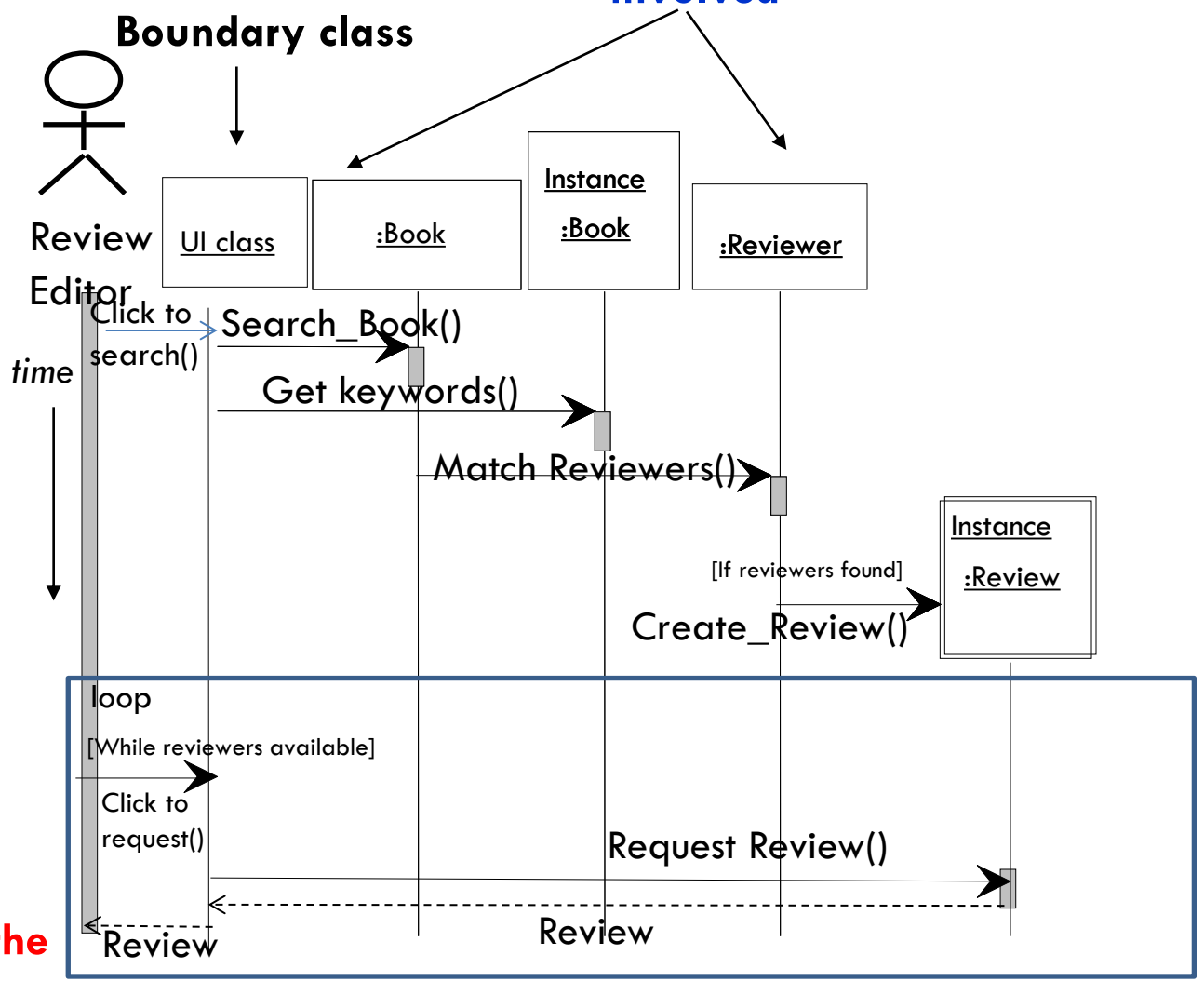
While Reviewers available

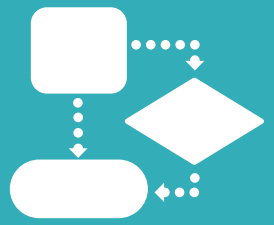
Request review

End While

End If

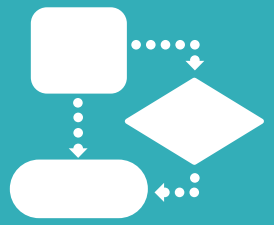
Important to provide the structured description





Review System
Some comments

- Αριστερά από το όριο του συστήματος είναι ο πραγματικός κόσμος - οι δραστηριότητες που καθοδηγούνται από τους actors και θέλουμε να μοντελοποιήσουμε.
  - Αντιγράψτε το κείμενο της περιγραφής του σεναρίου περίπτωσης χρήσης. Τοποθετήστε τα στα αριστερά του actor στο OSD - με αυτόν τον τρόπο βλέπουμε πώς το διάγραμμα ακολουθίας υλοποιεί τις περιπτώσεις χρήσης.
- Στα δεξιά είναι τα αντικείμενα/κλάσεις που περιμένουμε να συνεργαστούν μαζί για να γίνει μια συγκεκριμένη εργασία.
- Η δημιουργία των διαγραμμάτων είναι ένα **incremental and iterative activity**. Πάντα πάμε πίσω στα διαγράμματα περιπτώσεων χρήσης και στο διάγραμμα κλάσεων.
- Οι κλάσεις στο OSD θα πρέπει να είναι κλάσεις στο διάγραμμα κλάσεων. Μηνύματα μπορούν να ανταλλάσσονται μόνο μεταξύ κλάσεων που συνδέονται/σχετίζονται στο διάγραμμα κλάσεων.



Review System
Some comments

- Κάθε ένα από τα μηνύματα που εμφανίζονται στο διάγραμμα (Request review(), GetKeywords(), MatchReviewers() κ.λπ.) αποτελεί ένα έναυσμα για μια λειτουργία/μέθοδο που ορίζεται στην κατάλληλη κλάση.
  - Όπου δείχνει το βέλος, αυτή είναι η κλάση που περιλαμβάνει τη μέθοδο π.χ. η κλάση Book έχει μια μέθοδο μια δημόσια μέθοδο Search Book()

- Η ανάπτυξη ενός OSD είναι ένας κύριος τρόπος για να αρχίσετε να ορίζετε τις μεθόδους για τις κλάσεις που υπάρχουν στο conceptual class diagram, ώστε να μπορείτε να προχωρήσετε στο σχεδιασμό του συστήματος και το design class diagram.

- Τα κουτιά αναπαριστούν όλα τα αντικείμενα μιας κλάσης ή ένα συγκεκριμένο αντικείμενο.
  - **Ο τίτλος δείχνει ποια κλάση. Μπροστά από το ονομα της κλάσης μπορει να οριστεί το συγκεκριμένο αντικείμενο.**
  - ***(e.g. Instance: Review → meaning this is the single chosen object of the class Review.)***

- Έχουμε δείξει την κλάση Βιβλίο δύο φορές - μία φορά ως κλάση (π.χ. όλα τα βιβλία στο σύστημα), και μία φορά ως αντικείμενο το μοναδικό βιβλίο με το οποίο ασχολούμαστε. Αυτό βοηθάει καλύτερα στην κατανόηση.

Search for Book use case
Object sequence diagram

Search Book : Use Case

- Main scenario -

The Customer specifies an author on the Search Page and then presses the Search button.

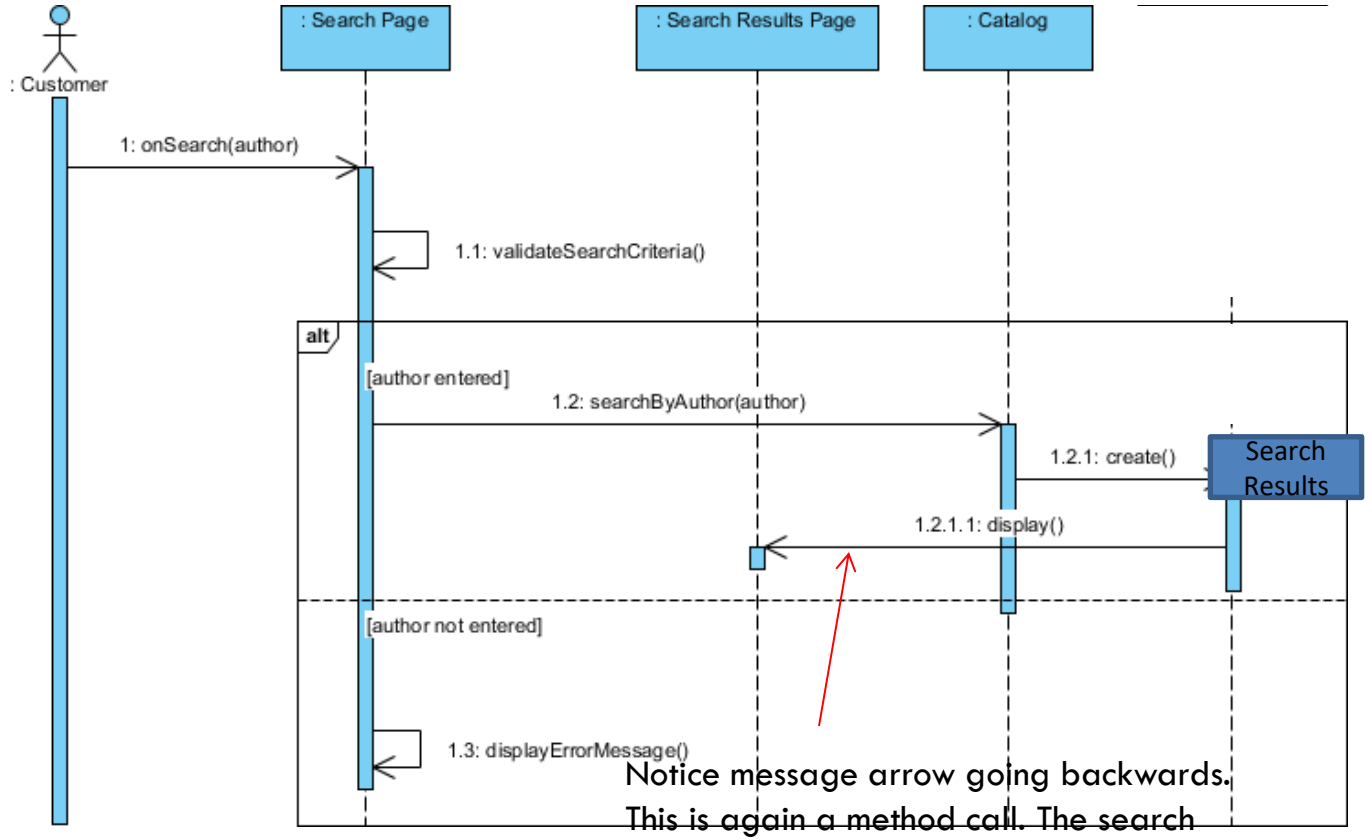
The system validates the Customer's search criteria.

If author is entered, the System searches the Catalog for books associated with the specified author.

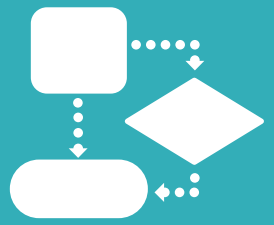
When the search is complete, the system displays the search results on the Search Results page.

- Alternate path -

If the Customer did not enter the name of an author before pressing the Search button, the System displays an error message



Notice message arrow going backwards. This is again a method call. The search results UI class should be presented after the catalogue.

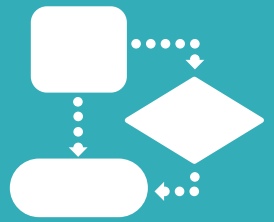


What does an OSD tells us?

- **Εάν ένα OSD γίνεται πολύ περίπλοκο, είναι πιθανό να οφείλεται στο ότι η περίπτωση χρήσης σας είναι πολύ περίπλοκη.**
  - Επιστρέψτε στο διάγραμμα της περίπτωσης χρήσης σας και εξετάστε αν η εν λόγω περίπτωση χρήσης μπορεί να απλοποιηθεί χρησιμοποιώντας <<extends>> and <<includes>>.

- Κατά την ανάπτυξη ενός OSD, μπορεί να χρειαστεί να **εισαγάγετε νέες κλάσεις.**
  - **Σκοπός του OSD να εντοπίσει τυχόν αντικείμενα που λείπουν (π.χ. boundary and control classes become more apparent at this stage).**
  - **Για παράδειγμα, ο actor στέλνει πάντα το πρώτο μήνυμα σε ένα boundary object (a user interface class).**
  - **Boundary or Control Objects** εξετάζονται λεπτομερέστερα καθώς προχωράμε περισσότερο στη σχεδίαση.

Οι μέθοδοι για κάθε κλάση είναι εύκολο να αναγνωριστούν με τη μεταβίβαση μηνυμάτων μεταξύ των αντικειμένων.



How to build an Object Sequence diagram?

- **Step 1: Αποφασίστε την περίπτωση χρήσης που θα θέλατε να αναπτύξετε περαιτέρω ως OSD.**
  - Αντιγράψτε το σενάριο της περίπτωσης χρήσης στην αριστερή πλευρά του OSD σας.
- **Step 2:** Καθορίστε όλες τις κλάσεις που εμπλέκονται στο συγκεκριμένο use case scenario, ελέγξτε το διάγραμμα κλάσεων για να δείτε πώς αυτές σχετίζονται μεταξύ τους και συμπεριλάβετε τις κλάσεις αυτές ως ορθογώνια πλαίσια στην επάνω δεξιά πλευρά του OSD σας.
  - Σχεδιάστε τις γραμμές ζωής τους ως διακεκομμένες γραμμές.
- **Step 3:** Καθορίστε το μήνυμα που ξεκινά την ακολουθία.
  - Αυτό είναι γενικά ένα μήνυμα που αποστέλλεται από τον χρήστη (the external actor initiating the use case) σε ένα boundary object (UI Class).
- **Step 4:** Καθορίστε τα επόμενα μηνύματα.
  - Στην πρώτη περίπτωση, αυτό περιλαμβάνει τη διερεύνηση του τι θα κάνει το boundary object μετά που ο χρήστης ξεκινά την εκτέλεση του Use case και της ακολουθίας.
  - Προσδιορίστε τις μεθόδους από το CONCEPTUAL class diagram.
- **Step 5:** Καθορίστε τυχόν μηνύματα επιστροφής

Recap
learning check…

- Object Sequence Diagrams are used to describe the dynamic behaviour of the system-to-be
- UML Notation
- Use-case Diagrams and Conceptual Class Diagrams are used to construct Object Sequence Diagrams
- OSDs help identify missing classes and behaviour and are used to modify the Conceptual Class Diagram and classes to reflect extended understanding of use-case scenarios
- … another step closer to Design Class Diagrams.

Next week: Object State Transition Diagrams

HOW DID WE DO THIS WEEK?
WHAT DID WE LEARN?

Reply at:
PollEv.com/avgoustakyri977

1. Go to **PollEv.com**
2. Enter **AVGOUSTAKYRI977**
3. Respond to activity

1. Text **AVGOUSTAKYRI977** to **22333**
2. Text in your message

Further Reading

1. Kung, D.C (2024) Software Engineering, 2nd Edition, McGraw Hill – Chapter 9

2. Pressman, R.S. (2020). Software Engineering: A practitioner's approach, 9th Edition, Mc-Graw Hill – Chapter 8