Διάγραμμα Κλάσεων

Class Diagram

Γενικά…

- Ορίζει τις κλάσεις αντικειμένων σε ένα σύστημα, τις μεθόδους και τις συναρτήσεις τους, και τις συσχετίσεις μεταξύ των κλάσεων.
- Περιγράφουν την δομή και συμπεριφορά των ΠΧ
- Δίνουν ένα εννοιολογικό μοντέλο του συστήματος, με τους όρους οντοτήτων και σχέσεων
- Τα αναλυτικά διαγράμματα κλάσεων χρησιμοποιούνται από τους προγραμματιστές

Περιλαμβάνουν…

- Κλάσεις
- Διασυνδέσεις (interfaces)
- Συνεργασίες
- Εξαρτήσεις, Γενικεύσεις, Συσχετίσεις και Σχέσεις
- Σημειώσεις και Περιορισμούς

Κλάση

- Κλάση είναι η περιγραφή ενός συνόλου αντικειμένων με κοινά χαρακτηριστικά (attributes) και συμπεριφορά (operations).

Αντικείμενο

- Ένα αντικείμενο αναπαριστά μία οντότητα, είτε του πραγματικού κόσμου, είτε όχι.
- Στη UML, το αντικείμενο αναπαριστάτε ως τετράγωνο, με το όνομα του αντικειμένου υπογραμμισμένο.

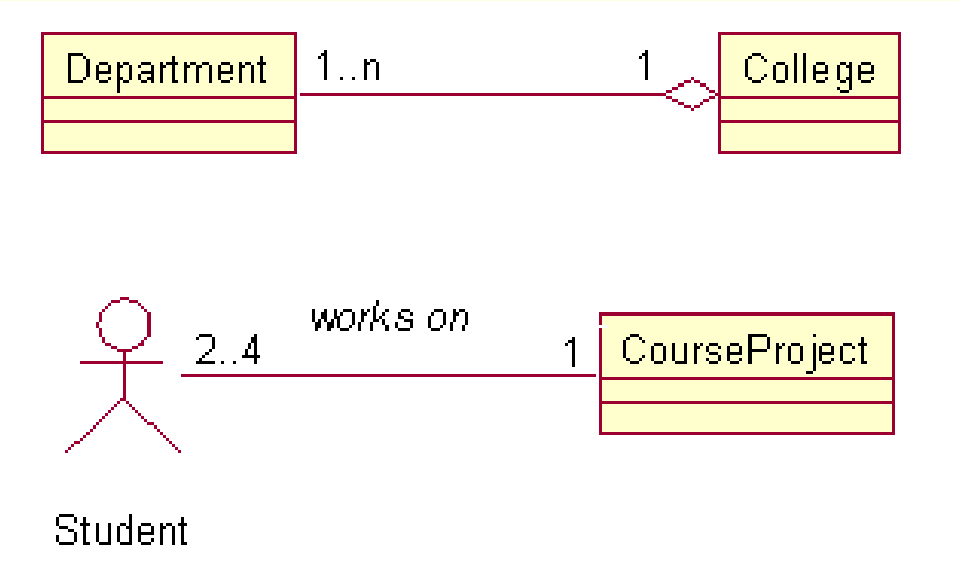
Σχέσεις μεταξύ κλάσεων

- Υπάρχουν δύο τύποι σχέσεων
  - Γενίκευση
  - Συσχέτιση
- Οι συσχετίσεις χωρίζονται επιπλέον σε
  - Απλή Συσχέτιση
  - Σύνθεση
  - Συγκρότηση

Πολλαπλότητα Σχέσεων

- Η πολλαπλότητα μίας συσχέτισης καθορίζει τον αριθμό των στιγμιότυπων μιας κλάσης με τα οποία μπορεί να συσχετιστεί μια άλλη κλάση.

- Παραδείγματα: ένα-προς-ένα, ένα-προς-πολλά, πολλά-προς-πολλά και οποιαδήποτε άλλη αριθμητική σχέση.

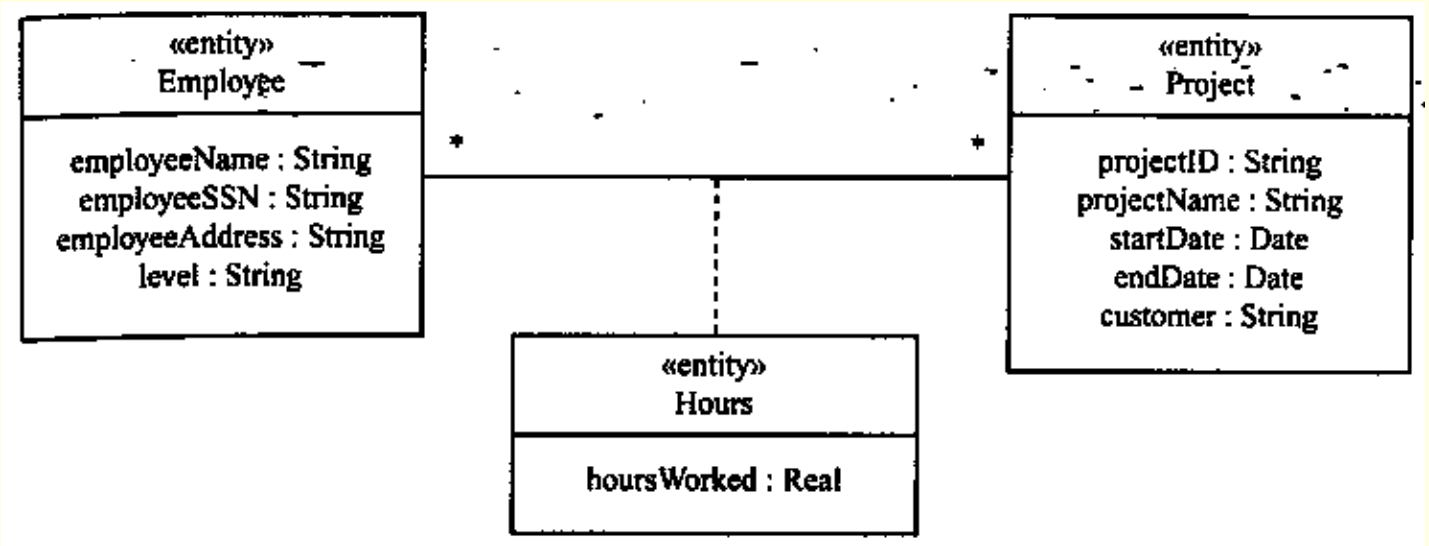


Multiplicity

Symbol	Meaning
1	One and only one
0..1	Zero or one
M..N	From M to N
*	From zero to any positive integer
0..*	From zero to any positive integer
1..*	From one to any positive integer

Κλάση Συσχέτισης

- Μία κλάση συσχέτισης είναι μια κλάση η οποία μοντελοποιεί την σχέση μεταξύ δύο άλλων κλάσεων.
- Βρίσκει κυρίως εφαρμογή σε σχέσεις πολλά-προς-πολλά.
- Οι ιδιότητες της σχέσης είναι οι ιδιότητες της κλάσης συσχέτισης

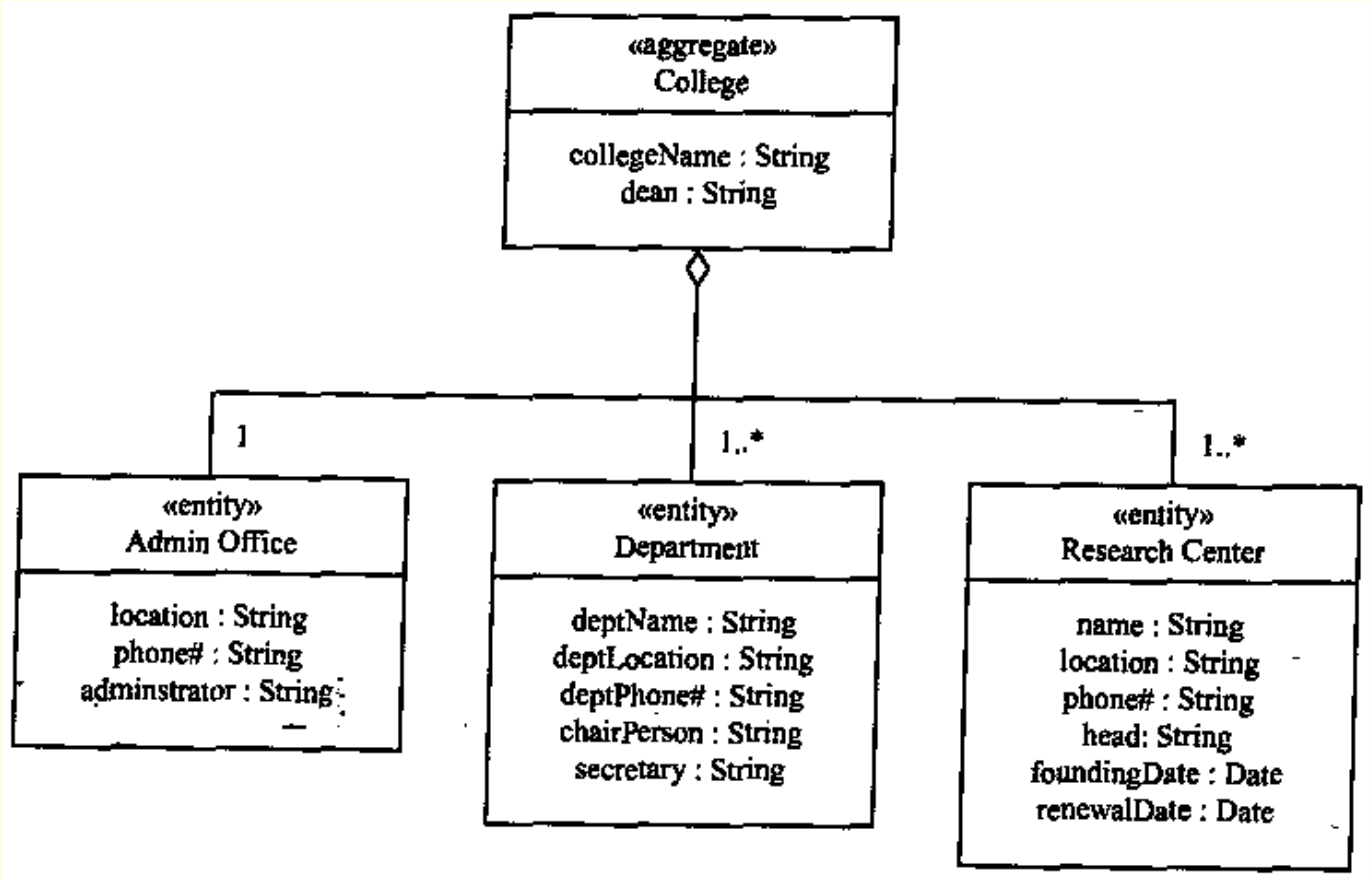


Συγκρότηση (1/2)

- **Συγκρότηση (Aggregation)**: εκφράζει μια σχέση μεταξύ στιγμιότυπων κλάσεων. Είναι σχέση περιεκτικότητας.
- Εκφράζει μια σχέση όπου ένα αντικείμενο κλάση-Container έχει την ευθύνη να διατηρήσει και να διαχειριστεί αντικείμενα κλάσης-Containee τα οποία έχουν δημιουργηθεί εκτός της κλάσης Container.
- Τα αντικείμενα (Container και Containees) μπορούν να διαχειριστούν ανεξάρτητα.
- Τα αντικείμενα (Container και Containees) δεν έχουν κάποια ιδιαίτερα δικαιώματα το ένα στο άλλο.

Συγκρότηση (2/2)

- IS-PART-OF relationships



Σύνθεση (1/3)

- **Σύνθεση (Composition):** εκφράζει μια σχέση μεταξύ στιγμιότυπων συσχετιζόμενων κλάσεων. Είναι σχέση της μορφής: Σύνθετο – Συστατικό

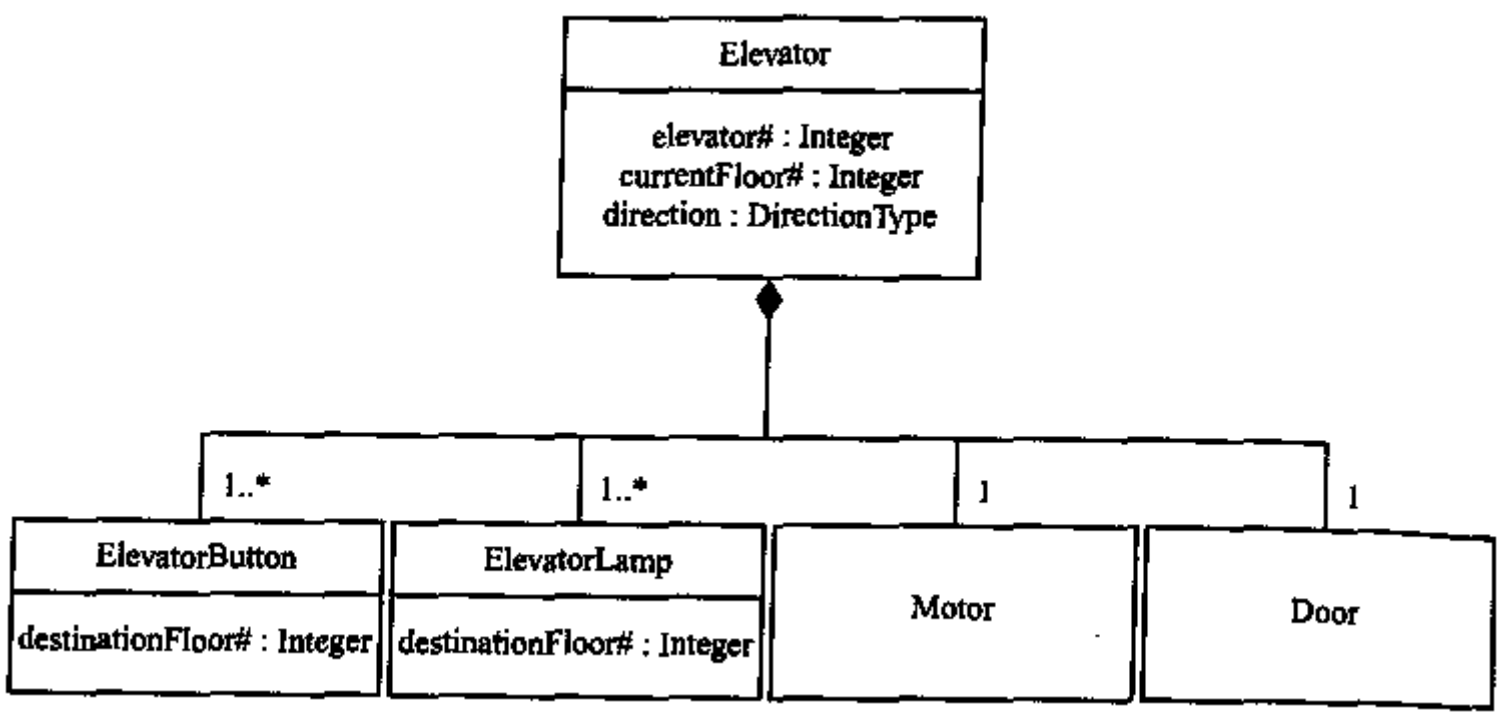
- Εκφράζει μια σχέση όπου ένα στιγμιότυπο μιας κλάσης – Σύνθετο, έχει την ευθύνη να δημιουργήσει και να αρχικοποιήσει κάθε κλάση-Συστατικό που περιέχει.

Σύνθεση (2/3)

- Η σύνθεση χρησιμοποιείτε σε περιπτώσεις όπου το αντικείμενο κλάση-σύνθετο έχει αποκλειστική πρόσβαση και έλεγχο στο στιγμιότυπο κλάση-συστατικό.
- Η συμπεριφορά του συστατικού δεν ορίζετε χωρίς την ύπαρξη του σύνθετου.
- Η συμπεριφορά του σύνθετου είναι ελλιπής χωρίς την ύπαρξη του συστατικού.

Σύνθεση (3/3)

- Ισχυρή συσχέτιση
- Τα συστατικά δημιουργούνται και καταστρέφονται με τον ιδιοκτήτη τους



Συγκρότηση vs. Σύνθεση

Η **Σύνθεση** αποτελεί μια ισχυρή μορφή συγκρότησης

- Τα συστατικά μπορούν να έχουν μόνο έναν ιδιοκτήτη
- Τα συστατικά δεν υπάρχουν χωρίς τον ιδιοκτήτη τους
- Τα συστατικά δημιουργούνται κ καταστρέφονται με τον ιδιοκτήτη τους (π.χ. Μηχανή - Αυτοκίνητο)

Στη **Συγκρότηση** τα αντικείμενα που συμμετέχουν μπορούν να υπάρχουν και ανεξάρτητα το ένα από το άλλο. (π.χ. Μάθημα - Βιβλίο)

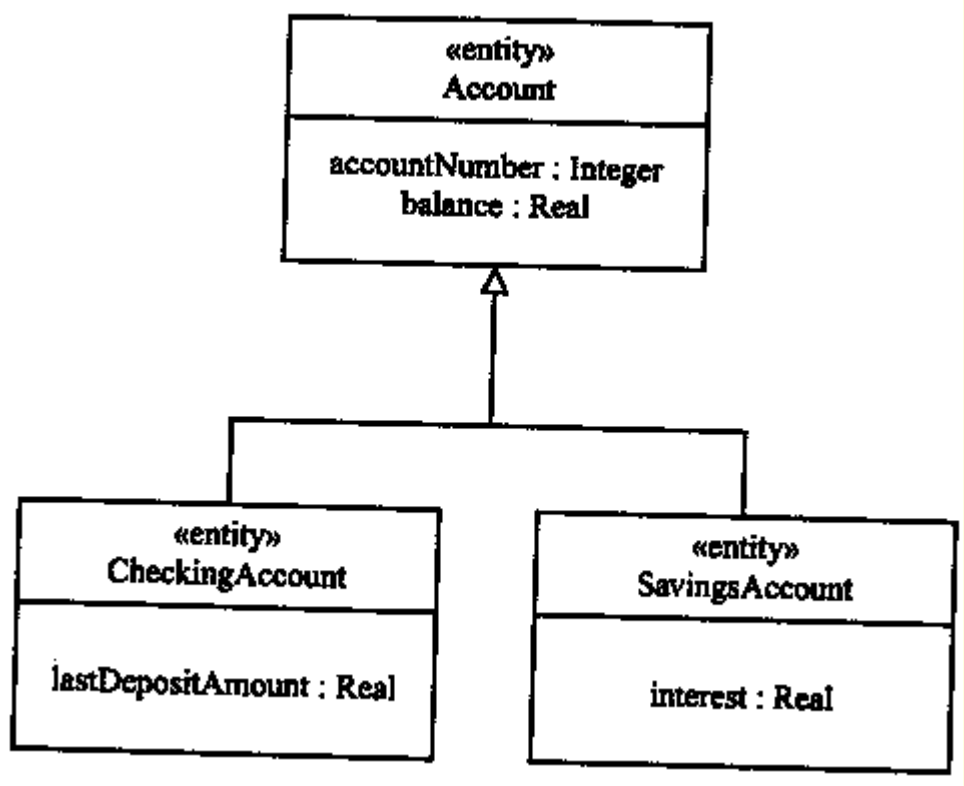
Γενίκευση - Κληρονομικότητα

Η Γενίκευση εκφράζει μια σχέση γονιού - παιδιού μεταξύ των συνδεδεμένων κλάσεων

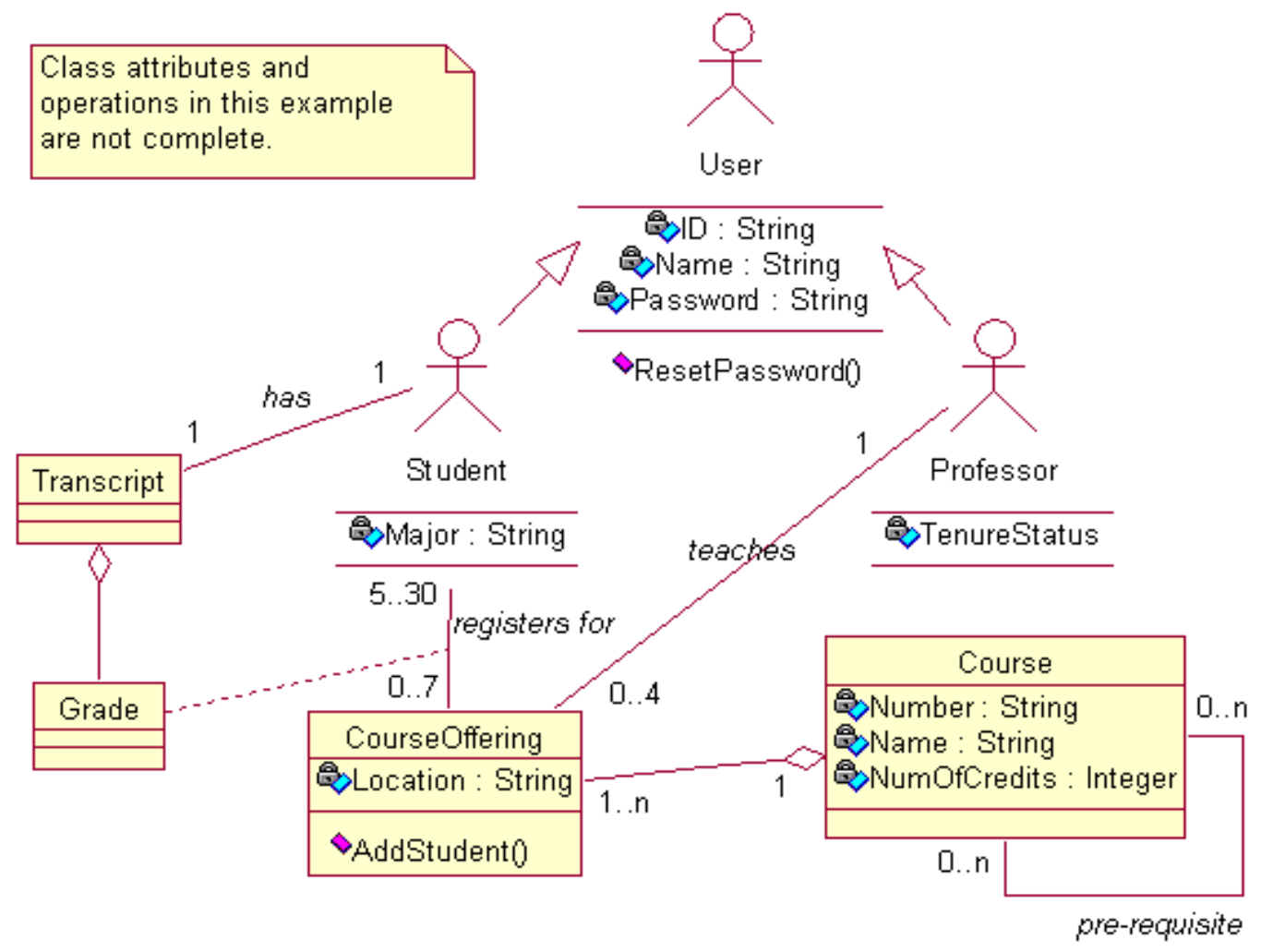
Χρησιμοποιείτε για να υλοποιεί αφαιρέσεις και για να κληροδοτήσει ιδιότητες - συμπεριφορές

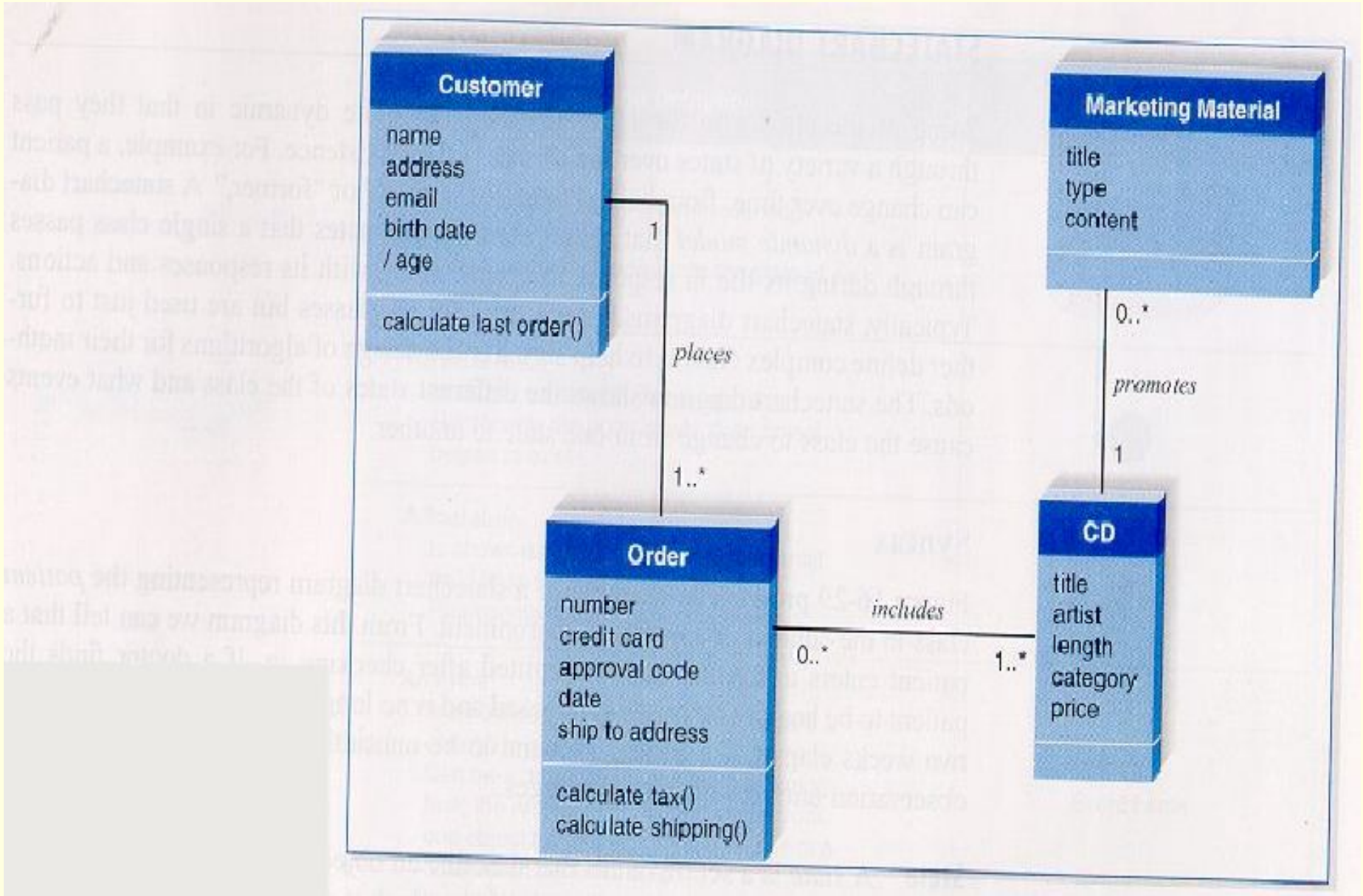
Γενίκευση - Κληρονομικότητα

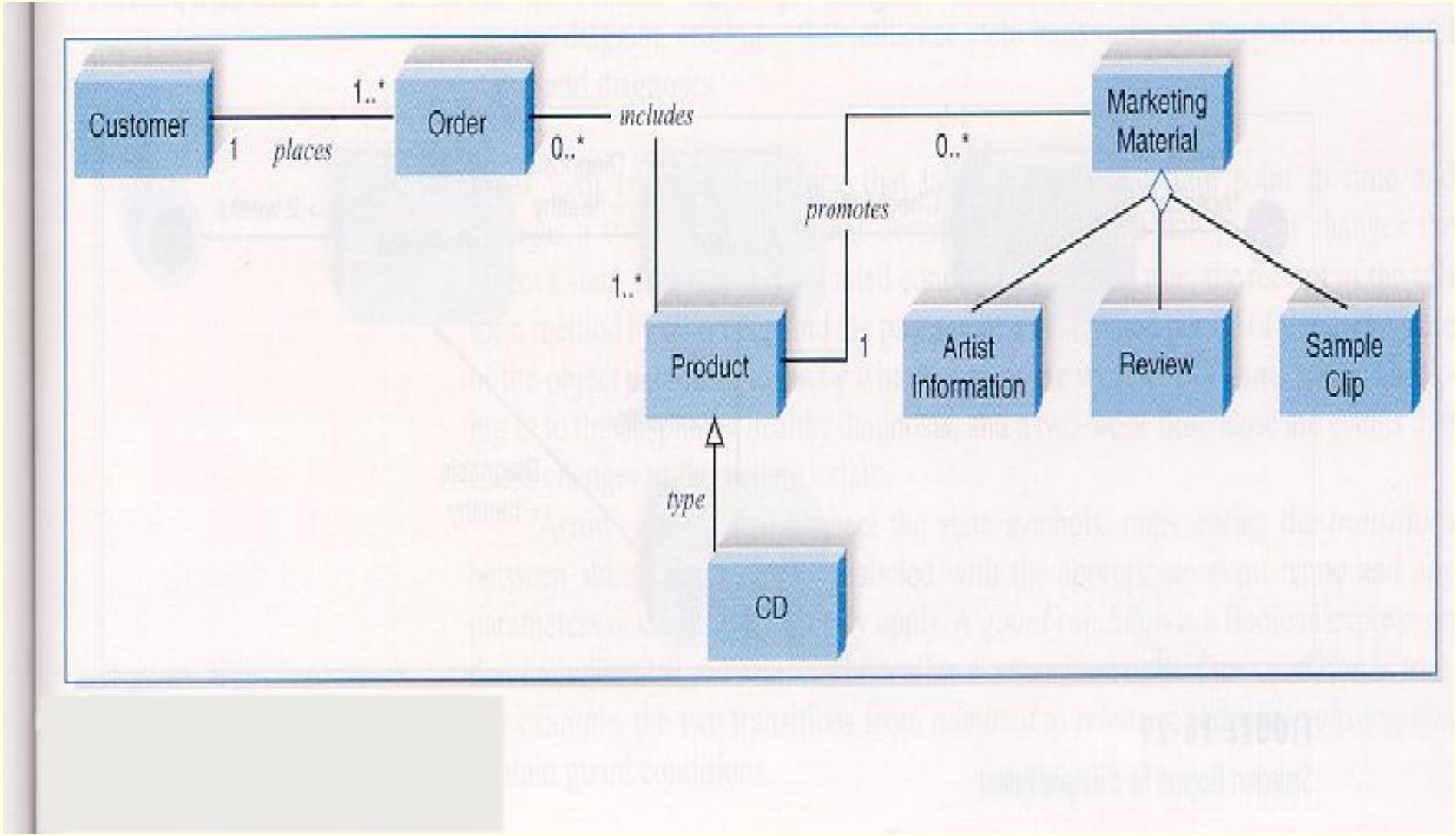
- IS-A relationships

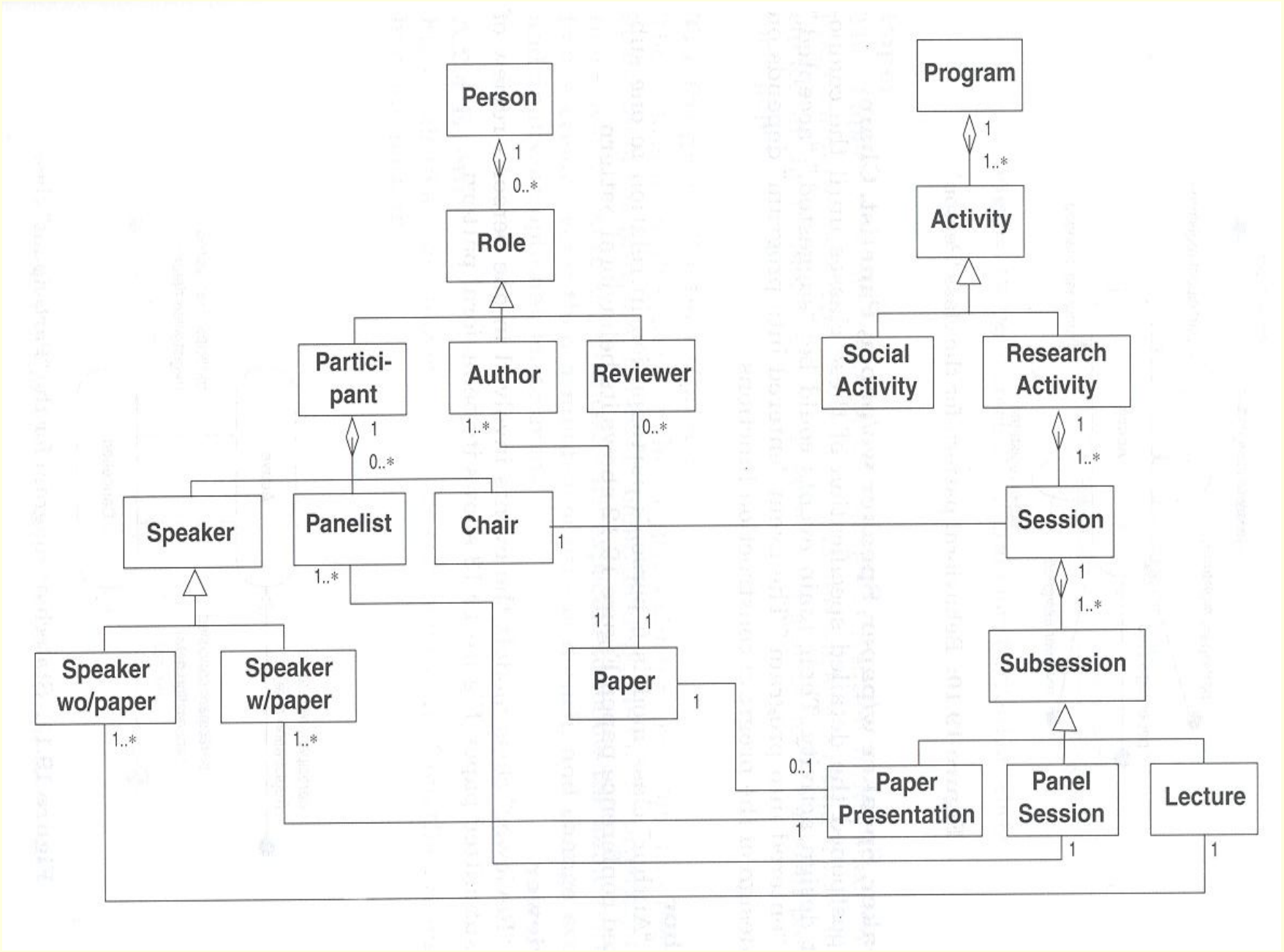


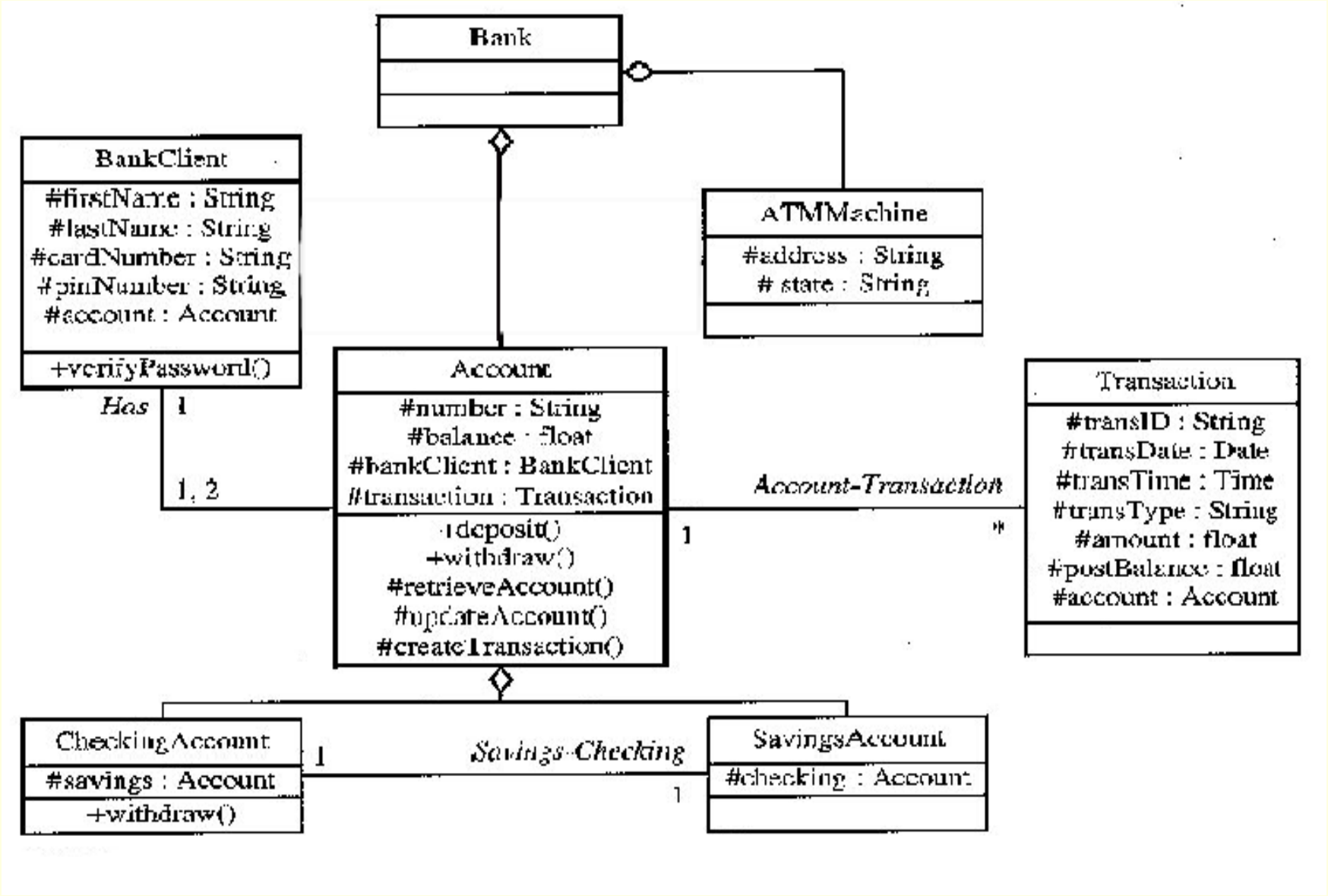
Παραδείγματα




Customer
name
address
email
birth date
/ age
calculate last order()
1
places
1..*
Order
number
credit card
approval code
date
ship to address
calculate tax()
calculate shipping()
0..*
includes
1..*
Marketing Material
title
type
content
0..*
promotes
1
CD
title
artist
length
category
price


Customer
1..*
1
places
Order
0..*
includes
1..*
Product
type
CD
promotes
0..*
1
Marketing Material
Artist Information
Review
Sample Clip


Person
Program
Role
Activity
Partici-
pant
Author
Reviewer
Social
Activity
Research
Activity
Speaker
Panelist
Chair
Session
Speaker
wo/paper
Speaker
w/paper
Paper
Subsession
Paper
Presentation
Panel
Session
Lecture
1
0..*
1..*
0..1


Bank
BankClient
#firstName : String
#lastName : String
#cardNumber : String
#pinNumber : String
#account : Account
+verifyPassword()
ATMMachine
#address : String
state : String
Has
1
1, 2
Account
#number : String
#balance : float
#bankClient : BankClient
#transaction : Transaction
+deposit()
+withdraw()
#retrieveAccount()
#updateAccount()
#createTransaction()
Account-Transaction
1
*
Transaction
#transID : String
#transDate : Date
#transTime : Time
#transType : String
#amount : float
#postBalance : float
#account : Account
CheckingAccount
#savings : Account
+withdraw()
1
Savings-Checking
1
SavingsAccount
#checking : Account

Μεταφορά της UML σε Java

Java	UML

`package BusinessObjects;`
 	
`public class Employee {`
 	
`}`
	

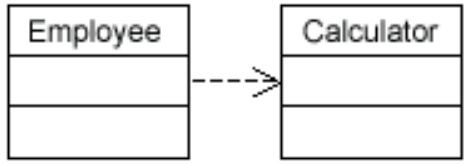
Μεταφορα της UML σε Java

Java	UML

`public interface CollegePerson {`
`  public Schedule getSchedule();`
 	
`}`
	

Μεταφορα της UML σε Java

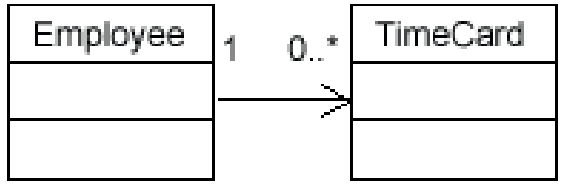
Java	UML

public class Employee {
public void calcSalary(CalculatorStrategy
{
…
}
}
	

Μεταφορα της UML σε Java

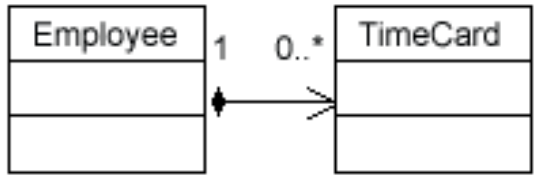
Θα έπρεπε να μπεί πίνακας

Java	UML

public class Employee {
private TimeCard _tc;
public void maintainTimeCard() {
…
}
}
	

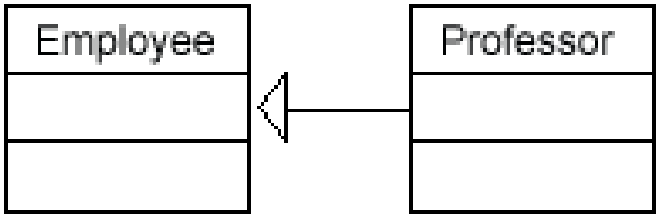
Μεταφορα της UML σε Java

Java	UML

`public class Employee {`
`  private TimeCard tc;`
`  public void maintainTimeCard() {`
`  …`
`  }`
`}`
	

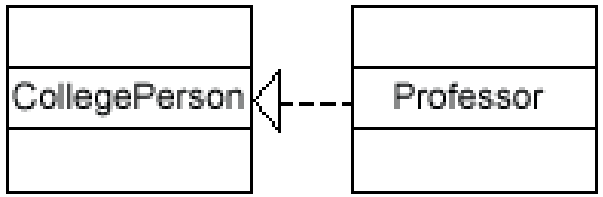
Μεταφορα της UML σε Java

Java	UML

`public abstract class Employee {`
 	
`}`
`public class Professor extends Employee {`
 	
`}`
	

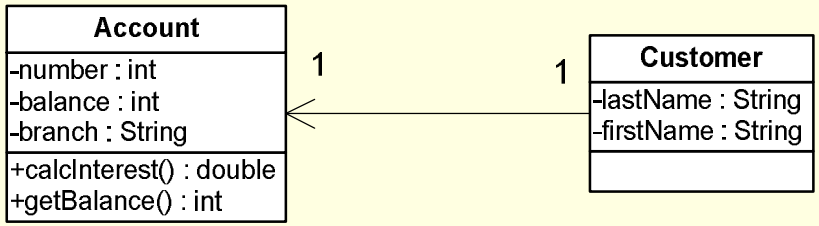
Μεταφορα της UML σε Java

Java	UML

public interface CollegePerson {
 	
}
public class Professor implements CollegePers
 	
}
	

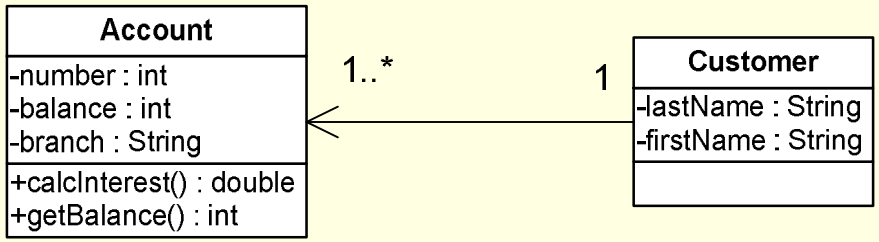
C++ σε UML

```
Class Customer
{
        public:
        Customer();
        Account* getAccount()
           {return theAccount;}
        void setAccount(Account
           *value)
           {theAccount=value;}
        private:
        string lastName;
        string firstName;
        Account* theAccount;
}
```



C++ σε UML

```
class Customer
{
        public:
        Customer();
        Account* getAccount (int
            index ) {return
            theAccounts[index];}
        void setAccount(int index,
            Account
            *value){relatedAccount=v
            alue;}
        private:
        string lastName;
        string firstName;
        Account* theAccounts[];
}
```

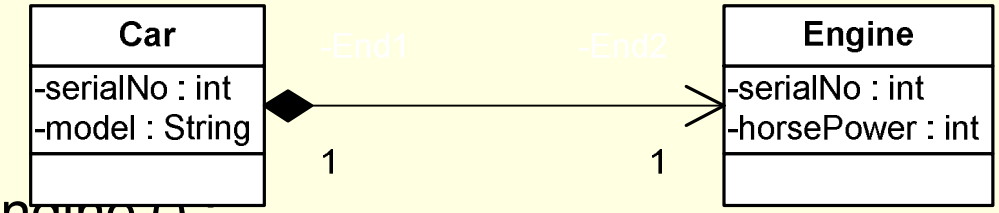


C++ σε UML

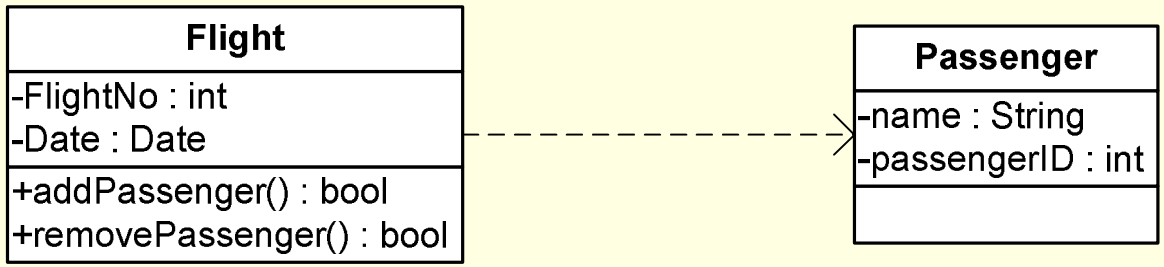
```
class Car
{
        public:
        Car();
        Engine* getEngine () ;
        void setEngine(Engine
           *value);
        private:
        string model;
        int serialNo;
        Engine* theEngine;
}
```

C++ σε UML

```
class Car
{
        public:
        Car();
        Engine getEngine () ;
        void setEngine(Engine
          value);
        private:
        string model;
        int serialNo;
        Engine theEngine;
}
```



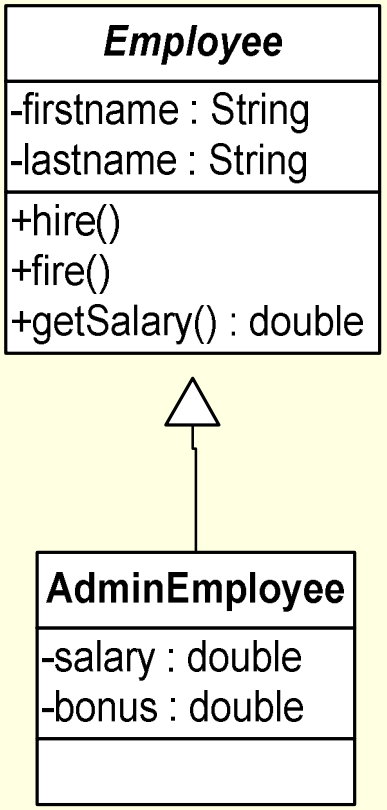
C++ σε UML




```
class Flight
{
        public:
        Flight();
        bool addPassenger (Passenger p) ;
        bool removePassenger (Passenger p);
        private:
        int flightNo;
        date flightdate;
}
```

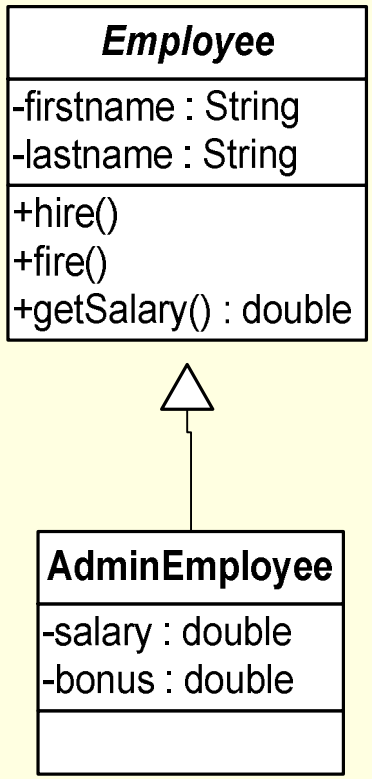
C++ σε UML

```
class Employee
{
        public:
        Employee();
        void Hire() ;
        void Fire();
        virtual double
           getSalary();
        private:
        string firstname;
        string lastname;
}
```



C++ σε UML

```
class AdminEmployee: public
    Employee
{
        public:
        AdminEmployee();
        double getSalary();
        private:
        double salary;
        double bonus;
}
```



Σύντομο Παράδειγμα μιας συνοπτικής μεθοδολογίας ανάπτυξης

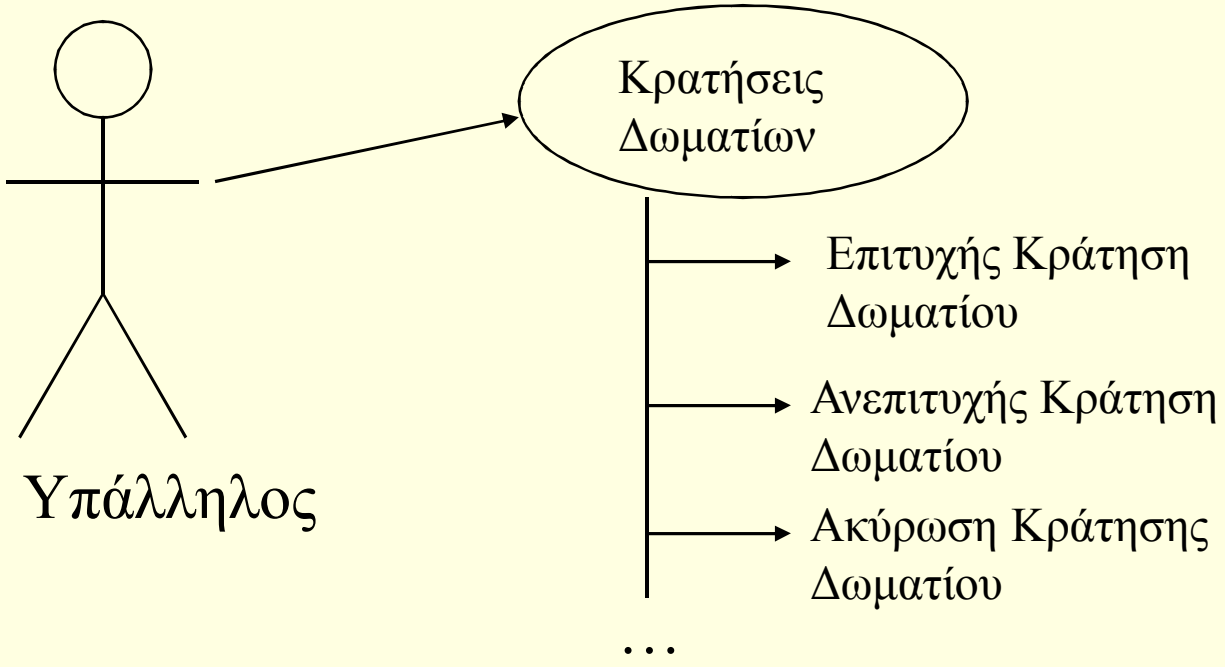
Περιγραφή Παραδείγματος

- Κράτηση δωματίου σε ξενοδοχείο
- Ο υπάλληλος δίνει τα στοιχεία του πελάτη το δωμάτιο (μονό, διπλό κλπ) και την περίοδο.
- Το σύστημα βρίσκει το δωμάτιο και κάνει κράτηση ή αποφαίνεται πως δεν υπάρχει κατάλληλο δωμάτιο για την περίοδο που προσδιορίστηκε

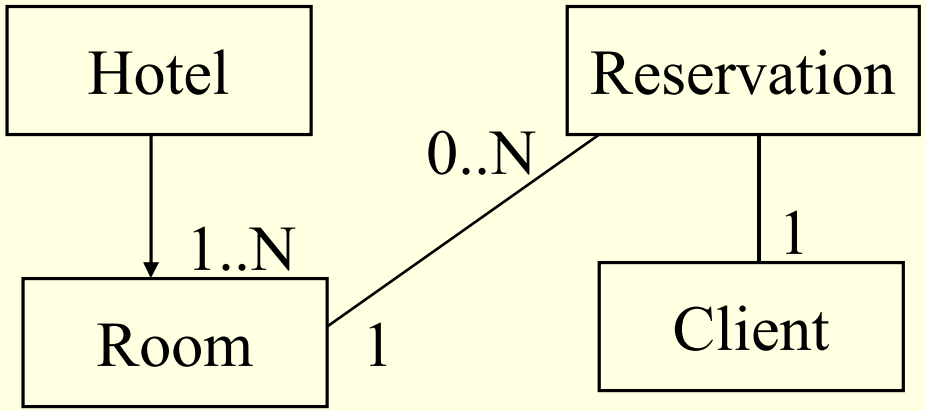
Σκοπός του Παραδείγματος

- Το παράδειγμα δεν αποσκοπεί στο να κάνουμε ένα πλήρες σύστημα κράτησης δωματίων σε ξενοδοχείο (π.χ. δεν θα ασχοληθούμε με την αποθήκευση των κρατήσεων σε Βάση Δεδομένων).
- Αποσκοπεί στο να καταλάβουμε:
  - Πως μπορούμε να χρησιμοποιήσουμε τα διαγράμματα ακολουθίας για να διαπιστώσουμε ποιες είναι οι κατάλληλες κλάσεις και λειτουργίες για τον προγραμματισμό ενός σεναρίου μιας περίπτωσης χρήσης.
  - Την ταυτόχρονη εξέλιξη ενός διαγράμματος κλάσεων προκειμένου να καλυφθούν οι απαιτήσεις μιας εφαρμογής
  - Την ροή των μηνυμάτων μεταξύ αντικειμένων

Περίπτωση Χρήσης και Σενάρια

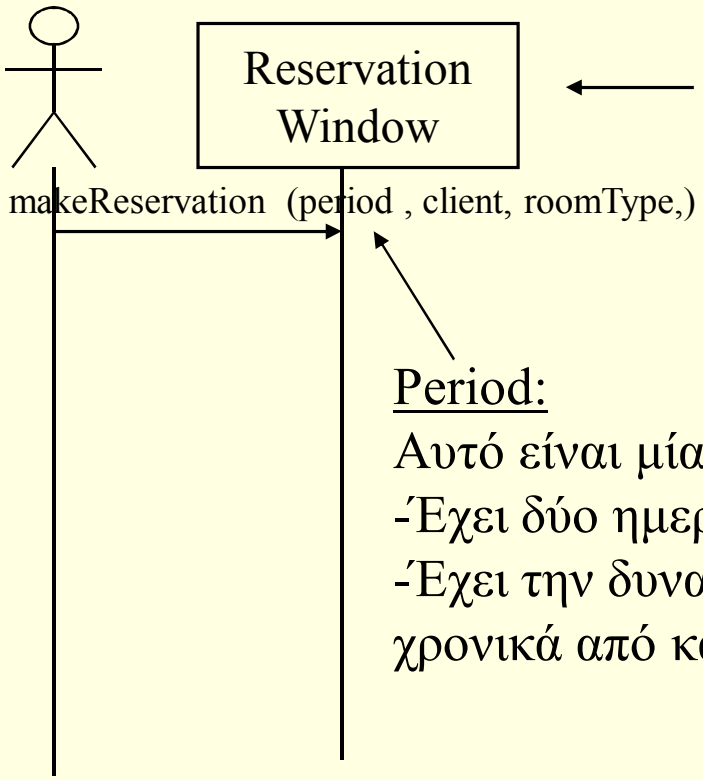


Αρχικό Διάγραμμα Κλάσεων



- Το ξενοδοχείο έχει πολλά δωμάτια
- Κάθε δωμάτιο έχει 0 ή περισσότερες κρατήσεις
- Κάθε κράτηση αφορά έναν πελάτη και ένα δωμάτιο

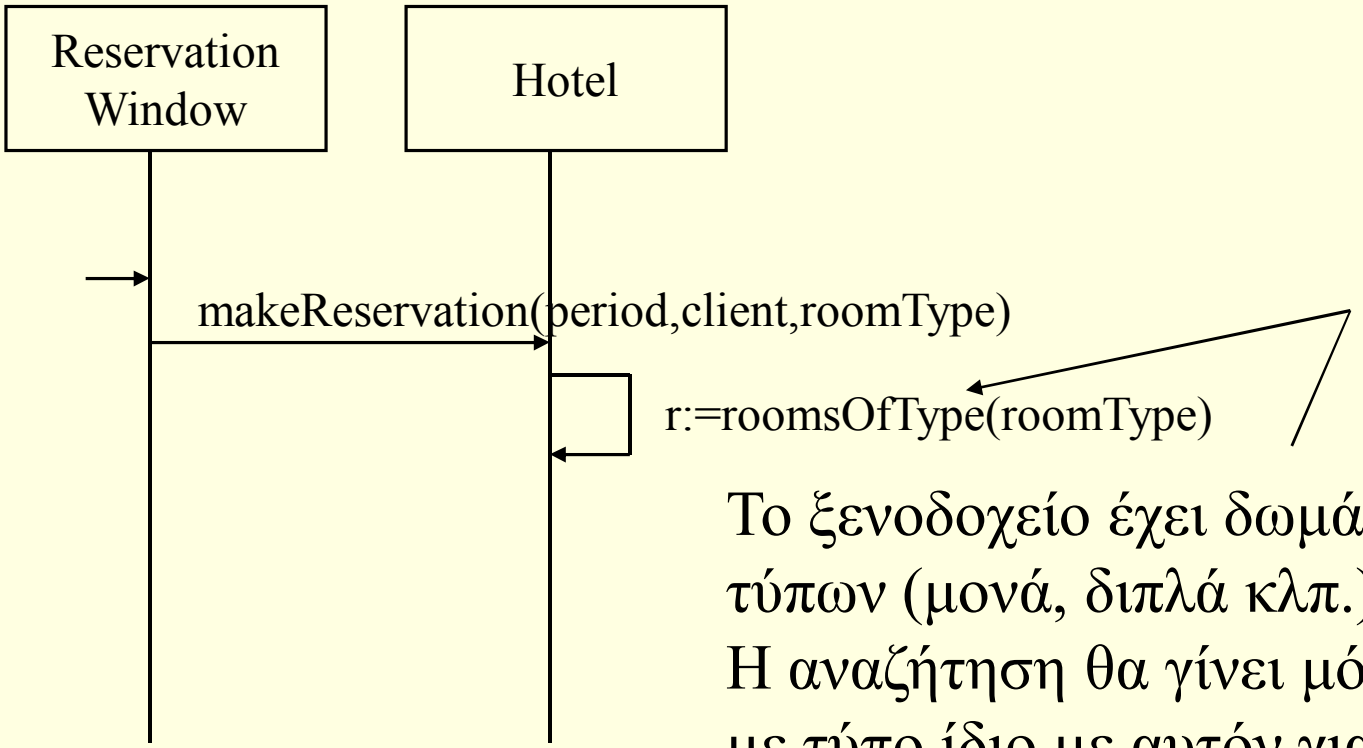
Διάγραμμα Ακολουθίας (1)



ReservationWindow:
Άλλη μία νέα κλάση.
Χωρίς γραφική διασύνδεση ο χρήστης δεν μπορεί να χρησιμοποιήσει το σύστημα.
Αυτές οι κλάσεις ονομάζονται «συνοριακές τάξεις»

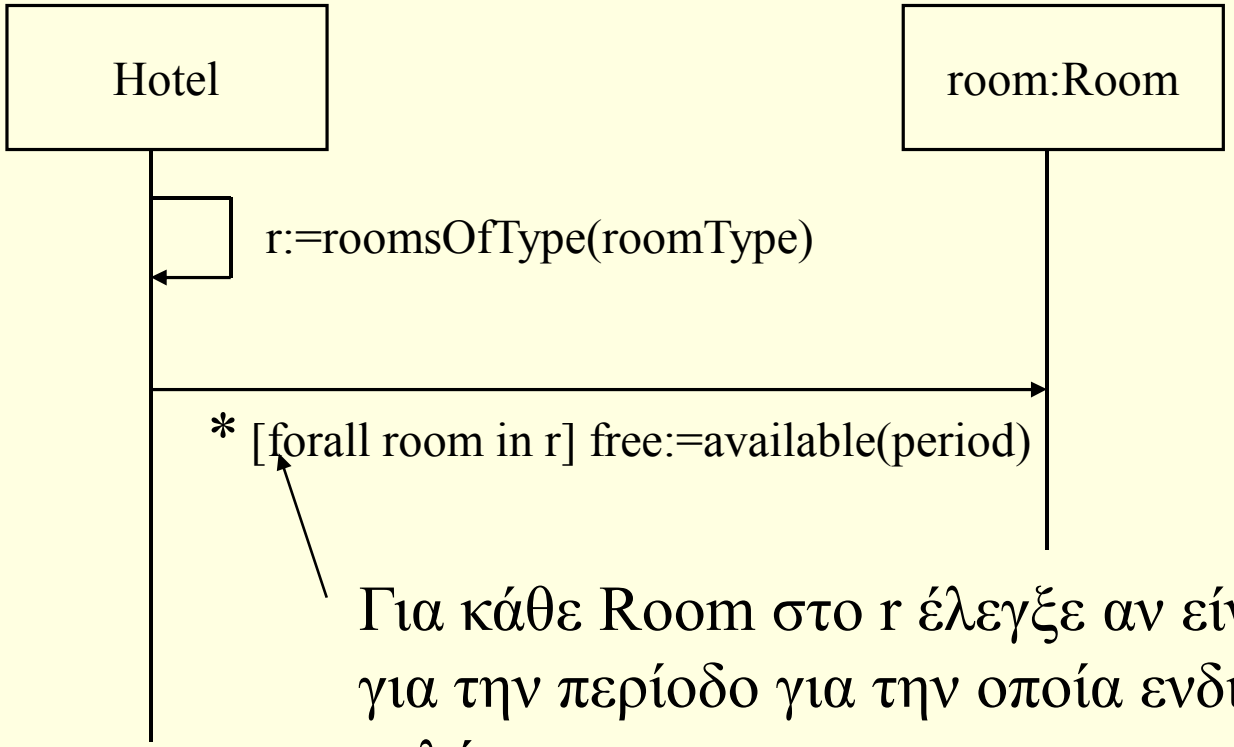
Period:
Αυτό είναι μία υποψήφια νέα κλάση;
-Έχει δύο ημερομηνίες (αρχής και τέλους).
-Έχει την δυνατότητα να μας πει αν μία περίοδος επικαλύπτεται χρονικά από κάποια άλλη περίοδο.

Διάγραμμα Ακολουθίας (2)



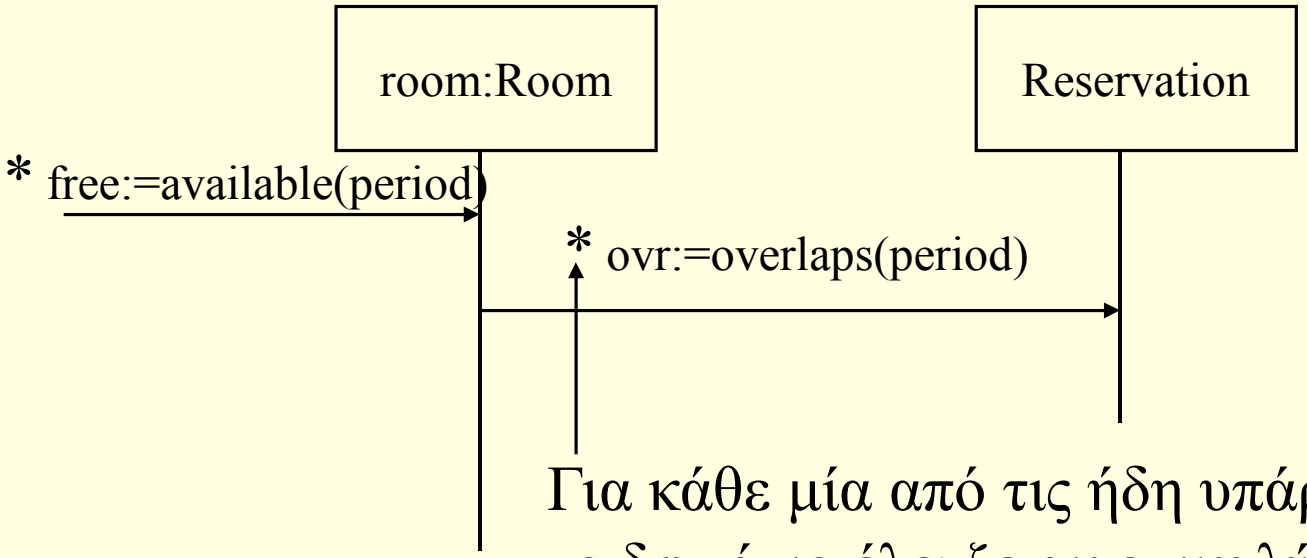
Το ξενοδοχείο έχει δωμάτια διαφόρων τύπων (μονά, διπλά κλπ.).
Η αναζήτηση θα γίνει μόνο στα δωμάτια με τύπο ίδιο με αυτόν για τον οποίο ενδιαφέρεται ο πελάτης

Διάγραμμα Ακολουθίας (3)



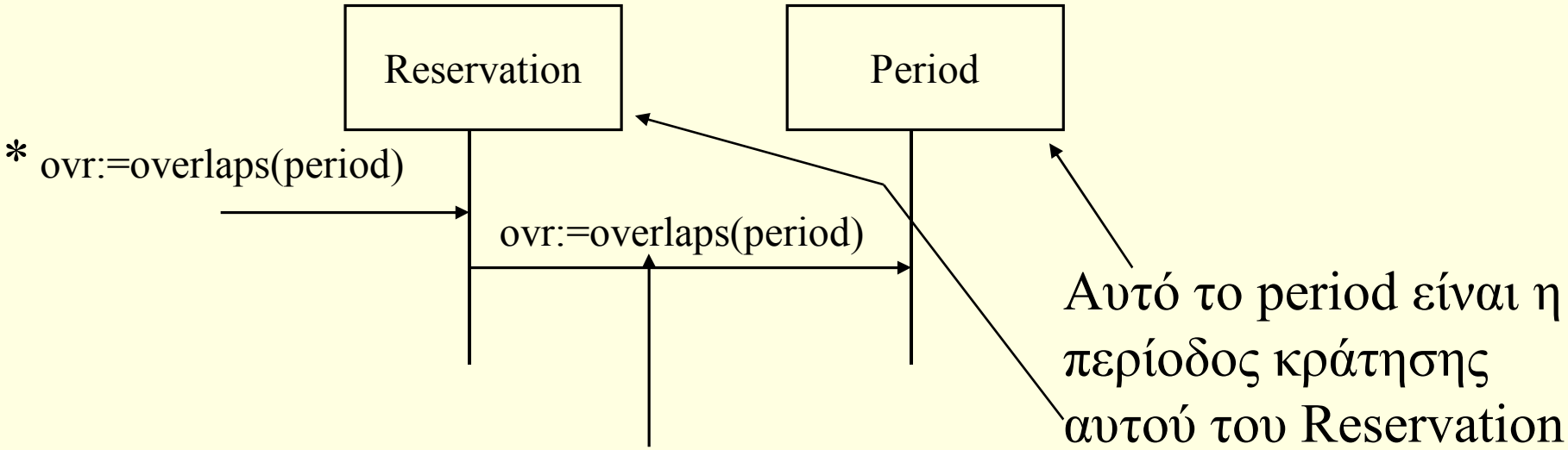
Για κάθε Room στο r έλεγξε αν είναι διαθέσιμο για την περίοδο για την οποία ενδιαφέρεται ο πελάτης.

Διάγραμμα Ακολουθίας (4)



Για κάθε μία από τις ήδη υπάρχουσες κρατήσεις για το δωμάτιο έλεγξε αν επικαλύπτει χρονικά την περίοδο για την οποία ενδιαφέρεται ο πελάτης

Διάγραμμα Ακολουθίας (5)

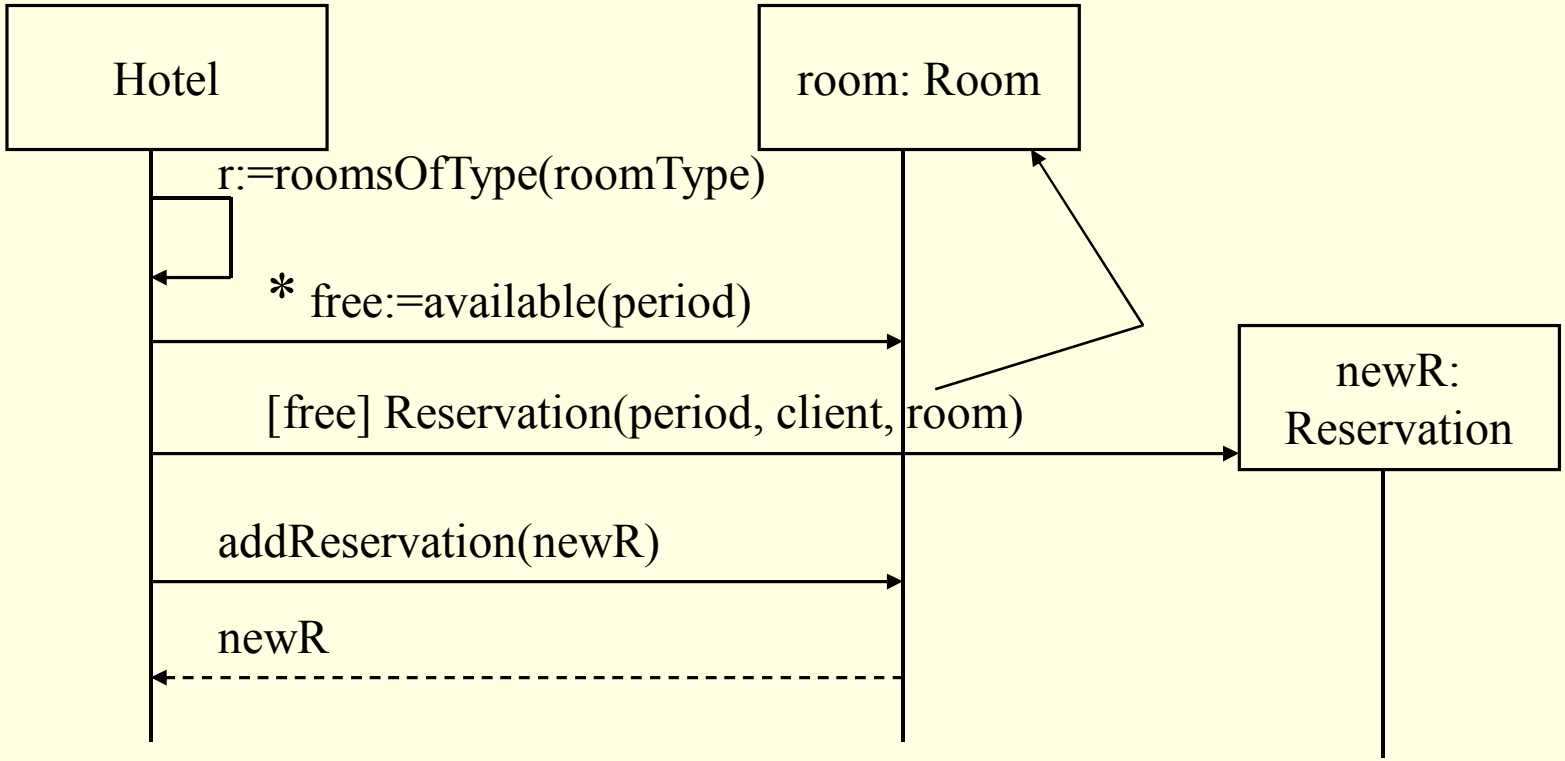


Η κράτηση ελέγχει αν επικαλύπτει χρονικά την περίοδο που ενδιαφέρει τον πελάτη, με την σύγκριση με την δική της περίοδο κράτησης.

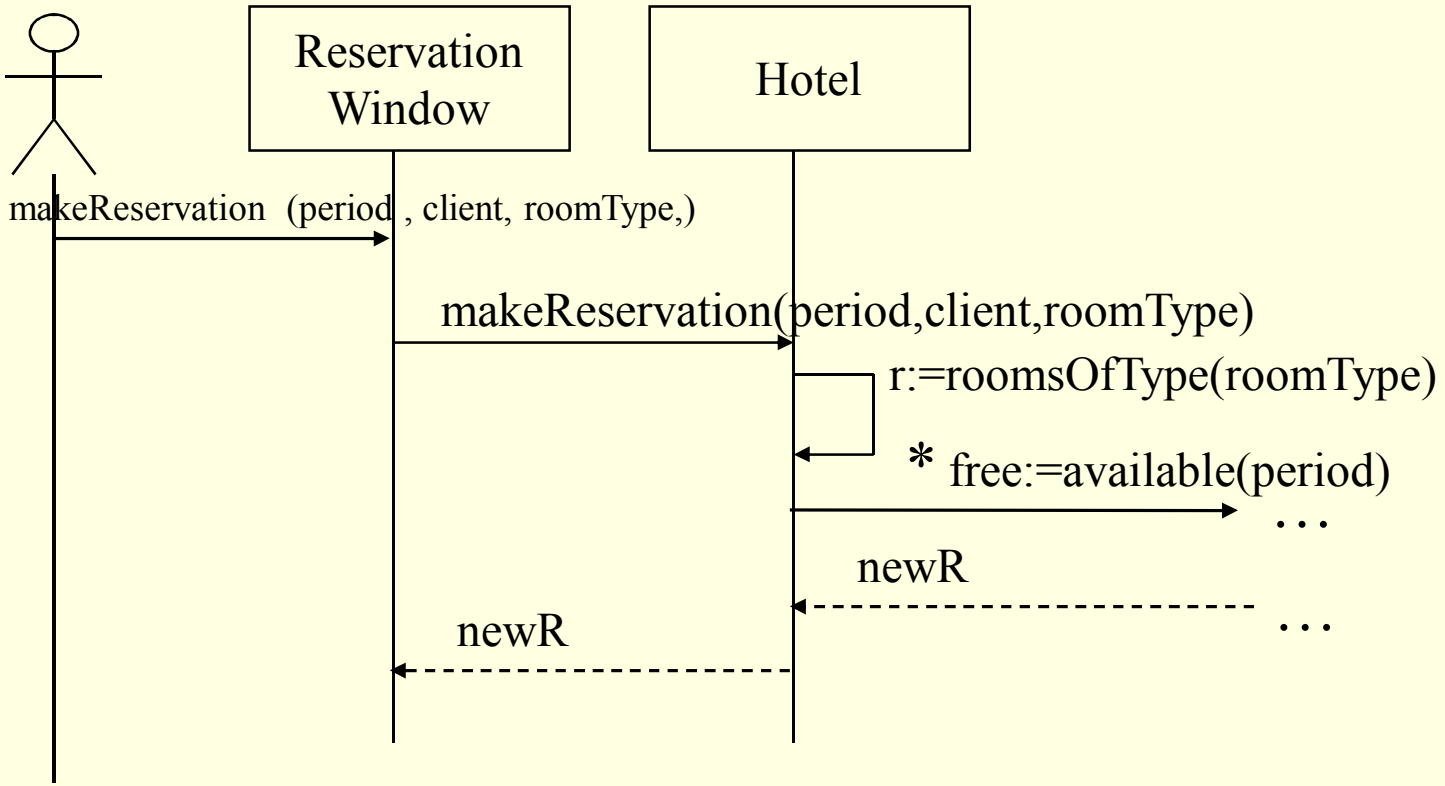
Επιτυχής Αναζήτηση

- Για κάποιο από τα δωμάτια η `available` θα επιστρέψει true για την περίοδο που ενδιαφέρει τον πελάτη.
- Σ' αυτή τη περίπτωση δημιουργείται μία νέα κράτηση για την περίοδο που ενδιαφέρει τον πελάτη και προστίθεται στις ήδη υπάρχουσες κρατήσεις γι' αυτό το δωμάτιο.

Διάγραμμα Ακολουθίας (6)

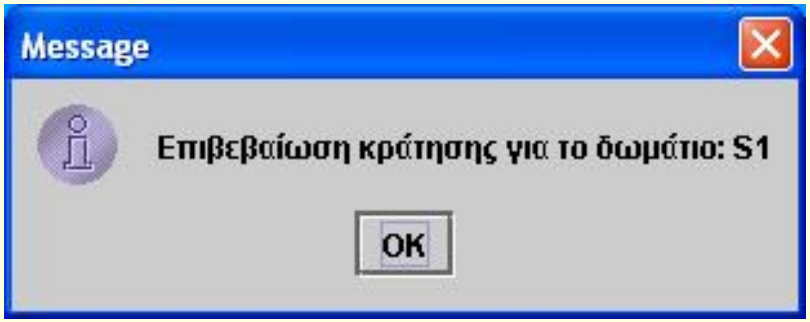


Διάγραμμα Ακολουθίας (7)

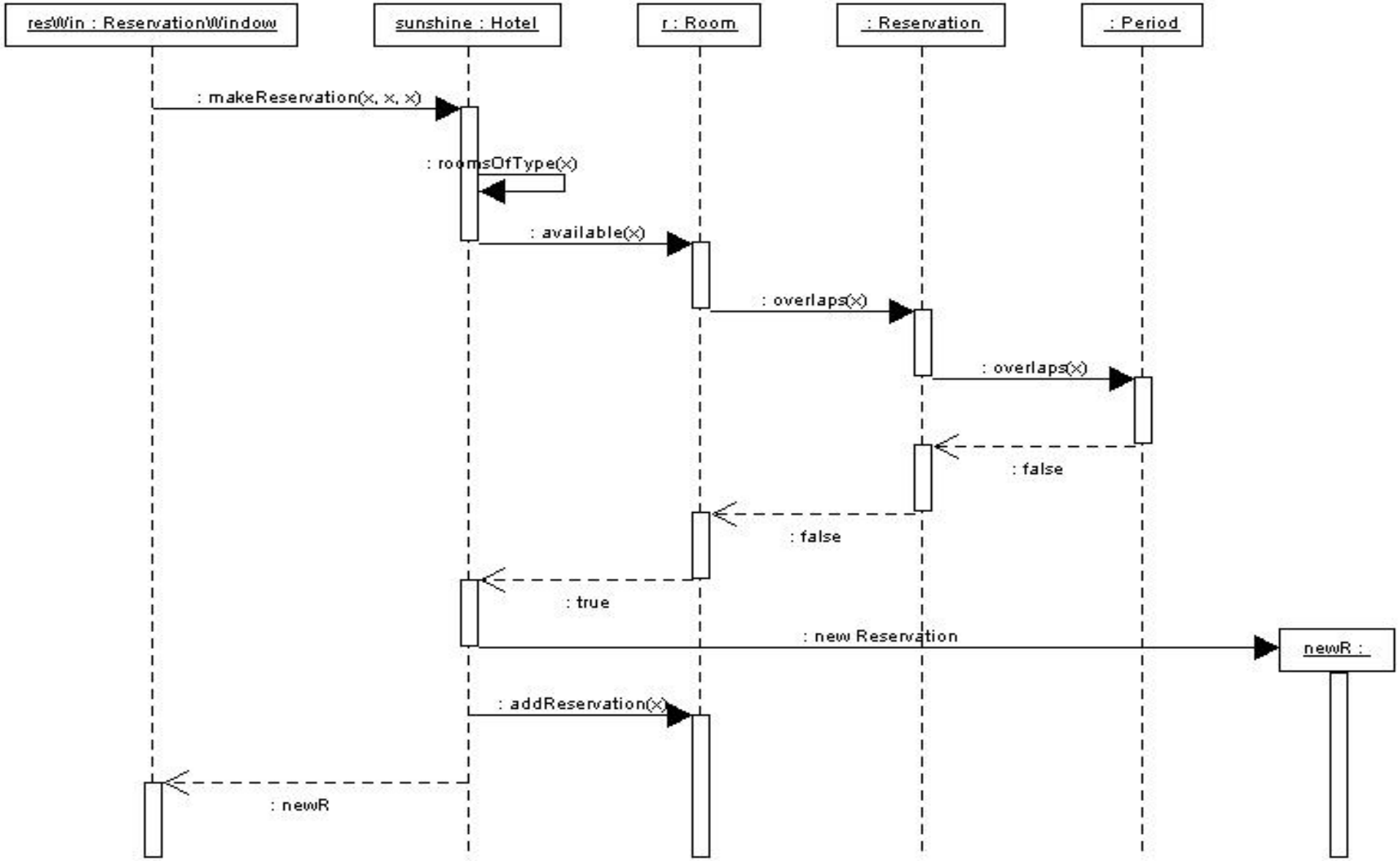


Τέλος Κράτησης

- Τελικά το ReservationWindow παίρνει την νέα κράτηση (newR) και εμφανίζει τις πληροφορίες της κράτησης στον υπάλληλο.
- Για παράδειγμα εμφανίζει ένα μήνυμα επιβεβαίωσης της κράτησης όπως το ακόλουθο:



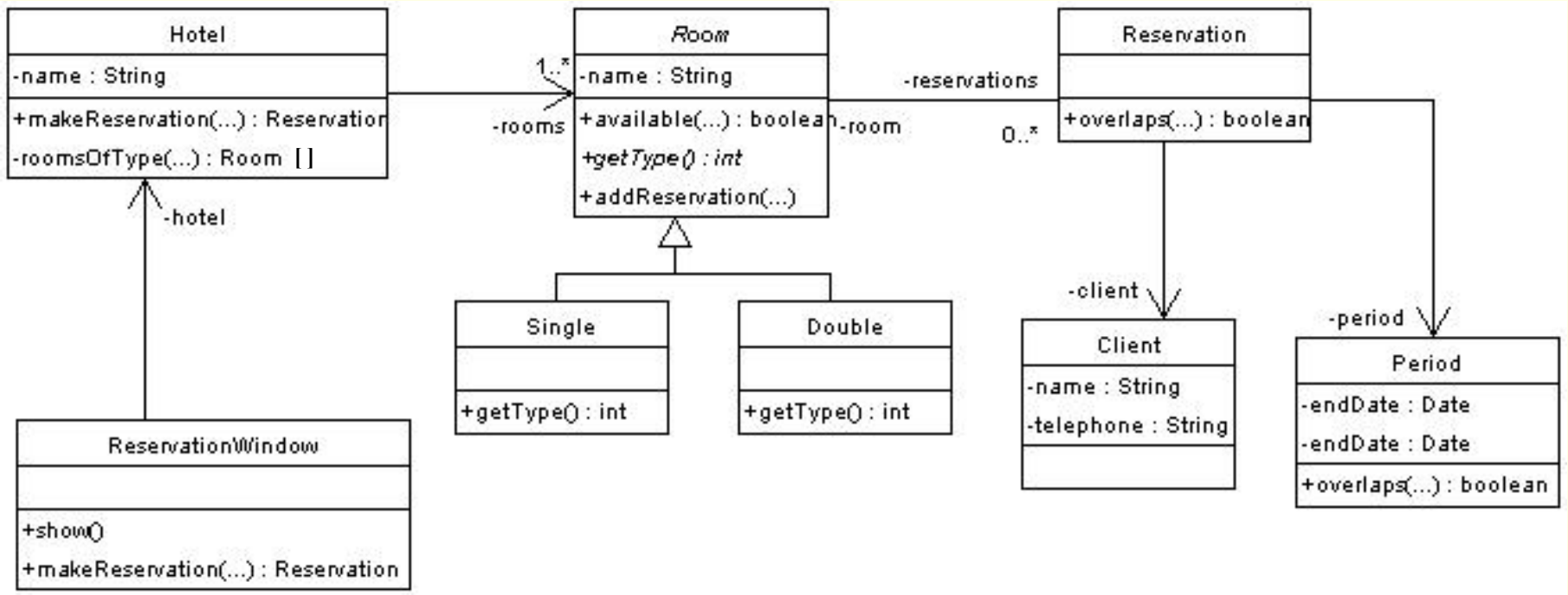
Διάγραμμα Ακολουθίας



Εμπλουτισμός του Διαγράμματος Κλάσεων

- Ξεκινήσαμε με ένα αρχικό διάγραμμα κλάσεων.
- Τώρα θα πρέπει να προσθέσουμε σε αυτό όλες τις νέες κλάσεις που ανακαλύψαμε (π.χ. Period, ReservationWindow) καθώς και τις μεθόδους που είναι απαραίτητες για την διεκπεραίωση της κράτησης (π.χ. την available στην τάξη Room, την overlaps στην τάξη Period κλπ.)

Διάγραμμα Κλάσεων



Πρόγραμμα Java

- Σημείωση: Η βηματική εκτέλεση σε debugger μας επιτρέπει να δούμε την ανταλλαγή των μηνυμάτων μεταξύ των αντικειμένων ακριβώς όπως αυτά απεικονίζονται στο διάγραμμα ακολουθίας.

Η κλάση Period

```
import java.util.Date;
public class Period {
    private Date startDate; private Date endDate;
    public Period(Date startDate, Date endDate) {
        this.startDate = startDate;
        this.endDate = endDate;
    }
    public Date getStartDate() { return startDate; }
    public Date getEndDate() { return endDate; }
    public boolean overlaps(Period p) {
        Date pStartD = p.getStartDate();
        Date pEndD = p.getEndDate();
        if ((pStartD.compareTo(startDate)>=0 &&
                        pStartD.compareTo(endDate)<=0) ||
           (pEndD.compareTo(startDate)>=0 && pEndD.compareTo(endDate)<=0))
        {
           return true;
        }
        return false;
    }

}
```

Η κλάση Room (1)

```
import java.util.Vector;
import java.util.Enumeration;

public abstract class Room {

    public final static int SINGLE_ROOM = 1;
    public final static int DOUBLE_ROOM = 2;

    private Vector reservations;
    private String name;

    public Room(String name) {
        reservations = new Vector();
        this.name = name;
    }

    public Reservation[] getReservations() {
        Object[] o = reservations.toArray();
        Reservation[] r = new Reservation[o.length];
        System.arraycopy(o, 0, r, 0, o.length);
        return r;
    }
. . .
```

Η κλάση Room (2)

```
public void addReservation(Reservation r) {
        reservations.add(r);
    }

    public boolean available(Period p) {
        Enumeration e = reservations.elements();

        while (e.hasMoreElements()) {
            Reservation r = (Reservation) e.nextElement();
            if (r.overlaps(p)) {
                return false;
            }
        }
        return true;
    }

    public String getName() {
        return name;
    }

    abstract public int getRoomType();
}
```

Οι κλάσεις DoubleRoom και SingleRoom

```
public class DoubleRoom extends Room {

    /** Creates a new instance of DoubleRoom */
    public DoubleRoom(String name) {
        super(name);
    }

    public int getRoomType() {
        return Room.DOUBLE_ROOM;
    }
} public class SingleRoom extends Room{
     /** Creates a new instance of SingleRoom */
     public SingleRoom(String name) {
         super(name);
      }
      public int getRoomType() {
          return Room.SINGLE_ROOM;
      }
}
```

Η κλάση Client

```
public class Client {
    private String name;
    private String telephone;
    /** Creates a new instance of Client */
    public Client(String name, String telephone) {
        this.name = name;
        this.telephone = telephone;
    }

    public String getName() { return name; }

    public String getTelephone() { return telephone;
  }

}
```

Η κλάση Reservation

```
public class Reservation {
    private Period period;
    private Client client;
    private Room room;

    public Reservation(Period period, Client client, Room room) {
        this.period = period;
        this.client = client;
        this.room = room;
    }

    public Client getClient() { return client; }

    public Period getPeriod() { return period; }

    public Room getRoom() { return room; }

    public boolean overlaps(Period p) {
        return period.overlaps(p);
    }

}
```

Άλλες κλάσεις

- Υπάρχουν επίσης άλλες δύο κλάσεις:
  - Η συνοριακή κλάση ReservationWindow που είναι το παράθυρο μέσω του οποίου γίνεται μία κράτηση
  - Η βασική κλάση Hotel που αποτελεί και το σημείο εισόδου στο πρόγραμμα