Σύντομο Παράδειγμα μιας συνοπτικής μεθοδολογίας ανάπτυξης

- Μεταφορά UML σε Java

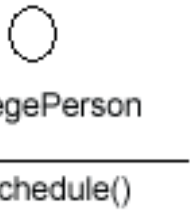
Μεταφορά της UML σε Java

Java	UML

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

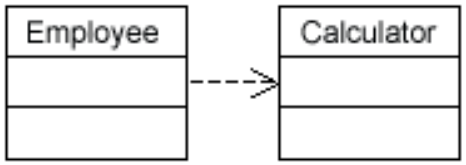
Μεταφορα της UML σε Java

Java	UML

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

Μεταφορα της UML σε Java

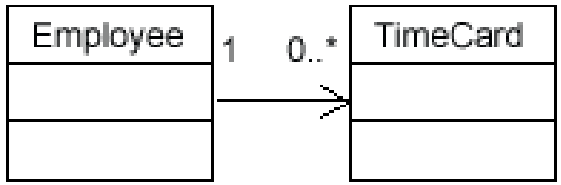
Java	UML

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

Μεταφορα της UML σε Java

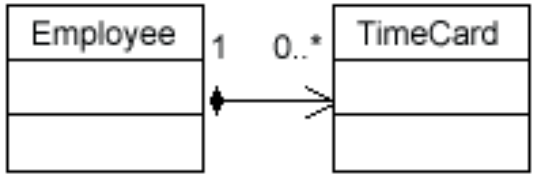
Θα έπρεπε να μπεί πίνακας

Java	UML

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

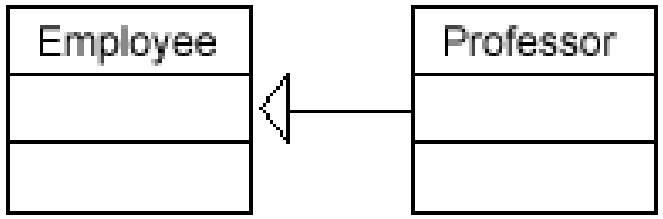
Μεταφορα της UML σε Java

Java	UML

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

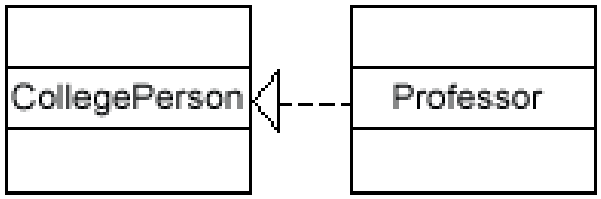
Μεταφορα της UML σε Java

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=value;}
        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

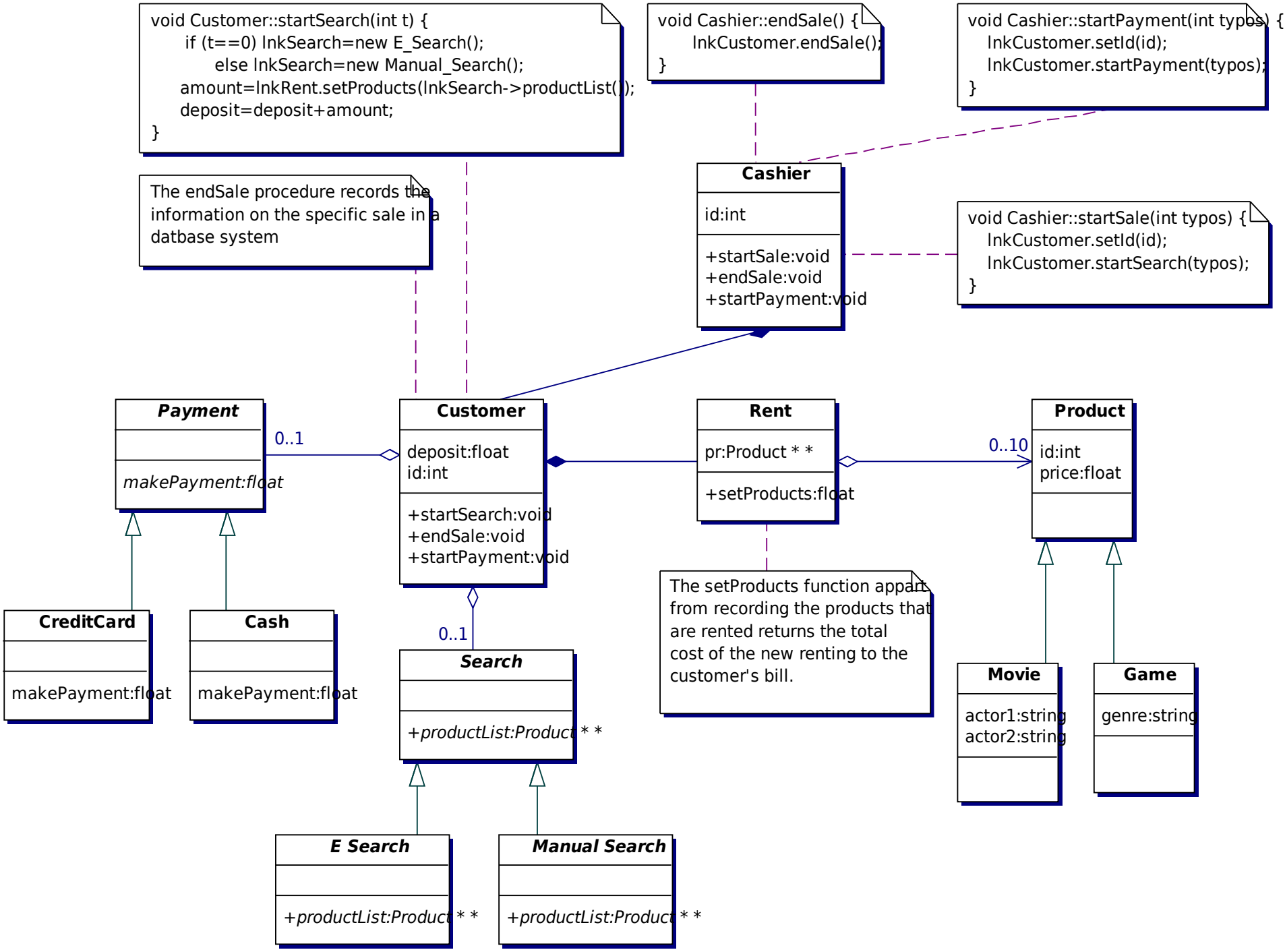
```
class Employee
{
        public:
        Employee();
                void Hire() ;
                void Fire();
                virtual double getSalary();
        private:
        string firstname;
        string lastname;
}
```

C++ σε UML

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

Παράδειγμα: DVD-club

- Το σύστημα αυτό θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:
- Εισαγωγή στο σύστημα (log in).
- Ενοικίαση (ταινίες, παιχνίδια).
- Διαχείριση συνδρομών πελατών.
- Διαχείριση υλικού (ταινίες, παιχνίδια).
- Διαχείριση χρηστών.
- Πληρωμή.
- Πληρωμή με πιστωτική κάρτα.
- Πληρωμή με μετρητά.
- Πληρωμή επιβάρυνσης λόγω καθυστέρησης επιστροφής.


void Customer::startSearch(int t) {
if (t==0) lnkSearch=new E_Search();
else lnkSearch=new Manual_Search();
amount=lnkRent.setProducts(lnkSearch->productList());
deposit=deposit+amount;
}
void Cashier::endSale() {
lnkCustomer.endSale();
}
void Cashier::startPayment(int typos) {
lnkCustomer.setId(id);
lnkCustomer.startPayment(typos);
}
The endSale procedure records the information on the specific sale in a datbase system
Cashier
id:int
+startSale:void
+endSale:void
+startPayment:void
void Cashier::startSale(int typos) {
lnkCustomer.setId(id);
lnkCustomer.startSearch(typos);
}
Payment
makePayment:float
0..1
Customer
deposit:float
id:int
+startSearch:void
+endSale:void
+startPayment:void
Rent
pr:Product * *
+setProducts:float
0..10
Product
id:int
price:float
CreditCard
makePayment:float
Cash
makePayment:float
0..1
Search
+productList:Product * *
The setProducts function apart from recording the products that are rented returns the total cost of the new renting to the customer's bill.
Movie
actor1:string
actor2:string
Game
genre:string
E Search
+productList:Product * *
Manual Search
+productList:Product * *

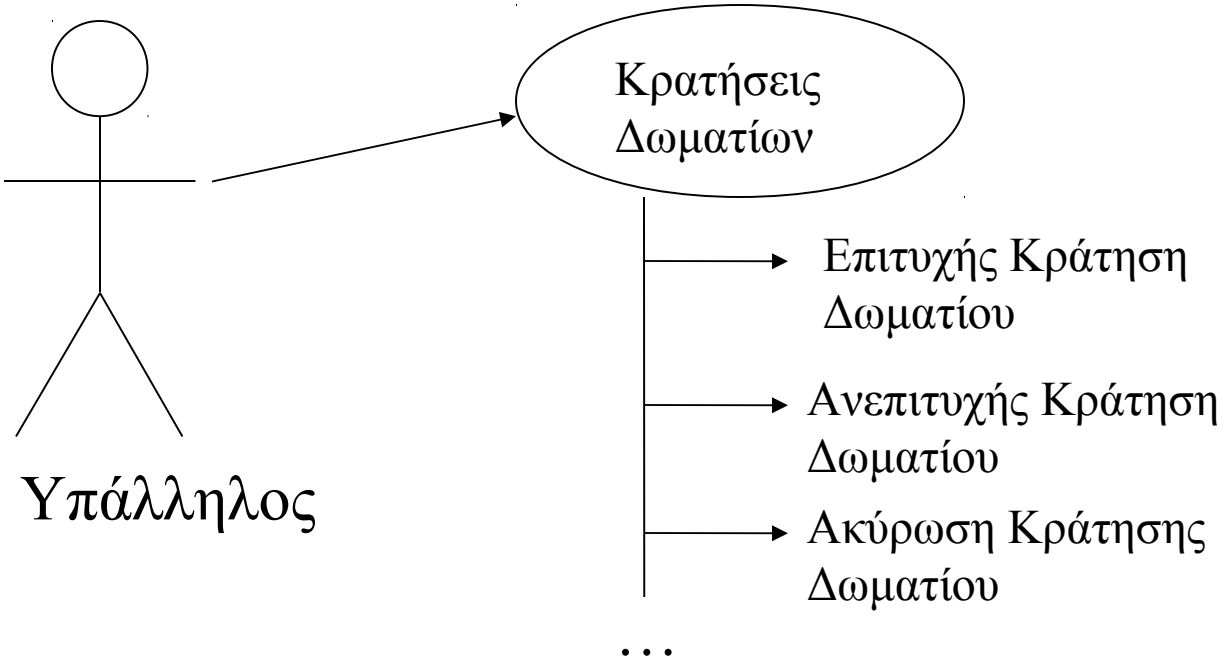
Παράδειγμα: Ξενοδοχείο

- 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

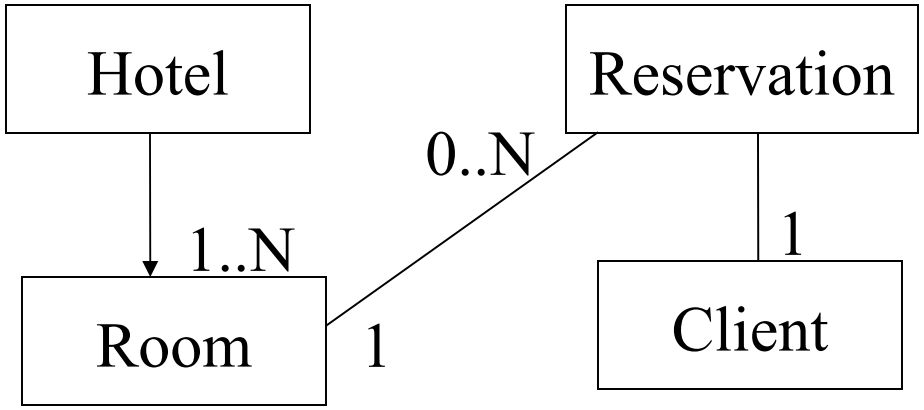
Σκοπός του Παραδείγματος

- 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 τάξεις 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

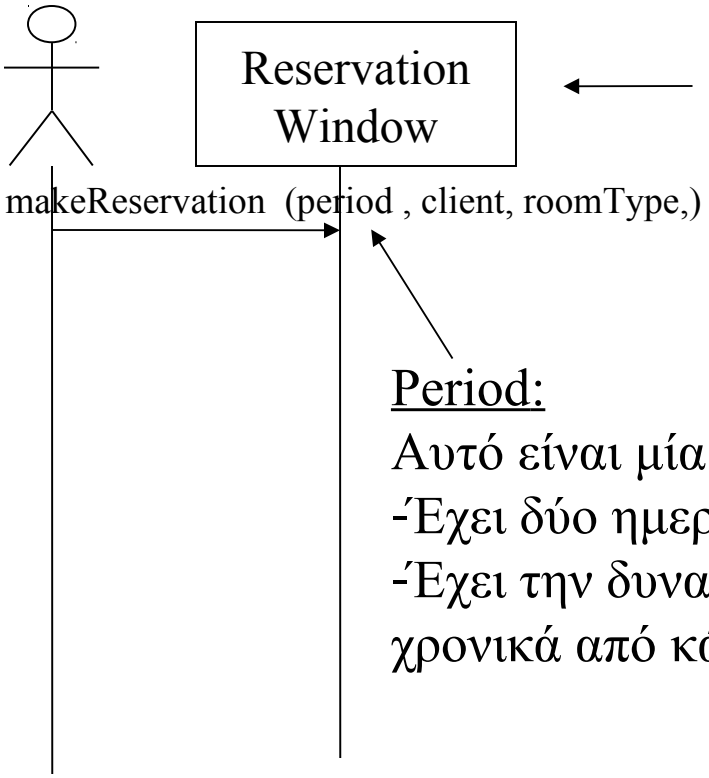


Αρχικό Διάγραμμα Κλάσεων



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

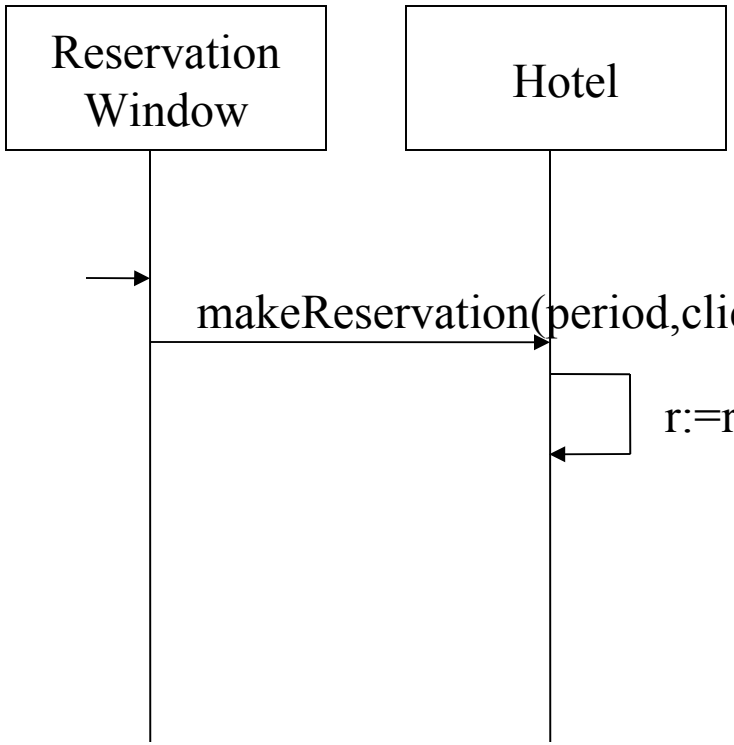
Διάγραμμα Ακολουθίας (1)



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

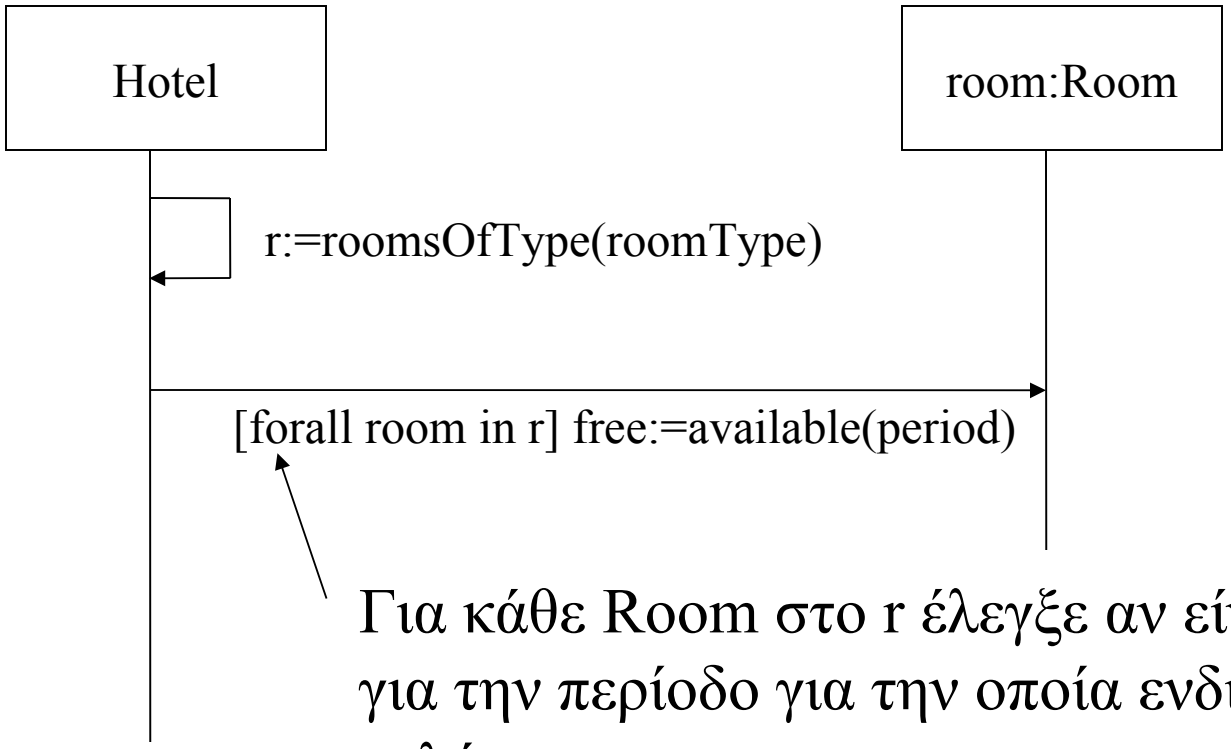
Period:
Αυτό είναι μία υποψήφια νέα κλάση;
-Έχει δύο ημερομηνίες (αρχής και τέλους).
-Έχει την δυνατότητα να μας πει αν μία περίοδος επικαλύπτεται χρονικά από κάποια άλλη περίοδο.

Διάγραμμα Ακολουθίας (2)



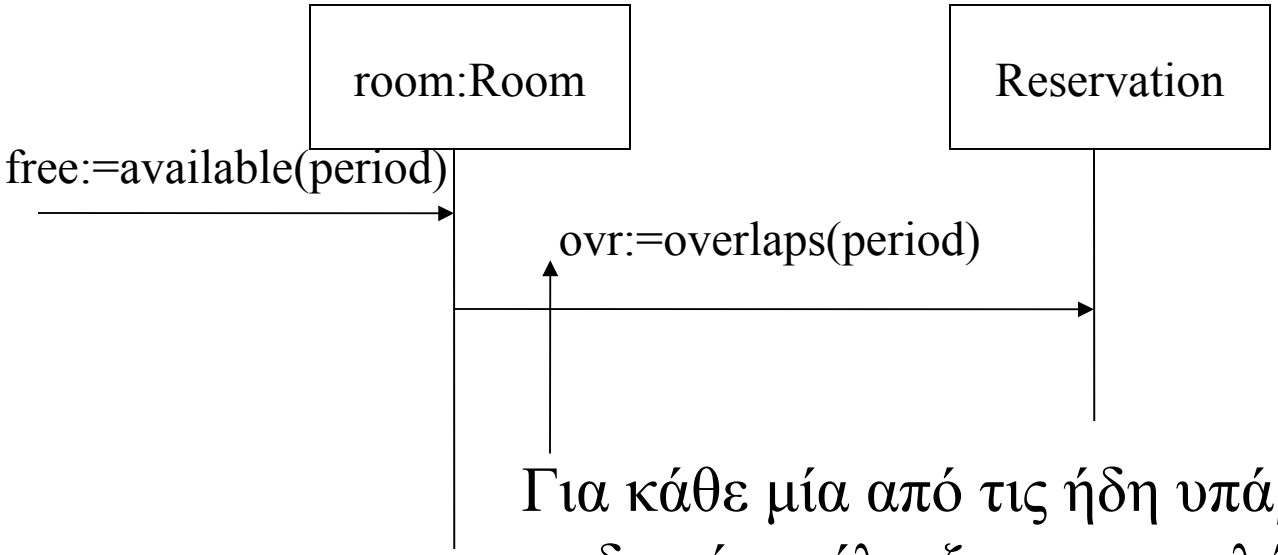
Το ξενοδοχείο έχει δωμάτια διαφόρων τύπων (μονά, διπλά κλπ.).
Η αναζήτηση θα γίνει μόνο στα δωμάτια με τύπο ίδιο με αυτόν για τον οποίο ενδιαφέρεται ο πελάτης

Διάγραμμα Ακολουθίας (3)



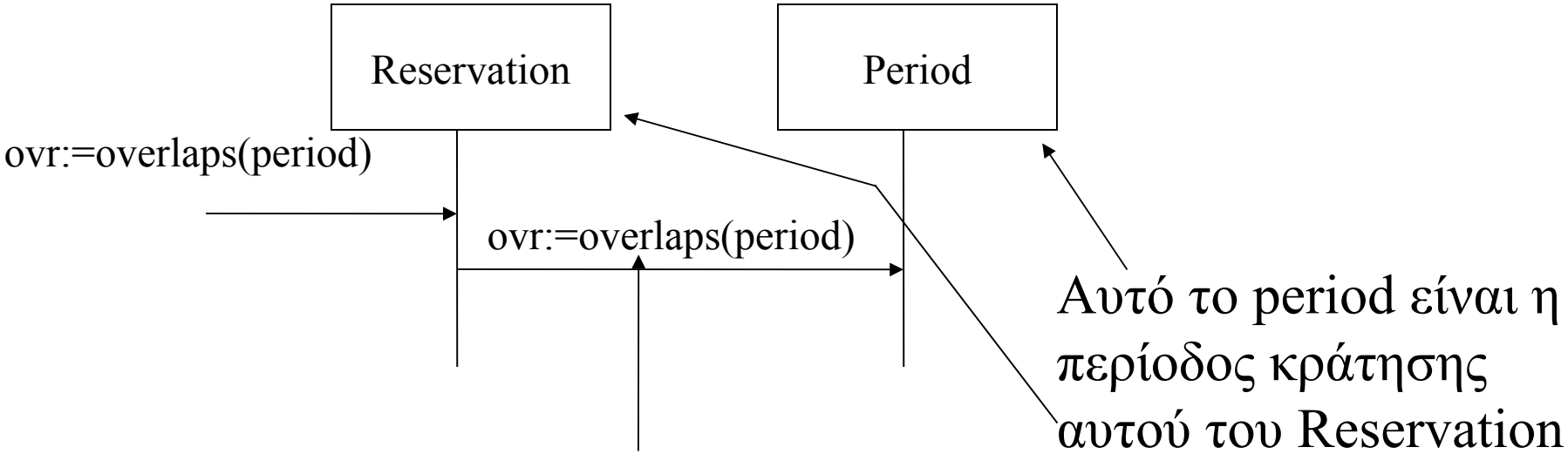
Για κάθε Room στο r έλεγξε αν είναι διαθέσιμο για την περίοδο για την οποία ενδιαφέρεται ο πελάτης.

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)

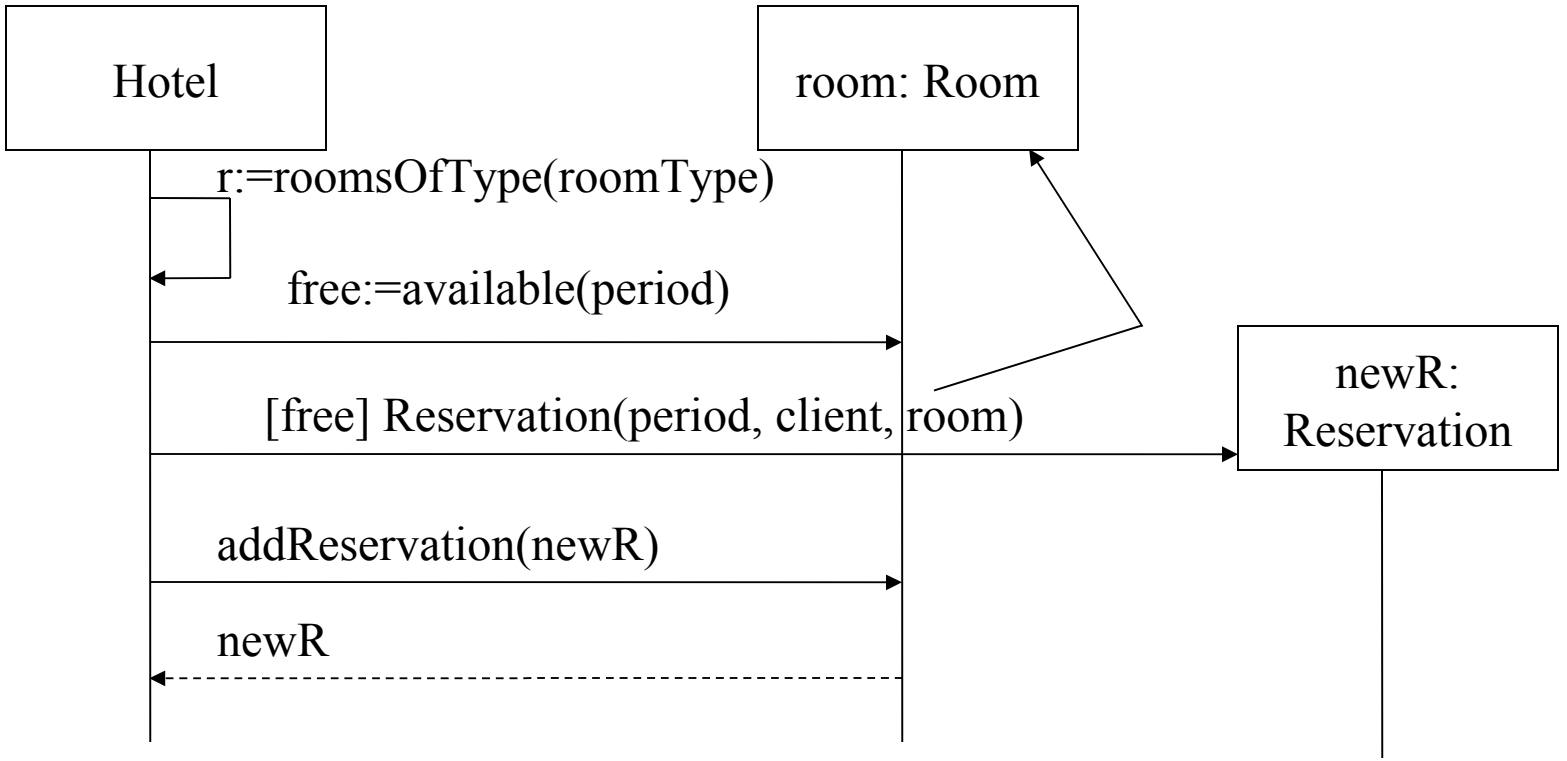


Η κράτηση ελέγχει αν επικαλύπτει χρονικά την περίοδο που ενδιαφέρει τον πελάτη, με την σύγκριση με την δική της περίοδο κράτησης.

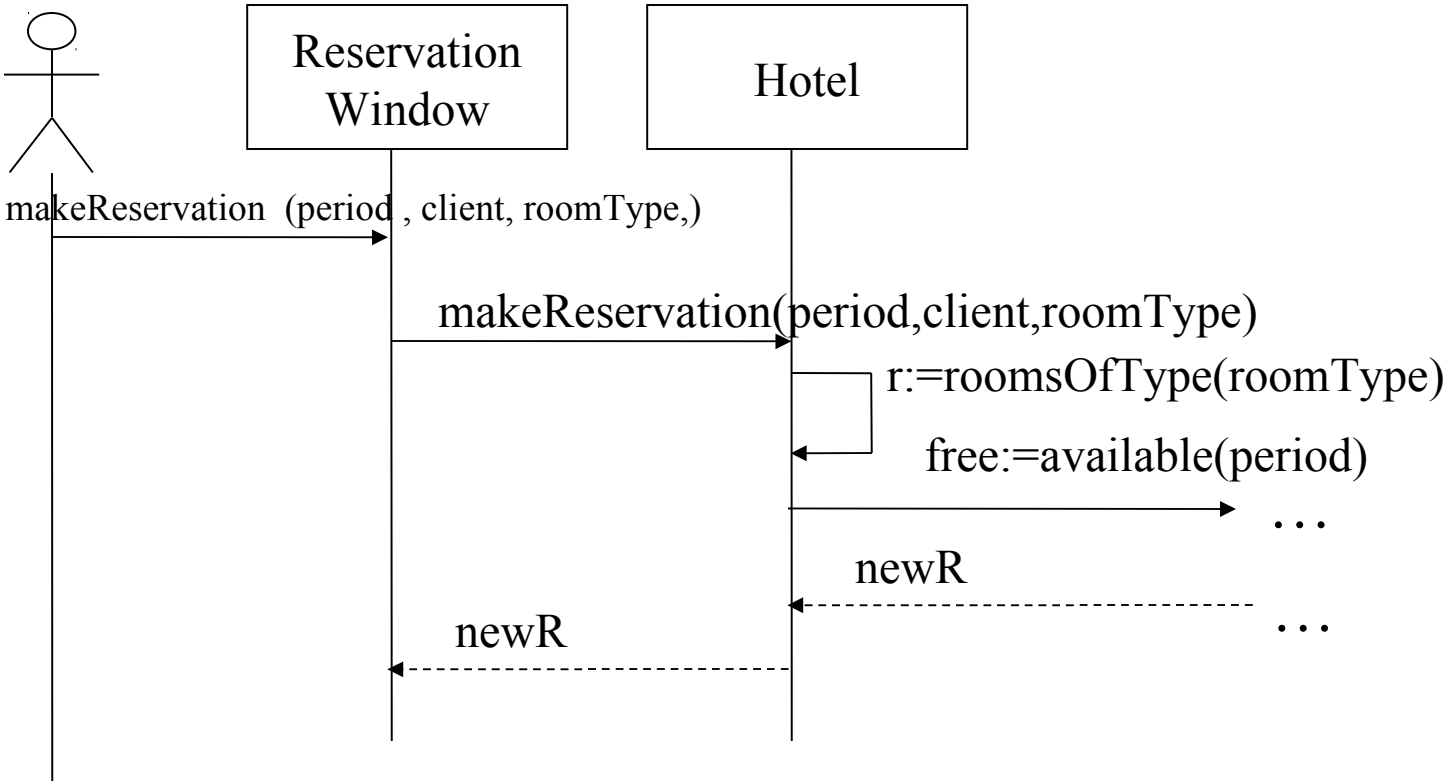
Επιτυχής Αναζήτηση

- 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)

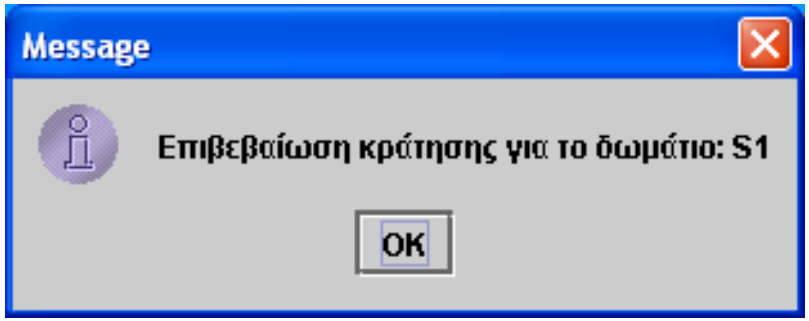


Διάγραμμα Ακολουθίας (7)

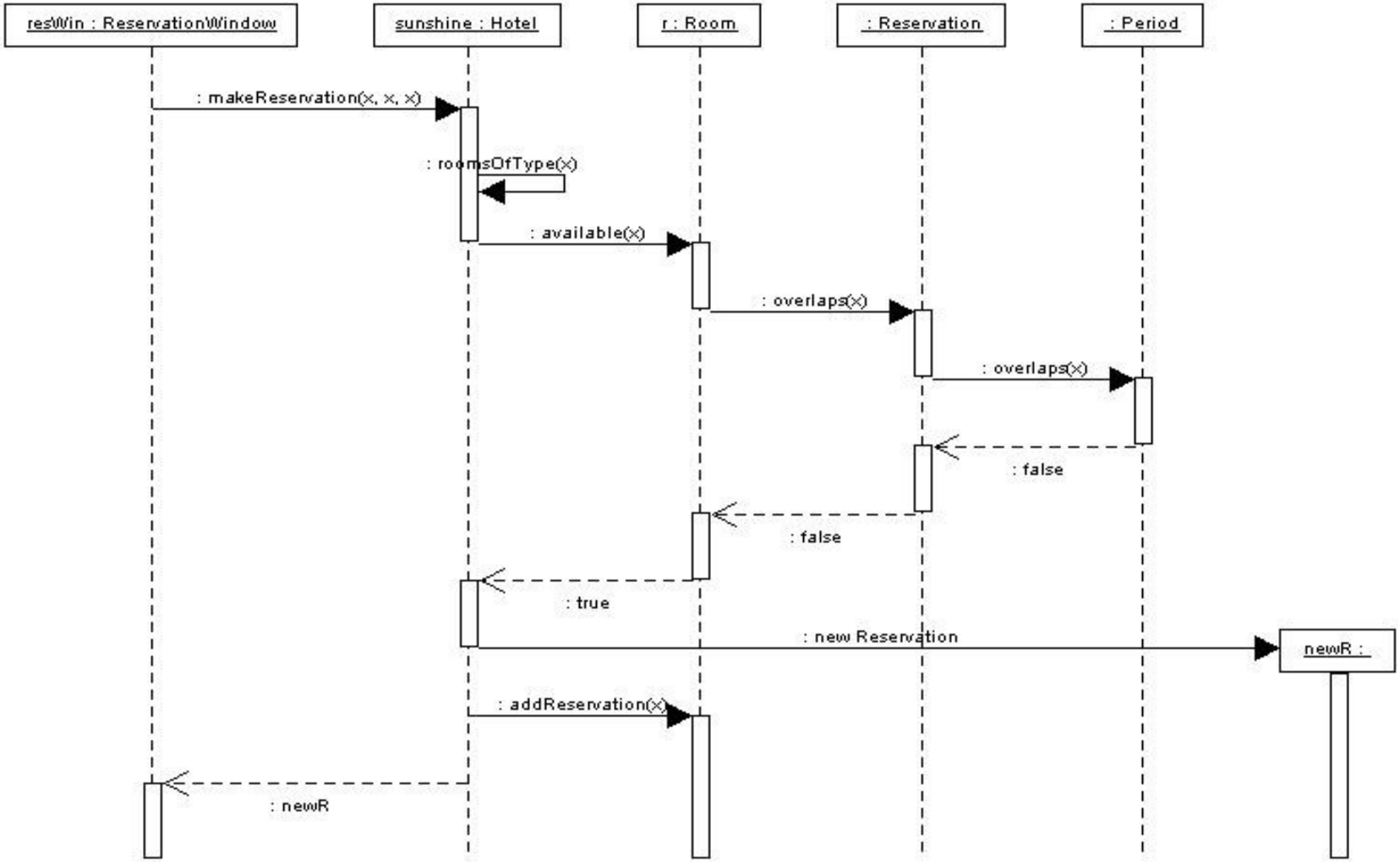


Τέλος Κράτησης

- 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:



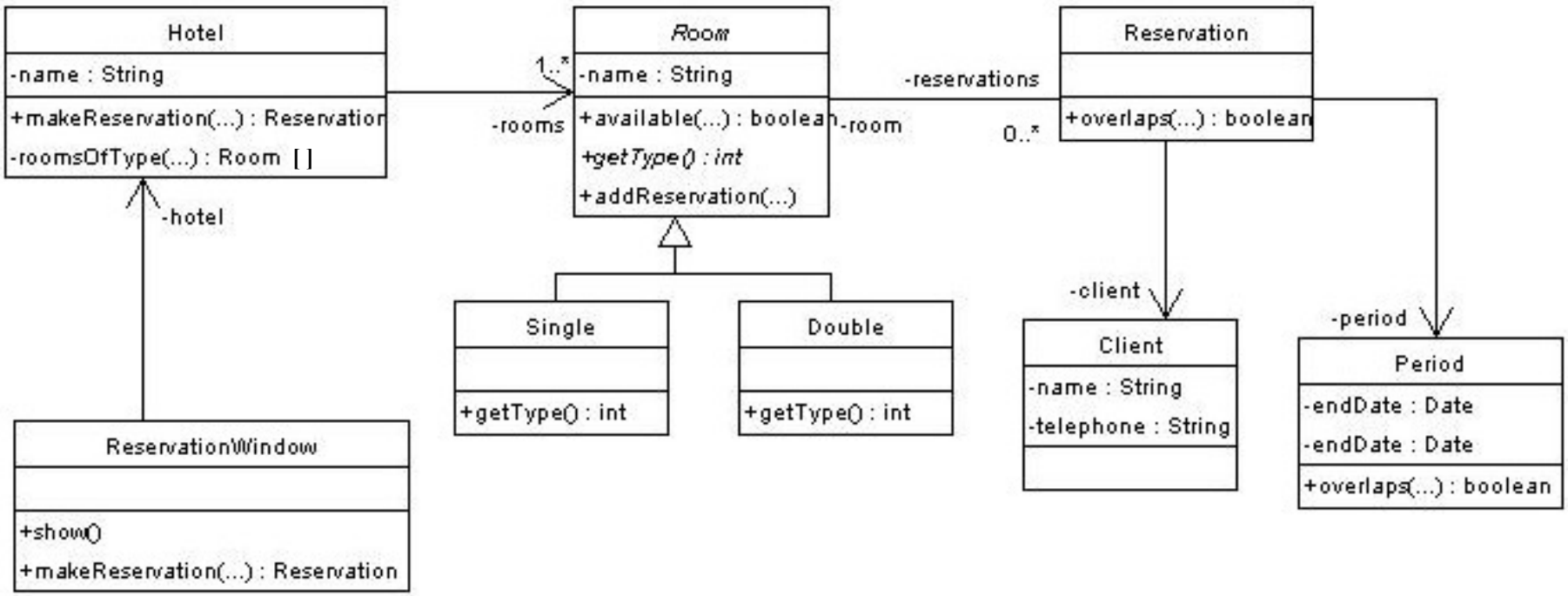
Διάγραμμα Ακολουθίας



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

Ασκήσεις

- Κάντε το διάγραμμα ακολουθίας για την «Ακύρωση Κράτησης» και εμπλουτίστε αντίστοιχα το διάγραμμα τάξεων
- Εξηγείστε σε ποια σημεία θα πρέπει να προστεθεί η πρόσβαση σε Βάση Δεδομένων για την αποθήκευση και ανάκτηση των κρατήσεων
- Πότε θα πρέπει να διαγραφεί μία κράτηση φυσιολογικά (όχι από ακύρωση);