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

Interaction Diagrams

Περιεχόμενα και Διάρθρωση

- Εισαγωγή
- Διαγράμματα αλληλουχίας
- Διαγράμματα επικοινωνίας
- Διαγράμματα κατάστασης
- Κανόνες σχεδίασης διαγραμμάτων συμπεριφοράς
- Παραδείγματα χρήσης διαγραμμάτων αλληλεπίδρασης και κατάστασης
- Υλοποίηση διαγραμμάτων κατάστασης σε C++
- Υλοποίηση με χρήση Προτύπου Σχεδίασης Καταστάσεων

Μοντελοποίηση στην Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων

- Λειτουργική Μοντελοποίηση
  - Περιπτώσεων Χρήσης (Use Case), Δραστηριοτήτων (Activity)
- Δομική Μοντελοποίηση
  - Κλάσεων (Class) , Αντικειμένων (Object), Πακέτων (Package), Παράταξης (Deployment) , Εξαρτημάτων (Component) , Σύνθετης Δομής (Composite Structure)
- Μοντελοποίηση Συμπεριφοράς
  - Sequence (Αλληλουχίας), Επικοινωνίας (Communication), Καταστάσεων (State), Χρονισμού (Timing), Interaction Overview, Protocol State Machine

Διαγράμματα αλληλεπίδρασης (Interaction Diagrams)

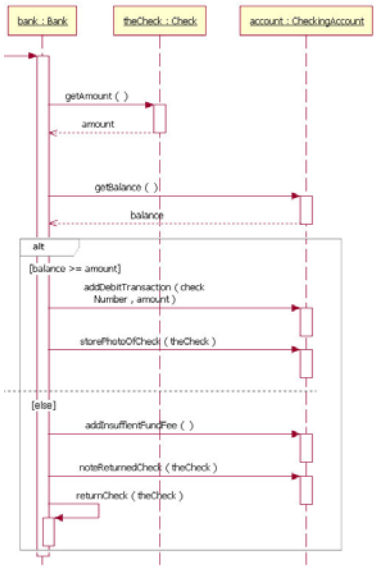
- Περιγράφουν το πώς αλληλεπιδρούν τα διάφορα αντικείμενα μεταξύ τους για την εκπλήρωση κάποιου στόχου
- Ένα διάγραμμα αλληλεπίδρασης στην ουσία συλλαμβάνει την συμπεριφορά μιας περίπτωσης χρήσης
  - Ποια μηνύματα ανταλλάσσονται κατά την εκτέλεση μιας περίπτωσης χρήσης μεταξύ των αντικειμένων που συμμετέχουν και με ποια χρονική σειρά.
- Η UML ορίζει δύο είδη διαγραμμάτων αλληλεπίδρασης:
  - ***Διαγράμματα αλληλουχίας (Sequence diagrams):*** *Παρουσιάζουν τις αλληλοεπιδράσεις μεταξύ αντικειμένων με έμφαση στη χρονική σειρά των ενεργειών που συμβαίνουν*
  - ***Διαγράμματα επικοινωνίας (Communication diagrams):*** Παρουσιάζουν τις αλληλοεπιδράσεις και την επικοινωνία μεταξύ αντικειμένων με έμφαση στις ενέργειες που συμβαίνουν

Διαγράμματα αλληλουχίας (Sequence diagrams)

- Τα βασικά στοιχεία που ορίζονται στα διαγράμματα αλληλουχίας είναι:
  - Αντικείμενα τα οποία έχουν κάποιο κύκλο ζωής (Lifelines)
  - Μηνύματα που ανταλλάσσονται μεταξύ των αντικειμένων (Messages)
  - Συνθήκες αποστολής μηνυμάτων μεταξύ των αντικειμένων (Guards)
  - Στοιχεία ομαδοποίησης μηνυμάτων (Combined Fragments) – UML2
    - **Εναλλακτικές λύσεις (Alternatives):** χρησιμοποιείται για να μοντελοποιήσει τη διαδικασία επιλογής μεταξύ δύο ή περισσότερων ακολουθιών από μηνύματα (**if then else**)
    - **Επιλογή (Option):** χρησιμοποιείται για να μοντελοποιήσει τη διαδικασία επιλογής εκτέλεσης μιας ακολουθίας μηνυμάτων με βάση κάποια συνθήκη (**if then**)
    - **Βρόγχοι (Loops):** χρησιμοποιείται για να μοντελοποιήσει τη διαδικασία επαναληπτικής εκτέλεσης μιας ακολουθίας μηνυμάτων με βάση κάποια συνθήκη (**while**)

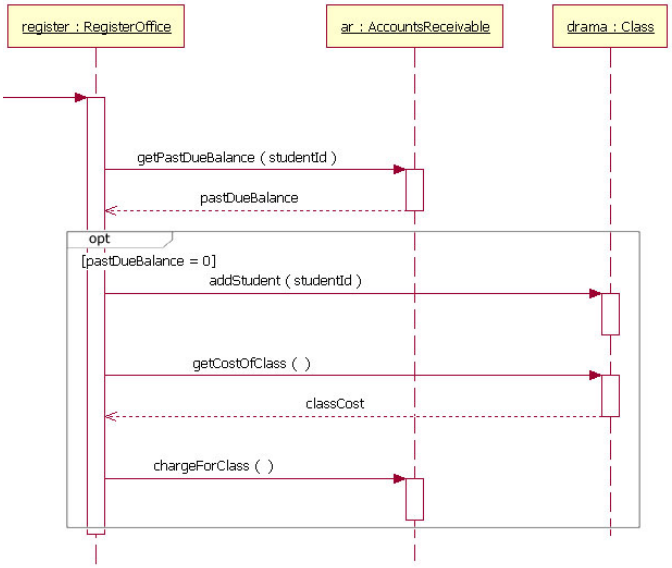
Παράδειγμα χρήσης εναλλακτικών λύσεων (alt)

- Διάγραμμα αλληλουχίας για την περίπτωση χρήσης «**Εξόφληση επιταγής**»
- Η τράπεζα ελέγχει αν τα χρήματα στον λογαριασμό επαρκούν οπότε και προχωράει στη δέσμευση των χρημάτων
- Σε διαφορετική περίπτωση επιστρέφει πίσω την επιταγή



Παράδειγμα χρήσης επιλογής (opt)

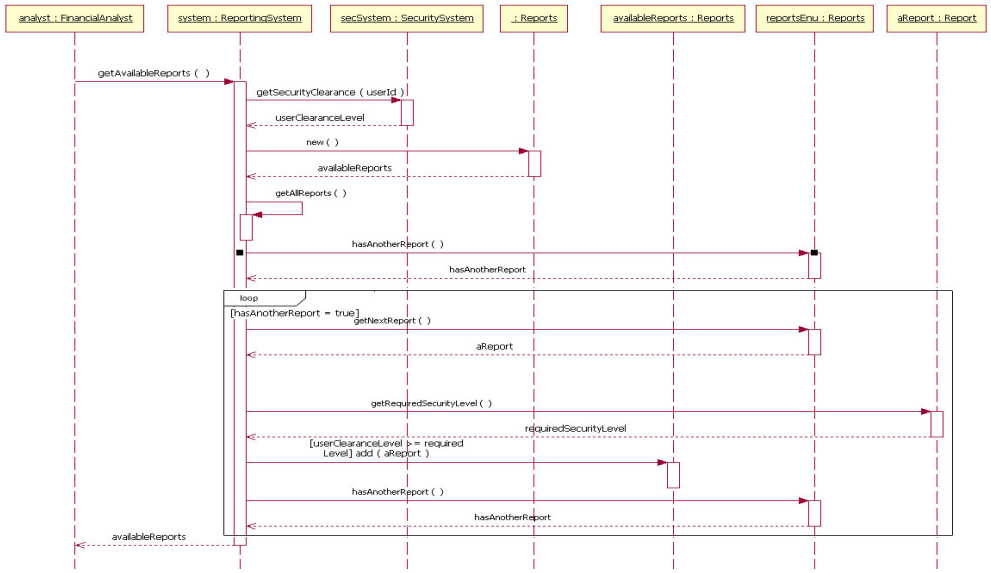
- Διάγραμμα αλληλουχίας για την περίπτωση χρήσης «**Χρέωση και εγγραφή σε τάξη**»
- Ελέγχεται αν ο μαθητής χρωστάει χρήματα από περασμένα χρόνια
- Αν όχι τότε εγγράφεται στην τάξη και χρεώνεται για την εγγραφή του.



Παράδειγμα χρήσης βρόγχων (loop)

- Διάγραμμα αλληλουχίας για την περίπτωση χρήσης «**Λήψη διαθέσιμων αναφορών**»
- Αρχικά γίνεται πιστοποίηση της ταυτότητας του αναλυτή
- Αν έχει εξουσιοδότηση τότε γίνεται η λήψη όλων των διαθέσιμων μη προστατευμένων αναφορών
- Στη συνέχεια εξετάζεται ο βαθμός ασφαλείας κάθε προστατευμένης αναφοράς ώστε να διαπιστωθεί αν ο αναλυτής έχει εξουσιοδότηση
- Αν έχει τότε η αναφορά επιστρέφεται μαζί με τις μη προστατευμένες
- Αυτή η διαδικασία επαναλαμβάνεται έως ότου εξαντληθούν όλες οι προστατευμένες αναφορές

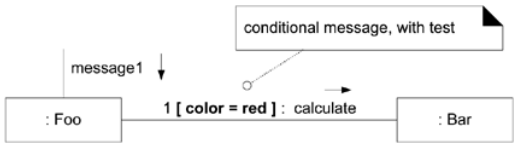
Παράδειγμα χρήσης βρόγχων (loop) (συνέχεια)



Διαγράμματα επικοινωνίας (Communication diagrams)

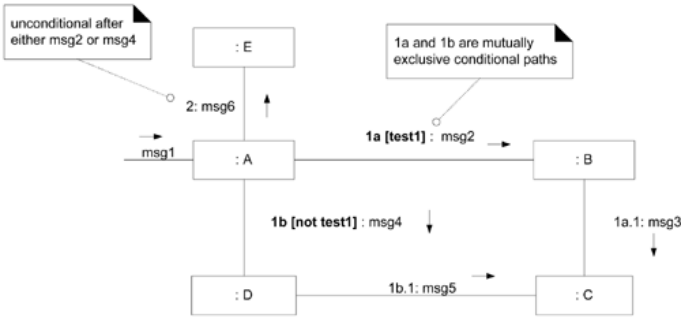
- Τα βασικά στοιχεία που ορίζονται στα διαγράμματα επικοινωνίας είναι:
  - Αντικείμενα (Objects)
  - Συνδέσεις μεταξύ των αντικειμένων (Links)
  - Μηνύματα που ανταλλάσσονται μεταξύ των αντικειμένων (Messages)
    - πολλαπλά μηνύματα ανταλλάσσονται μέσω της ίδιας σύνδεσης και προς τις δύο κατευθύνσεις
    - κάποια μηνύματα μπορεί να περιέχουν μια συνθήκη αποστολής τους (**if then**)
    - κάποια μηνύματα μπορεί να περιέχουν μια συνθήκη αποστολής τους με μια εναλλακτική (**if then else**)
    - κάποια μηνύματα μπορεί να περιέχουν μια συνθήκη επαναληπτικής αποστολής τους (**while**)
  - Ακολουθιακή αρίθμηση των μηνυμάτων που ανταλλάσσονται (Sequential Numbering)

Παράδειγμα χρήσης μηνυμάτων συνθήκης



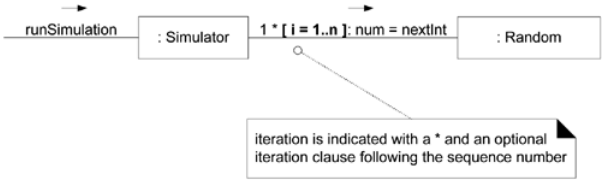
- Αν ισχύει η συνθήκη color=red τότε μόνο θα αποσταλεί το μήνυμα 1 από το αντικείμενο Foo στο αντικείμενο Bar

Παράδειγμα χρήσης μηνυμάτων συνθήκης με εναλλακτική



- Αν η συνθήκη test1 είναι αληθείς τότε θα αποσταλεί το μήνυμα 1α από το αντικείμενο A στο B
- Διαφορετικά θα αποσταλεί το μήνυμα 1b

Παράδειγμα χρήσης μηνυμάτων συνθήκης επανάληψης



- Το μήνυμα 1 θα αποστέλλεται από το αντικείμενο Simulator στο αντικείμενο Random έως ότου ισχύει η συνθήκη
- Το μήνυμα 1 θα αποσταλεί συνολικά n φορές

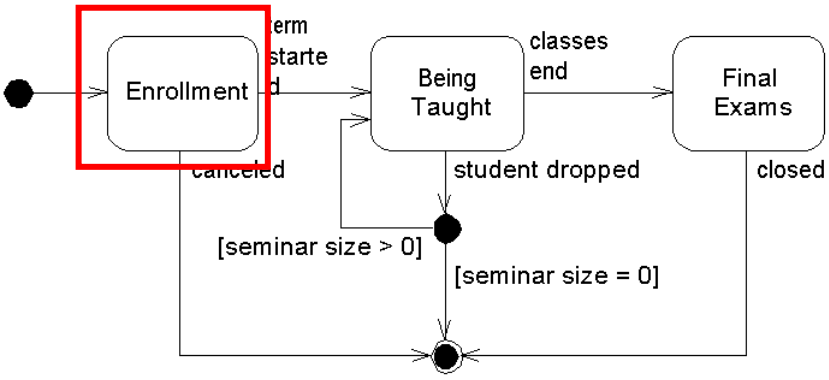
Διαγράμματα Κατάστασης

- Παρουσιάζουν τις διάφορες καταστάσεις και τους τρόπους αλλαγών της κατάστασης ενός αντικειμένου
- Τα βασικά στοιχεία που ορίζονται στα διαγράμματα κατάστασης είναι:
  - Καταστάσεις ενός αντικειμένου σε διάφορες χρονικές στιγμές
    - Ειδικούς συμβολισμούς έχουν η αρχική και τελική κατάσταση ενός αντικειμένου
  - Μεταβάσεις μεταξύ των καταστάσεων ενός αντικειμένου
  - Συνθήκες μετάβασης μεταξύ των καταστάσεων
  - Γεγονότα και ενέργειες που προκαλούν τις μεταβάσεις
  - Ομαδοποίηση καταστάσεων προς σύνθεση μεγαλύτερων

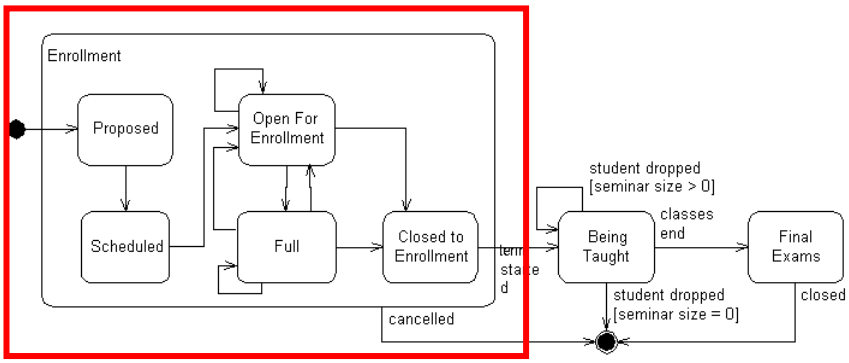
Παράδειγμα ομαδοποίησης καταστάσεων

- Δημιουργήστε ένα διάγραμμα καταστάσεων το οποίο να μοντελοποιεί τον κύκλο ζωής ενός σεμιναρίου από τη στιγμή που θα προταθεί η διενέργειά του μέχρι τη στιγμή που θα ολοκληρωθεί

Παράδειγμα ομαδοποίησης καταστάσεων



Παράδειγμα ομαδοποίησης καταστάσεων



Κανόνες σχεδίασης διαγραμμάτων αλληλουχίας

- Η τοποθέτηση των αντικειμένων που αλληλεπιδρούν γίνεται στην κορυφή του διαγράμματος κατά μήκος του Χ-άξονα
  - Τα αντικείμενα που ξεκινούν την αλληλεπίδραση τοποθετούνται αριστερά και βαθμιαία τοποθετούνται δεξιά τα υπόλοιπα με βάση το χρόνο εμπλοκής τους στην αλληλεπίδραση
- Η τοποθέτηση των μηνυμάτων που αποστέλλονται μεταξύ των αντικειμένων γίνεται κατά μήκος του Υ-άξονα
  - Τα μηνύματα που αποστέλλονται πρώτα βρίσκονται υψηλότερα και ακολουθούν με χρονική σειρά τα υπόλοιπα

Κανόνες σχεδίασης διαγραμμάτων επικοινωνίας

- Τα αντικείμενα που συμμετέχουν στην αλληλεπίδραση τοποθετούνται ως κορυφές ενός γράφου
- Οι συνδέσεις μεταξύ των αντικειμένων αποτελούν τις ακμές του γράφου
- Πάνω στις ακμές γίνεται η τοποθέτηση των μηνυμάτων που ανταλλάσσουν τα αντικείμενα μεταξύ τους

Κανόνες σχεδίασης διαγραμμάτων κατάστασης

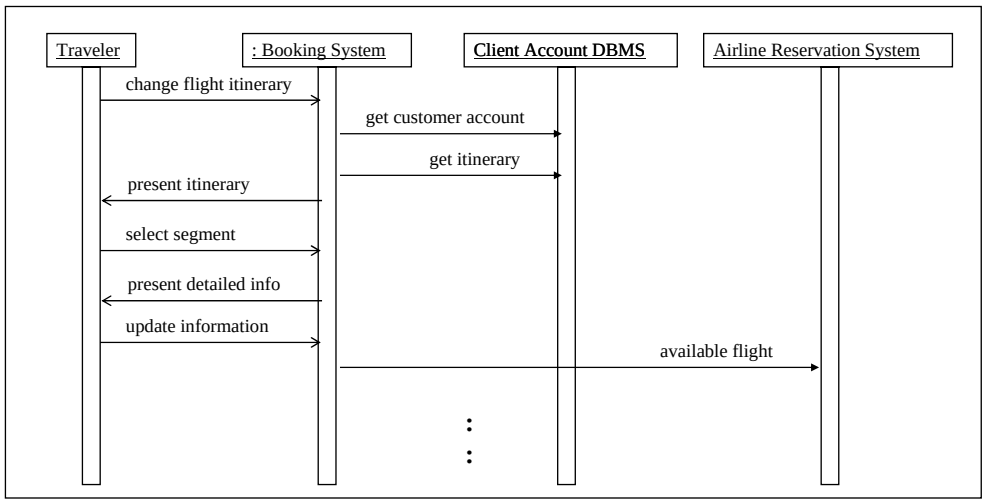
- Εξετάστε όλες τις πιθανές καταστάσεις στις οποίες μπορεί να βρεθούν τα αντικείμενα μιας κλάσης
- Οι διάφορες καταστάσεις που μπορεί να βρεθούν τα αντικείμενα τοποθετούνται ως κορυφές ενός γράφου
- Εξετάστε πώς γίνεται η μετάβαση μεταξύ των καταστάσεων ως αποτέλεσμα των γεγονότων που διαχειρίζεται το αντικείμενο
- Οι μεταβάσεις μεταξύ των καταστάσεων αποτελούν τις ακμές του βρόγχου
- Πάνω στις ακμές γίνεται η τοποθέτηση των συνθηκών μετάβασης από τη μια κατάσταση στην άλλη

Παραδείγματα χρήσης διαγραμμάτων αλληλεπίδρασης και κατάστασης

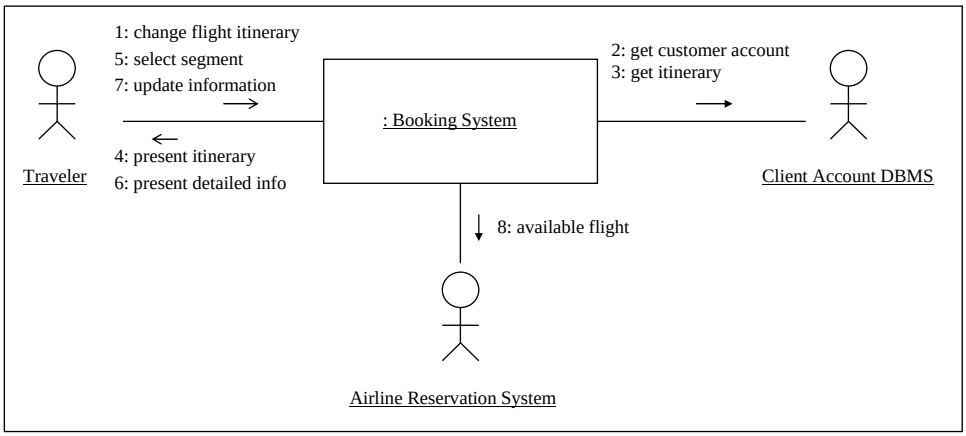
Παράδειγμα 1 - Αλλαγή Ώρας Πτήσης

- **Δράστες**:
  - ταξιδιώτης, βάση δεδομένων, σύστημα κράτησης θέσεων Αεροπορικής Εταιρίας
- **Προαπαιτούμενα (Preconditions)**:
  - Ο Ταξιδιώτης έχει ήδη εγγραφεί στο σύστημα και έχει επιλέξει «αλλαγή ταξιδιού»
- **Βασική Ροή της Περίπτωσης Χρήσης**
  - Το σύστημα ανακτά τις πληροφορίες του λογαριασμού και τη πτήση (πτήσεις) του ταξιδιώτη από τη βάση δεδομένων
  - Το σύστημα ρωτά τον ταξιδιώτη να επιλέξει κάποια συγκεκριμένη πτήση. Ο ταξιδιώτης επιλέγει συγκεκριμένη πτήση.
  - Το σύστημα ρωτά τον ταξιδιώτη για την προτεινόμενη νέα ώρα αναχώρησης. Ο ταξιδιώτης δίνει την συγκεκριμένη πληροφορία.
  - Εάν η πτήση είναι διαθέσιμη ...
  - …
  - Το σύστημα παρουσιάζει τη περίληψη της συναλλαγής.
- **Εναλλακτικές Ροές της Περίπτωσης Χρήσης**
  - Εάν η πτήση δεν είναι διαθέσιμη τότε …

Διάγραμμα αλληλουχίας – Αλλαγή Ώρας Πτήσης

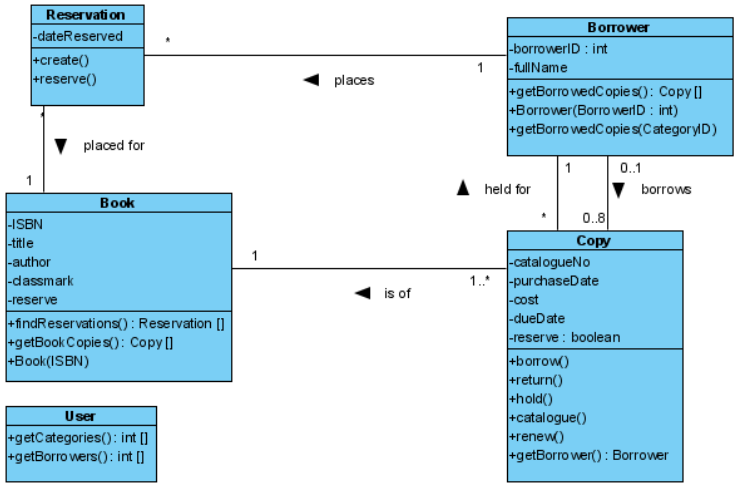


Διάγραμμα επικοινωνίας – Αλλαγή Ώρας Πτήσης



Παράδειγμα 2: Λειτουργίες Βιβλιοθήκης

- Το παρακάτω διάγραμμα κλάσεων μοντελοποιεί μια βιβλιοθήκη



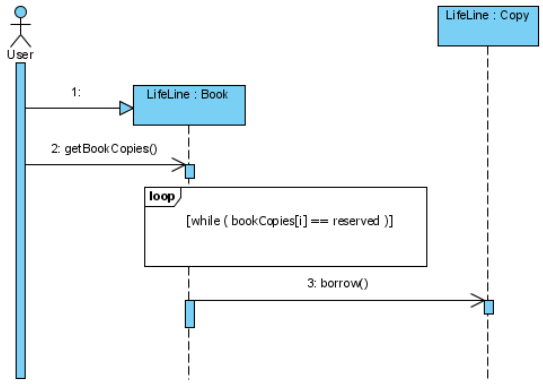
Παράδειγμα 2 : Λειτουργίες Βιβλιοθήκης

- Σχεδιάσετε διαγράμματα αλληλουχίας για την υλοποίηση των εξής περιπτώσεων χρήσης: (α) δανεισμός, (β) κράτηση, (γ) επιστροφή, (δ) ανανέωση βιβλίου και (ε) εύρεση των κρατήσεων

- Κατόπιν σχεδιάσετε το διάγραμμα επικοινωνίας για τη λειτουργία της επιστροφής βιβλίου (για την αρίθμηση των μηνυμάτων χρησιμοποιήσετε την εμφωλευμένη μορφή -nested numbering).

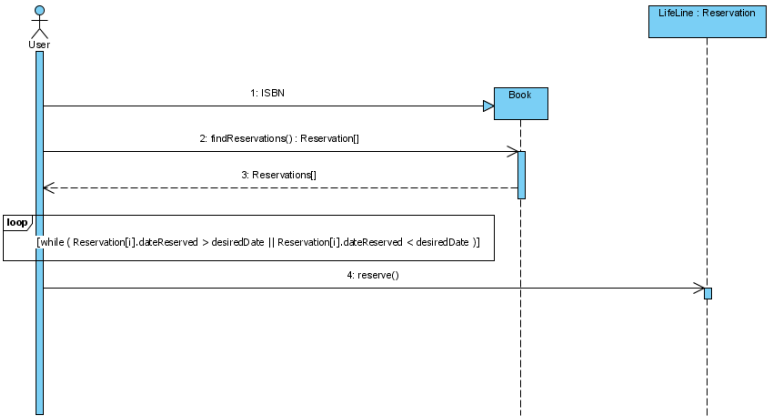
Παράδειγμα 2 : Λειτουργία Δανεισμού

- Για να δανειστεί κάποιος ένα αντίγραφο ενός βιβλίου θα πρέπει να υπάρχει τουλάχιστον ένα αντίγραφο αυτού του βιβλίου για το οποίο κάποιος δεν έχει κάνει κράτηση.



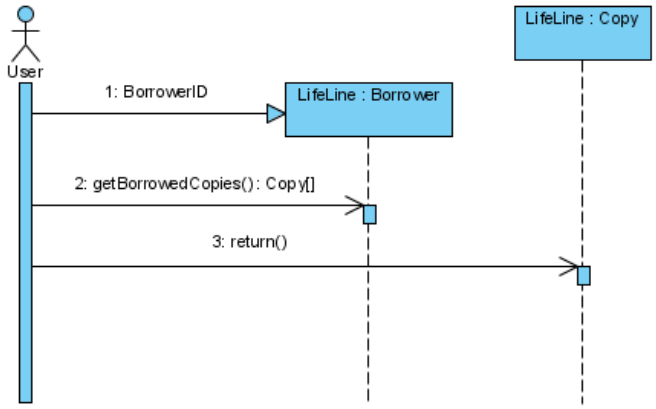
Παράδειγμα 2: Λειτουργία Κράτησης

- Για να γίνει μια κράτηση θα πρέπει να υπάρχει τουλάχιστον ένα αντίγραφο ενός βιβλίου για το οποίο κάποιος δεν έχει κάνει κράτηση



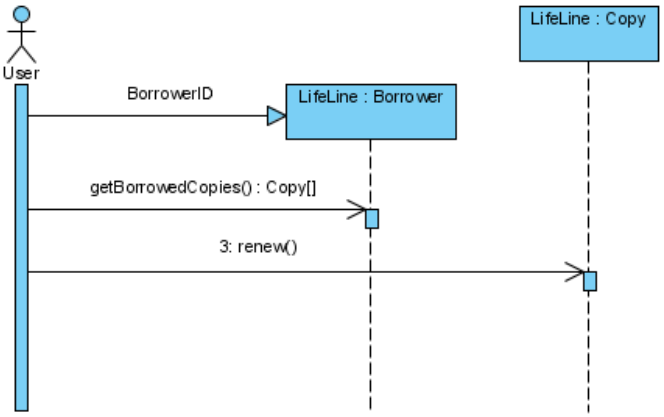
Παράδειγμα 2 : Λειτουργία Επιστροφής

- Για να γίνει επιστροφή ενός βιβλίου θα πρέπει να βρεθεί το αντίγραφο το οποίο έχει δανειστεί ο δανειζόμενος και να επιστραφεί.



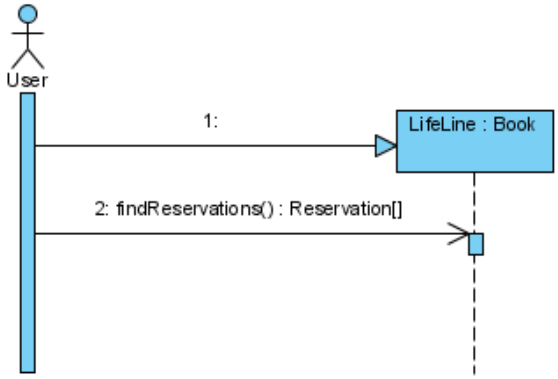
Παράδειγμα 2 : Λειτουργία Ανανέωσης

- Για να γίνει η ανανέωση θα πρέπει να βρούμε το αντίγραφο του βιβλίου το οποίο έχει δανειστεί ο δανειζόμενος και να το ανανεώσουμε.

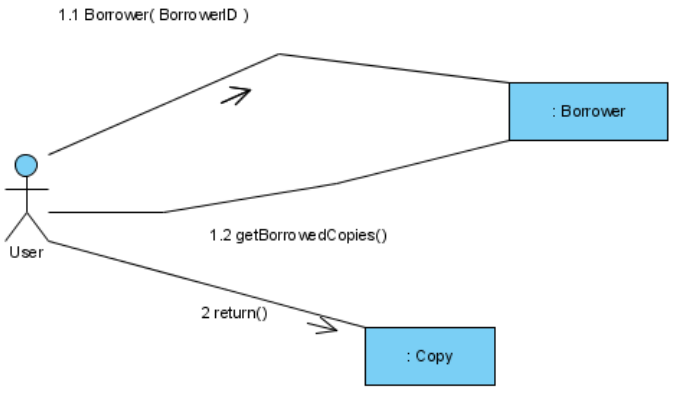


Παράδειγμα 2 : Λειτουργία Εύρεσης Κρατήσεων

- Για να βρούμε τις κρατήσεις ξεκινάμε από το βιβλίο και ζητάμε τις κρατήσεις



Παράδειγμα 2 : Λειτουργία Επιστροφής βιβλίου



Παράδειγμα 2 – Εκτέλεση Σύνθετων Λειτουργιών

- Έστω ότι ο βιβλιοθηκάριος θέλει:
  - να γνωρίζει πόσα βιβλία έχει δανειστεί κάθε πελάτης ανά κατηγορία
  - να γνωρίζει το πλήθος των εμπρόθεσμων και εκπρόθεσμων επιστροφών ανά χρήστη
- Σχεδιάσετε τα διαγράμματα αλληλουχίας που περιγράφουν τις παραπάνω επαναληπτικές διαδικασίες.
- Σε πρώτη φάση σχεδιάστε αλγοριθμικά την επαναληπτική διαδικασία και έπειτα το διάγραμμα ακολουθίας

Παράδειγμα 3 – Εκτέλεση Σύνθετων Λειτουργιών

Βρες όλες τις κατηγορίες

Βρες όλους όσους έχουν δανειστεί βιβλία

Για κάθε δανειζόμενο

  Για κάθε κατηγορία

    Πάρε πλήθος των δανεισμών που έχει κάνει για κάθε κατηγορία

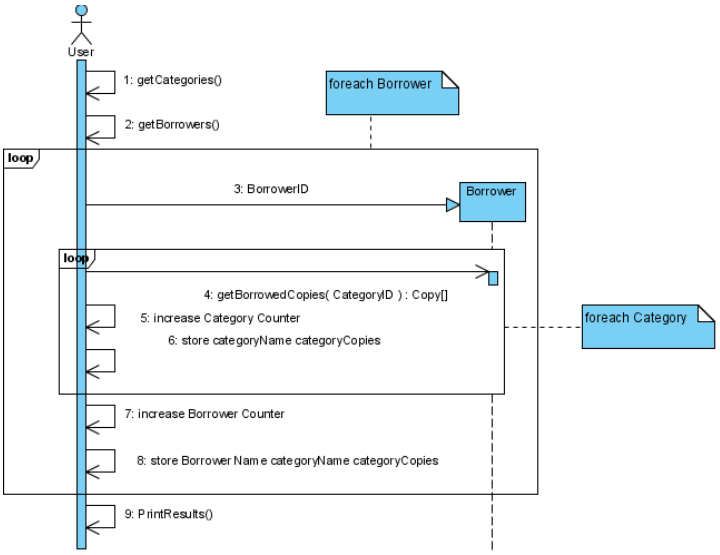
    Αποθήκευσε τα δεδομένα

  επόμενος κατηγορία

επόμενη δανειζόμενος

τύπωσε αποτελέσματα

Παράδειγμα 3 – Εκτέλεση Σύνθετων Λειτουργιών



Παράδειγμα 3 – Εκτέλεση Σύνθετων Λειτουργιών

Βρες όλους τους δανειζόμενους

Για κάθε δανειζόμενο

Βρες τα αντίτυπα που έχει δανειστεί

Για κάθε αντίτυπο

αν είναι εμπρόθεσμη η επιστροφή αύξησε τα εμπρόθεσμα
αλλιώς αύξησε τα εκπρόθεσμα

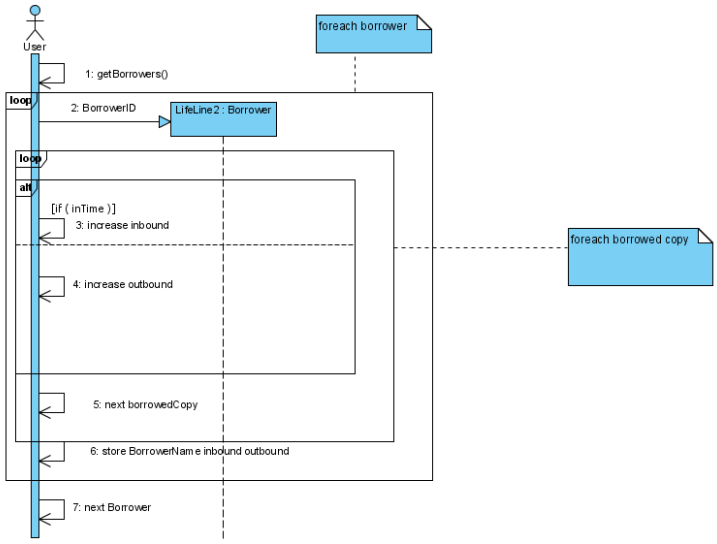
επόμενο αντίτυπο

αποθήκευσε τα δεδομένα

επόμενος δανειζόμενος

τύπωσε αποτελέσματα

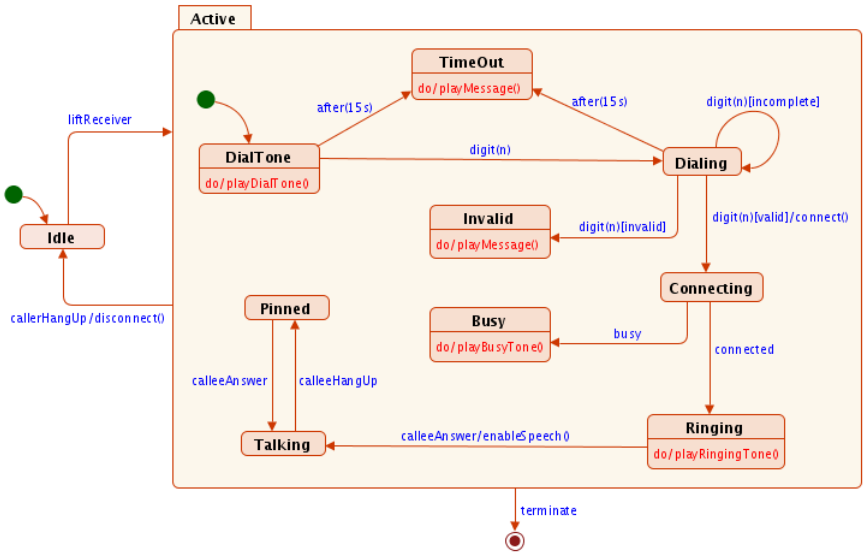
Παράδειγμα 3 – Εκτέλεση Σύνθετων Λειτουργιών



Παράδειγμα 4 – Μοντελοποίηση τηλεφωνικής κλήσης

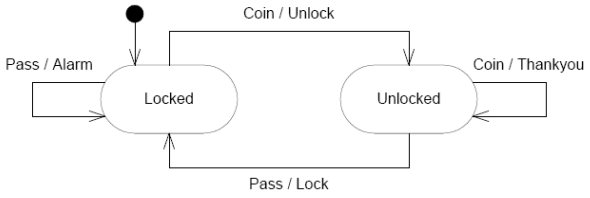
- Ο χρήστης σηκώνει το ακουστικό οπότε η γραμμη άπο **ανενεργή** να γίνει **ενεργή**
- Η κατάσταση **ενεργή** είναι μια σύνθετη κατάσταση μέσα στην οποία εκτελούνται ένα πλήθος από μεταβάσεις μεταξύ υποκαταστάσεων
- Αν υπάρξει τόνος τότε μεταβαίνουμε στην κατάσταση κλήσης του αριθμού
- Σε περίπτωση σύνδεσης το τηλέφωνο μπορεί να είναι είτε κατειλημμένο είτε ελεύθερο οπότε και θα πραγματοποιηθεί η συνομιλία
- Όταν τελειώσει η συνομιλία τότε η γραμμή γίνεται και πάλι ανενεργή

Παράδειγμα 4 – Μοντελοποίηση τηλεφωνικής κλήσης



Υλοποίηση διαγραμμάτων κατάστασης σε C++

Διάγραμμα κατάστασης



Υλοποίηση διαγράμματος κατάστασης

```
enum State {Locked, Unlocked};
enum Event {Pass, Coin};
void Unlock();
void Lock();
void Thankyou();
void Alarm();
```

```
void Transition(Event e){
static State s = Locked;
switch(s){
  case Locked:
    inner switch case 1;
    break;
  case Unlocked:
     inner switch case 2;
     break;
 }}
```

inner switch case 1

```
switch(e){
  case Coin:
     s = Unlocked;
     Unlock();
     break;
  case Pass:
    Alarm();
    break;}
```

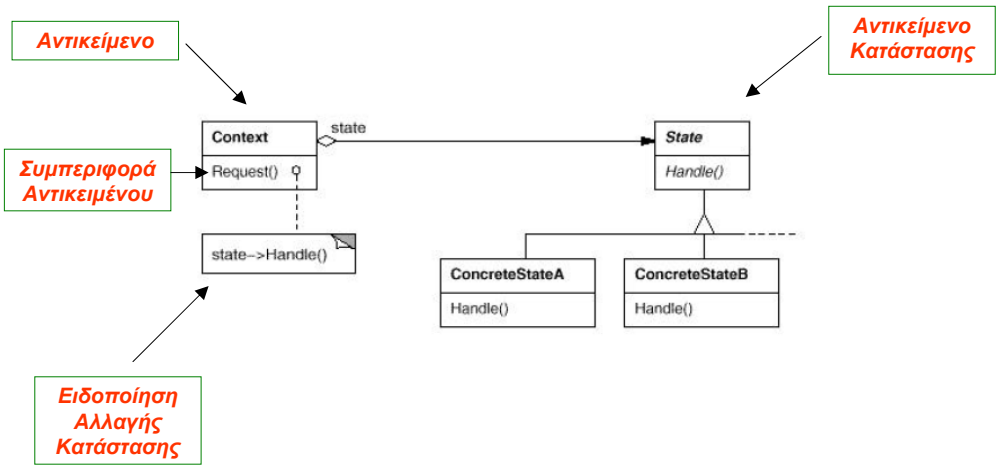
inner switch case 2

```
switch(e){
   case Coin:
      Thankyou();
      break;
   case Pass:
      s = Locked;
      Lock();
   break;}
```

Πρότυπο Σχεδίασης Καταστάσεων

- Επιτρέπει σε ένα αντικείμενο να ειδοποιεί για τη συμπεριφορά του όταν αλλάζει η εσωτερική του κατάσταση
- Χρησιμοποιείται οποτεδήποτε:
  - Η συμπεριφορά ενός αντικειμένου εξαρτάται από την κατάστασή στην οποία βρίσκεται και πρέπει να αλλάξει την συμπεριφορά του σε χρόνο εκτέλεσης (run time) με βάση τη νέα του κατάσταση
- Πλεονεκτήματα χρήσης:
  - Τοποθετεί όλη τη συμπεριφορά που σχετίζεται με κάποια κατάσταση μέσα σε ένα αντικείμενο κατάστασης
  - Επιτρέπει την ενσωμάτωση λογικών μεταβάσεων κατάστασης μέσα σε ένα αντικείμενο κατάστασης
  - Βοηθάει στην αποφυγή μη συνεπών καταστάσεων εφόσον οι αλλαγές στις καταστάσεις πραγματοποιούνται με χρήση ενώ μονάχα αντικειμένου κατάστασης

Πρότυπο Σχεδίασης Καταστάσεων

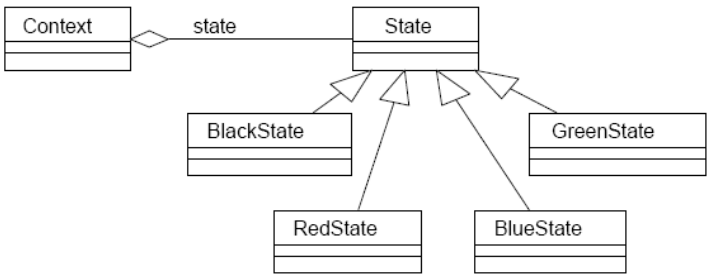


Design Patterns In Java - Bob Tarr

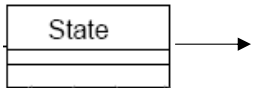
Παράδειγμα Χρήσης Προτύπου Σχεδίασης Καταστάσεων

- Θεωρήστε μία κλάση η οποία έχει δύο μεθόδους, push() και pull(), των οποίων η συμπεριφορά αλλάζει ανάλογα την κατάσταση που βρίσκεται το αντικείμενο
- Οι καταστάσεις που μπορεί να βρεθεί το αντικείμενο είναι μαύρο, κόκκινο, μπλε και πράσινο

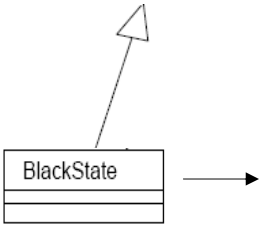
Παράδειγμα Χρήσης Προτύπου Σχεδίασης Καταστάσεων



Παράδειγμα Χρήσης Προτύπου Σχεδίασης Καταστάσεων




```
public abstract class State {
   public abstract void handlePush(Contextc);
   public abstract void handlePull(Context c);
   public abstract Color getColor();
}
```




```
public class BlackState extends State {
   public void handlePush(Contextc) {
        c.setState(new RedState());
    }
    public void handlePull(Contextc) {
        c.setState(new GreenState());
    }
   public Color getColor(){
       return(Color.black);
   }
}
```


Παράδειγμα Χρήσης Προτύπου Σχεδίασης Καταστάσεων

- Οι μέθοδοι pull() και push() εκτελούν διαφορετικές ενέργειες ανάλογα την κατάσταση του αντικειμένου

- Με χρήση του προτύπου σχεδίασης καταστάσεων εξουσιοδοτούμε αυτή τη συμπεριφορά στο αντικείμενο κατάστασης

```
public class Context {
   private State state = null;
   public Context(State state) {this.state = state;}
   public Context() {this(new RedState());}
   public State getState() {return state;}
   public void setState(State state) {this.state = state;}
   public void push() {state.handlePush(this);}
   public void pull() {state.handlePull(this);}
}
```