

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εργασία 2

ΠΑΝΤΕΛΕΗΜΩΝ ΠΡΩΙΟΣ

ice18390023

6ο Εξάμηνο

ice18390023@uniwa.gr

Τμήμα ΣΤ5 Τετάρτη 14:00-16:00



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

Υπεύθυνοι καθηγητές

ΜΑΜΑΛΗΣ ΒΑΣΙΛΗΣ

ΠΑΝΤΖΙΟΥ ΓΡΑΜΜΑΤΗ

ΔΟΚΑ ΑΙΚΑΤΕΡΙΝΗ

Τμήμα Μηχανικών και Πληροφορικής Υπολογιστών
30 Μαΐου 2021

Περιεχόμενα

1	To interface ClientInterface	1
2	To interface ServerInterface	2
3	Η κλάση Zone	5
4	Η κλάση Ticket	7
5	Η κλάση Notify	9
6	Η κλάση TheaterSeats	11
7	Η κλάση TheaterServer	16
8	Η κλάση TheaterClient	19

Κώδικες

1.1	Κώδικας ClientInterface	1
2.1	Κώδικας ServerInterface	2
3.1	Κώδικας Zone	5
4.1	Κώδικας Ticket	7
5.1	Κώδικας Notify	9
6.1	Κώδικας TheaterSeats	11
7.1	Κώδικας TheaterServer	16
8.1	Κώδικας TheaterClient	20

Κατάλογος σχημάτων

7.1	To terminal του server	16
8.1	Ενδεικτικό τρέξιμο του client A	19
8.2	Ενδεικτικό τρέξιμο του client B	20
8.3	Ενδεικτικό τρέξιμο του client Γ	20

1 To interface ClientInterface

To ClientInterface (κώδικας 1.1) επεκτείνεται από την κλάση TheaterClient έτσι ώστε ο remote server να μπορεί να καλέσει τον client για να τον ενημερώσει

```
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3
4 /**
5  * @author padelis
6  *
7  * To ClientInterface επεκτείνει την κλάση Remote
8  * έτσι ώστε να μπορεί να κληθεί από κάποιο
9  * μη τοπικό VM αν γίνει κλήση στην μέθοδο notifyCustomer
10 *
11 */
12 public interface ClientInterface extends Remote
13 {
14     public void notifyCustomer(int number) throws RemoteException;
15 }
```

Κώδικας 1.1: Κώδικας ClientInterface

2 To interface ServerInterface

To interface `ServerInterface` (κώδικας 2.1) είναι η διεπαφή όπου ο client μπορεί να καλεί μεθόδους του server. Έχει 5 μεθόδους οι οποίες είναι

1. Η `list`, όπου επιστρέφει μια λίστα των ζωνών και της διαθεσιμότητάς τους
2. Η `book`, όπου κάνει κράτηση αναλόγως των παραμέτρων
3. Η `guests`, όπου επιστρέφει την λίστα των καλεσμένων
4. Η `cancel`, ακυρώνει κάποια υπάρχουσα κράτηση
5. Η `addQueue`, όπου προσθέτει στην λίστα αναμονής κάποιον client

```
1 import java.rmi.Remote;
2 import java.rmi.RemoteException;
3 import java.util.ArrayList;
4
5 /**
6  * @author padelis
7  *
8  * To ServerInterface επεκτείνει την κλάση Remote
9  * έτσι ώστε να μπορεί να κληθεί από κάποιο
10 * μη τοπικό VM αν γίνει κλήση σε κάποια μέθοδο
11 *
12 */
13 public interface ServerInterface extends Remote
14 {
15     /**
16      * Η μέθοδος list(), επιστρέφει την λίστα με τις
17      * ελεύθερες θέσεις κάθε ζώνης.
18      *
19      * @return ArrayList<Zone>
20      * @throws RemoteException
21      */
22     public ArrayList<Zone> list() throws RemoteException;
23
24     /**
25      * Η μέθοδος book κάνει κράτηση αναλόγως
26      * των παραμετρών της και επιστρέφει
27      * zero: αν δεν υπάρχει καμία θέση
28      * price: αν έγινε η κράτηση και επιστρέφει το κόστος
29      * που διαχωρίζεται με ';'
30      * avail: αν υπάρχει μικρότερος αριθμός διαθέσιμων θέσεων
31      * από όσες ζητήθηκαν αρχικά
```

```
32 * unknown: αν ο τύπος θέσεων είναι άγνωστη
33 *
34 *
35 * @param type, ο τύπος θέσεις
36 * @param number, ο αριθμός θέσεων
37 * @param name, το όνομα κράτησης
38 * @return "price;<cost>"
39 * @return "zero"
40 * @return "avail;<available seats>"
41 * @return "unknown"
42 * @throws RemoteException
43 */
44 public String book(String type, int number, String name) throws RemoteException;
45
46 /**
47 * Η μέθοδος guests, επιστρέφει μια λίστα με αυτούς που
48 * έχουν κλήσει εισιτήρια.
49 *
50 * @return ArrayList<Ticket>
51 * @throws RemoteException
52 */
53 public ArrayList<Ticket> guests() throws RemoteException;
54
55 /**
56 * Η μέθοδος cancel, ακυρώνει μια κράτηση βάσει των παραμέτρων
57 * που δόθηκαν.
58 *
59 * @param type
60 * @param number
61 * @param name
62 * @return false αν δεν υπάρχει κάτι τέτοιος αλλιώς true
63 * @throws RemoteException
64 */
65 public boolean cancel(String type, int number, String name) throws RemoteException;
66
67 /**
68 * Η μέθοδος προσθέτει στην αναμονή κάποιον χρήστη που θέλει
69 * να περιμένει ενημέρωση εάν υπάρχει ελεύθερος ο αριθμός
70 * θέσεων όπου ζήτησε αρχικά.
71 *
72 * @param cli
73 * @param type
74 * @param number
75 * @throws RemoteException
```

```
76  */  
77  public void addQueue(ClientInterface cli, String type, int number) throws RemoteException;  
78  }
```

Κώδικας 2.1: Κώδικας ServerInterface

3 Η κλάση Zone

Η κλάση Zone (κώδικας 3.1) επεκτείνει το Serializable, διότι κάποιες φορές επιστρέφεται στον απομακρυσμένο client (επίσης ότι αλλαγή κάνει ο client δεν αλλάζει στο αντικείμενο που γνωρίζει ο server). Η κλάση αυτή κρατάει μερικά δεδομένα για μία ζώνη όπως

1. **maxCapacity** η μέγιστη χωρητικότητα των θέσεων της ζώνης
2. **number** οι ελεύθερες θέσεις της ζώνης
3. **type** το όνομα τύπου της ζώνης
4. **price** το κόστος μίας θέσης της ζώνης

```
1 import java.io.Serializable;
2
3 /**
4  * @author padelis
5  *
6  * Η κλάση Zone υλοποιεί την Serializable
7  * γιατί επιστρέφει και στον client που είναι
8  * απομακρυσμένος και δεν βρήσκεται στο ίδιο VM.
9  *
10 * Η κλάση έχει μόνο getters, setters και έναν
11 * constructor. Τα χαρακτηριστικά της είναι
12 * int maxCapacity: μέγιστη χωρητικότητα
13 * int number: υπολοιπόμενο μέγεθος
14 * String type: τύπος τοποθεσίας
15 * double price: κόστος θέσεων
16 *
17 */
18 public class Zone implements Serializable{
19
20     private int maxCapacity;
21     private int number;
22     private String type;
23     private double price;
24
25     // Constructor
26     public Zone(int number, String type, double price) {
27         this.maxCapacity = number;
28         this.number = number;
29         this.type = type;
30         this.price = price;
31     }
```

```
32
33 // Getters
34 public int getMaxCapacity() {
35     return maxCapacity;
36 }
37 public int getNumber() {
38     return number;
39 }
40 public String getType() {
41     return type;
42 }
43 public double getPrice() {
44     return price;
45 }
46
47 // Setters
48 public void setMaxCapacity(int maxCapacity) {
49     this.maxCapacity = maxCapacity;
50 }
51 public void setNumber(int number) {
52     this.number = number;
53 }
54 public void setType(String type) {
55     this.type = type;
56 }
57 public void setPrice(double price) {
58     this.price = price;
59 }
60
61 }
```

Κώδικας 3.1: Κώδικας Zone

4 Η κλάση Ticket

Η κλάση Ticket (κώδικας 4.1) επεκτείνει το Serializable, διότι κάποιες φορές επιστρέφεται στον απομακρυσμένο client. Η κλάση αυτή κρατάει μερικά δεδομένα για ένα εισιτήριο όπως

1. **number** ο αριθμός θέσεων που έχει γίνει η κράτηση
2. **type** ο τύπος ζώνης
3. **name** το όνομα που έγινε η κράτηση
4. **price** το συνολικό κόστος

```
1 import java.io.Serializable;
2
3 /**
4  *
5  * @author padelis
6  *
7  * Η κλάση Ticket υλοποιεί την Serializable
8  * γιατί επιστρέφει και στον client που είναι
9  * απομακρυσμένος και δεν βρήσκεται στο ίδιο VM.
10 *
11 * Η κλάση έχει μόνο getters, setters και έναν
12 * constructor. Τα χαρακτηριστικά της είναι
13 * int number: αριθμός θέσεων
14 * String type: τύπος θέσεων
15 * String name: όνομα κατόχου εισιτηρίου
16 * double price: τιμή που έκλεισε την προσφορά
17 *
18 */
19 public class Ticket implements Serializable{
20     private int number;
21     private String type;
22     private String name;
23     private double price;
24
25     // Constructor
26     public Ticket(int number, String type, String name) {
27         this.number = number;
28         this.type = type;
29         this.name = name;
30         this.price = 0;
31     }
32 }
```

```
33 // Getters
34 public int getNumber() {
35     return number;
36 }
37
38 public String getType() {
39     return type;
40 }
41
42 public String getName() {
43     return name;
44 }
45
46 public double getPrice() {
47     return price;
48 }
49
50 // Setters
51 public void setNumber(int number) {
52     this.number = number;
53 }
54
55 public void setType(String type) {
56     this.type = type;
57 }
58
59 public void setName(String name) {
60     this.name = name;
61 }
62
63 public void setPrice(double price) {
64     this.price = price;
65 }
66 }
```

Κώδικας 4.1: Κώδικας Ticket

5 Η κλάση Notify

Η κλάση Notify (κώδικας 5.1) σε αντίθεση με την κλάση Zone και την Ticket, δεν επεκτείνει την Serializable διότι παραμένει το αντικείμενο μόνο στην τοπική VM του server. Η κλάση κρατάει πληροφορίες για τους χρήστες που θέλουν να ενημερωθούν αν υπάρχουν οι διαθέσιμες θέσεις που αναζήτησαν. Τα χαρακτηριστικά που έχει είναι

1. **cli** που είναι το interface του client για να καλέσει την μέθοδο ενημέρωσης
2. **type** είναι ο τύπος ζώνης που ενδιαφέρετε ο χρήστης
3. **number** ο αριθμός θέσεων που ενδιαφέρετε ο χρήστης

```
1  /**
2   * @author padelis
3   *
4   * Η κλάση έχει μόνο getters, setters και έναν
5   * constructor. Τα χαρακτηριστικά της είναι
6   * ClientInterface cli: όπου κρατάει τον απομακρισμένο client,
7   * έτσι ώστε να μπορεί να γίνει κλήση στις μεθόδους του
8   * String type: ο τύπος θέσεων που ενδιαφέρεται για
9   * να γίνει η ενημέρωση
10  * int number: ο αριθμός θέσεων που ενδιαφέρεται να ενημερωθεί
11  *
12  */
13  public class Notify {
14
15      private ClientInterface cli;
16      private String type;
17      private int number;
18
19      // Constructor
20      public Notify(ClientInterface cli, String type, int number) {
21          this.cli = cli;
22          this.type = type;
23          this.number = number;
24      }
25
26      // Getters
27      public ClientInterface getCli() {
28          return cli;
29      }
30      public String getType() {
31          return type;
32      }
33  }
```

```
33 public int getNumber() {  
34     return number;  
35 }  
36  
37 // Setters  
38 public void setCli(ClientInterface cli) {  
39     this.cli = cli;  
40 }  
41 public void setType(String type) {  
42     this.type = type;  
43 }  
44 public void setNumber(int number) {  
45     this.number = number;  
46 }  
47  
48 }
```

Κώδικας 5.1: Κώδικας Notify

6 Η κλάση TheaterSeats

Η κλάση TheaterSeats (κώδικας 6.1) διαχειρίζεται τις ζώνες, τις κρατήσεις και τις λίστες αναμονής. Ποίο συγκεκριμένα έχει 3 χαρακτηριστικά όπου

- **zonesList** είναι μια δυναμική λίστα όπου έχει όλες τις ζώνες
- **ticketList** είναι μια δυναμική λίστα όπου έχει όλες τις κρατήσεις
- **notifyList** έχει ως key το όνομα της ζώνης και ως value την λίστα με τους χρήστες που είναι σε αναμονή

```
1 import java.rmi.RemoteException;
2 import java.util.ArrayList;
3 import java.util.HashMap;
4
5 /**
6  *
7  * @author padelis
8  *
9  * Η κλάση TheaterSeats έχει τα χαρακτηριστικά
10 * ArrayList<Zone> zonesList: όπου είναι η λίστα με τα zones
11 * που υπάρχουν
12 *
13 * ArrayList<Ticket> ticketList: όπου είναι η λίστα με τα
14 * εισιτήρια που έχει γίνει κράτηση
15 *
16 * HashMap<String,ArrayList<Notify>> notifyList: όπου
17 * key είναι το όνομα της ζώνης και value είναι
18 * η λίστα των ατόμων που περιμένουν ενημέρωση εάν
19 * ελευθεροθούν θέσεις
20 *
21 */
22 public class TheaterSeats {
23
24     private ArrayList<Zone> zonesList;
25     private ArrayList<Ticket> ticketList;
26     private HashMap<String,ArrayList<Notify>> notifyList;
27
28     // Getters
29     public ArrayList<Zone> getZonesList() {
30         return zonesList;
31     }
32
33     public ArrayList<Ticket> getTicketList() {
```

```
34     return ticketList;
35 }
36
37 public HashMap<String, ArrayList<Notify>> getNotifyList() {
38     return notifyList;
39 }
40
41 // Setters
42 public void setZonesList(ArrayList<Zone> zonesList) {
43     this.zonesList = zonesList;
44 }
45
46 public void setTicketList(ArrayList<Ticket> ticketList) {
47     this.ticketList = ticketList;
48 }
49
50 public void setNotifyList(HashMap<String, ArrayList<Notify>> notifyList) {
51     this.notifyList = notifyList;
52 }
53
54 // Constructor
55 public TheaterSeats() {
56     this.zonesList = new ArrayList<Zone>();
57     this.ticketList = new ArrayList<Ticket>();
58     this.notifyList = new HashMap<String, ArrayList<Notify>>();
59 }
60
61 // Methods
62 /**
63  * Η μέθοδος newZone προσθέτει μια νέα ζώνη στην λίστα βάση
64  * των παραμέτρων. Επίσης, κατασκευάζει μια λίστα
65  * αναμονής για την ζώνη αυτή
66  *
67  * @param number
68  * @param type
69  * @param price
70  */
71 public void newZone(int number, String type, double price) {
72     zonesList.add(new Zone( number, type, price));
73     notifyList.put(type, new ArrayList<Notify>());
74 }
75
76 /**
77  *
```

```
78 * Η μέθοδος newTickets κάνει κράτηση αναλόγως
79 * της παραμετρου της και επιστρέφει
80 * zero: αν δεν υπάρχει καμία θέσει
81 * price: αν έγινε η κράτηση και επιστρέφει το κόστος
82 * που διαχωρίζεται με ';'
83 * avail: αν υπάρχει μικρότερος αριθμός διαθέσιμων θέσεων
84 * από όσες ζητήθηκαν αρχικά
85 * unknown: αν ο τύπος θέσεων είναι άγνωστη
86 *
87 *
88 * @param ticket
89 * @return "price;<cost>"
90 * @return "zero"
91 * @return "avail;<available seats>"
92 * @return "unknown"
93 */
94 public String newTickets(Ticket ticket) {
95
96     for(Zone tmp : zonesList) {
97
98         if( tmp.getType().equals( ticket.getType() )) {
99             if (tmp.getNumber() >= ticket.getNumber() ) {
100
101                 tmp.setNumber(tmp.getNumber()-ticket.getNumber());
102                 ticket.setPrice(tmp.getPrice() * ticket.getNumber());
103                 ticketList.add(ticket);
104                 return "price;"+ticket.getPrice();
105             }
106             else {
107                 if(tmp.getNumber() == 0)
108                     return "zero";
109                 else
110                     return "avail;"+tmp.getNumber();
111             }
112         }
113     }
114     return "unknown";
115 }
116
117 /**
118 * Ακυρώνει κάποια κράτηση εαν υπάρχει.
119 *
120 * @param type
121 * @param number
```

```
122 * @param name
123 * @return false αν δεν υπάρχει τέτοιος type αλλιώς true
124 */
125 public boolean cancelBoock(String type, int number, String name) {
126
127     String bType;
128     int bNumber;
129     String bName;
130     boolean ret = false;
131     int index = 0;
132     int newNumberZone = 0;
133
134     for(Ticket tmp: ticketList) {
135
136         bName = tmp.getName();
137         bNumber = tmp.getNumber();
138         bType = tmp.getType();
139
140         if(bName.equals(name) && bNumber == number && bType.equals(type)) {
141             ret = true;
142             break;
143         }
144         index++;
145     }
146
147     if(ret) {
148
149         ticketList.remove(index);
150
151         for(Zone tmpZ : zonesList) {
152             if(tmpZ.getType().equals(type)) {
153                 newNumberZone = tmpZ.getNumber()+number;
154                 tmpZ.setNumber(newNumberZone);
155                 break;
156             }
157         }
158
159         ArrayList<Notify> newQueue = new ArrayList<Notify>();
160         ArrayList<Notify> tmpArr = notifyList.get(type);
161
162         for(Notify tmpN : tmpArr ) {
163             if (tmpN.getNumber() <= newNumberZone) {
164                 try {
165                     tmpN.getCli().notifyCustomer(newNumberZone);
```

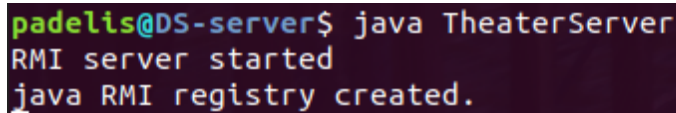


```
166     } catch (RemoteException e) {
167         System.out.println("Error: " + e.toString());
168     }
169 }
170 else {
171     newQueue.add(tmpN);
172 }
173 }
174
175 notifyList.remove(type);
176 notifyList.put(type, newQueue);
177 }
178
179 return ret;
180 }
181
182 /**
183  * Προσθέτει στην λίστα αναμονής τον πελάτη
184  * έτσι ώστε να τον ενημερώσει αν ελευθεροθούν
185  * οι θέσεις που ζήτησε αρχικά
186  *
187  * @param cli
188  * @param type
189  * @param number
190  */
191 public void addQueue(ClientInterface cli, String type, int number) {
192
193     Notify n = new Notify(cli,type,number);
194     notifyList.get(type).add(n);
195 }
196
197 }
```

Κώδικας 6.1: Κώδικας TheaterSeats

7 Η κλάση TheaterServer

Η κλάση TheaterServer (κώδικας 7.1) επεκτείνει την ServerInterface έτσι ώστε να μπορεί κάποιος γνωρίζοντας το interface να καλέσει μεθόδους της κλάσης και κληρονομεί την κλάση UnicastRemoteObject έτσι ώστε να διαχειρίζεται την remote λειτουργία. Η κλάση τρέχει χωρίς ορίσματα (εικόνα 7.1).



```
padelis@DS-server$ java TheaterServer
RMI server started
java RMI registry created.
```

Σχήμα 7.1: Το terminal του server

```
1 import java.rmi.server.UnicastRemoteObject;
2 import java.util.ArrayList;
3 import java.rmi.Naming;
4 import java.rmi.RemoteException;
5 import java.rmi.registry.LocateRegistry;
6
7 /**
8  * @author padelis
9  *
10 * Η κλάση TheaterServer επεκτείνει την UnicastRemoteObject
11 * για να μπορεί να υλοποιεί καθήκοντα απομακρισμένου server
12 * και υλοποιεί το interface ServerInterface. Επίσης, έχει
13 * ένα χαρακτηριστικό το TheaterSeats tSeats, έτσι ώστε
14 * να διαχειρίζεται το θέατρο.
15 *
16 */
17 public class TheaterServer extends UnicastRemoteObject implements ServerInterface
18 {
19
20     private TheaterSeats tSeats;
21
22     /**
23      * Καλή τον constructor της UnicastRemoteObject και
24      * δημιουργεί τις ζώνες που υπάρχουν
25      *
26      * @throws RemoteException
27      */
28     public TheaterServer() throws RemoteException
29     {
30         super(0);
```

```
31 this.tSeats = new TheaterSeats();
32 this.tSeats.newZone(200, "pa", 50);
33 this.tSeats.newZone(300, "pb", 40);
34 this.tSeats.newZone(500, "pg", 30);
35 this.tSeats.newZone(100, "ke", 25);
36 this.tSeats.newZone(50, "pth", 20);
37 }
38
39 /**
40  * Η main δημιουργεί ένα instance του εαυτού της
41  * δημιουργεί εάν registry για το port 50444 και
42  * δίνει "" για το url το instance που δημιούργησε
43  */
44 public static void main(String[] args) {
45     try {
46
47         System.out.println("RMI server started");
48         TheaterServer srv = new TheaterServer();
49         LocateRegistry.createRegistry(50444);
50         System.out.println("java RMI registry created.");
51
52         String url = "rmi://127.0.0.1:50444/TheaterServer";
53
54         Naming.rebind(url, srv);
55
56     } catch (Exception e) {
57         System.out.println(e.toString());
58     }
59 }
60
61
62 /**
63  * Καλή την μέθοδο getZonesList και επιστρέφει ένα ArrayList<Zone>
64  */
65 @Override
66 public synchronized ArrayList<Zone> list() throws RemoteException {
67     return tSeats.getZonesList();
68 }
69
70 /**
71  * Δημιουργεί ένα ticket βάση των παραμέτρων που έλαβε και
72  * επιστρέφει το αποτέλεσμα της μέθοδου newTickets
73  */
74 @Override
```

```
75 public synchronized String book(String type, int number, String name) throws
    RemoteException {
76
77     Ticket t = new Ticket(number, type, name);
78
79     return tSeats.newTickets(t);
80 }
81
82 /**
83  * Επιστρέφει το αποτέλεσμα της μεθόδου getTicketList
84  */
85 @Override
86 public synchronized ArrayList<Ticket> guests() throws RemoteException {
87     return tSeats.getTicketList();
88 }
89
90 /**
91  * Επιστρέφει το αποτέλεσμα της μεθόδου cancelBoock
92  */
93 @Override
94 public synchronized boolean cancel(String type, int number, String name) throws
    RemoteException {
95
96     return tSeats.cancelBoock(type, number, name);
97 }
98
99 @Override
100 public void addQueue(ClientInterface cli, String type, int number) throws RemoteException{
101     tSeats.addQueue(cli, type, number);
102 }
103
104 }
```

Κώδικας 7.1: Κώδικας TheaterServer

8 Η κλάση TheaterClient

Η κλάση TheaterClient (κώδικας 8.1) επεκτείνει την ClientInterface έτσι ώστε να μπορεί ο server γνωρίζοντας το interface να καλέσει την μέθοδο ενημέρωσης της κλάσης και κληρονομεί την κλάση UnicastRemoteObject έτσι ώστε να διαχειρίζεται την remote λειτουργία εάν ζητήσει ενημέρωση. Μερικά ενδεικτικά τρεξίματα είναι στις εικόνες ?? , 8.2 και 8.3.

```
padelis@DS-client$ java TheaterClient
Usage:
list <hostname>
    Displays the available seats in each zone
book <hostname> <type> <number> <name>
    Book <number> tickets in zone <type> by the <name>
guests <hostname>
    Displays a list of all viewers
cancel <hostname> <type> <number> <name>
    Cancels a book
padelis@DS-client$ java TheaterClient list 127.0.0.1
+-----+-----+-----+
|Number|Type|Price|
+-----+-----+-----+
|200/200|pa|50.0|
|300/300|pb|40.0|
|500/500|pg|30.0|
|100/100|ke|25.0|
|50/50|pth|20.0|
+-----+-----+-----+
padelis@DS-client$ java TheaterClient guests 127.0.0.1
+-----+-----+-----+
|Name|Number|Type|Price|
+-----+-----+-----+
padelis@DS-client$ java TheaterClient book 127.0.0.1 pa 5 "padelis protos"
The book is confirmed for
Name: padelis protos
Type: pa
Number: 5
Price: 250.0
padelis@DS-client$ java TheaterClient book 127.0.0.1 pa 194 "uniwa ice"
The book is confirmed for
Name: uniwa ice
Type: pa
Number: 194
Price: 9700.0
padelis@DS-client$ java TheaterClient guests 127.0.0.1
+-----+-----+-----+
|Name|Number|Type|Price|
+-----+-----+-----+
|padelis protos|5|pa|250.0|
|uniwa ice|194|pa|9700.0|
+-----+-----+-----+
padelis@DS-client$ java TheaterClient list 127.0.0.1
+-----+-----+-----+
|Number|Type|Price|
+-----+-----+-----+
|1/200|pa|50.0|
|300/300|pb|40.0|
|500/500|pg|30.0|
|100/100|ke|25.0|
|50/50|pth|20.0|
+-----+-----+-----+
```

Σχήμα 8.1: Ενδεικτικό τρέξιμο του client A

```
padellis@DS-client$ java TheaterClient book 127.0.0.1 pa 6 "padelis protos"
There are 1 seats only
Do you want to make a book if it's possible[yes/else]: yes
The book is confirmed for
Name: padelis protos
Type: pa
Number: 1
Price: 50.0
padellis@DS-client$ java TheaterClient book 127.0.0.1 pa 6 "padelis protos"
There are no empty seats
Do you want to get notified, if there is 6 of seats available[yes/else]: yes
There are 6 available seats
[]

padellis@padelis-linux: ~/Documents/distributed_systems/project2/project2/bin
padellis@DS-client2$ java TheaterClient guests 127.0.0.1
+-----+
|Name      |Number |Type |Price |
+-----+
|padelis protos |5      |pa   |250.0 |
|uniwa ice  |194    |pa   |9700.0|
|padelis protos |1      |pa   |50.0  |
+-----+
padellis@DS-client2$ java TheaterClient cancel 127.0.0.1 pa 5 "padelis protos"
cancel
+-----+
|Number    |Type |Price |
+-----+
|5/200     |pa   |50.0  |
|300/300   |pb   |40.0  |
|500/500   |pg   |30.0  |
|100/100   |ke   |25.0  |
|50/50     |pth  |20.0  |
+-----+
padellis@DS-client2$ java TheaterClient cancel 127.0.0.1 pa 1 "padelis protos"
cancel
+-----+
|Number    |Type |Price |
+-----+
|6/200     |pa   |50.0  |
|300/300   |pb   |40.0  |
|500/500   |pg   |30.0  |
|100/100   |ke   |25.0  |
|50/50     |pth  |20.0  |
+-----+
padellis@DS-client2$ java TheaterClient guests 127.0.0.1
+-----+
|Name      |Number |Type |Price |
+-----+
|uniwa ice  |194    |pa   |9700.0|
+-----+
```

Σχήμα 8.2: Ενδεικτικό τρέξιμο του client B

```
padellis@DS-client$ java TheaterClient book 127.0.0.1 pa 3 "padelis protos"
The book is confirmed for
Name: padelis protos
Type: pa
Number: 3
Price: 150.0
padellis@DS-client$ java TheaterClient book 127.0.0.1 pa 4 "padelis protos"
There are 3 seats only
Do you want to make a book if it's possible[yes/else]: no
Do you want to get notified, if there is 4 of seats available[yes/else]: yes
There are 6 available seats
[]

padellis@padelis-linux: ~/Documents/distributed_systems/project2/project2/bin
padellis@DS-client2$ java TheaterClient cancel 127.0.0.1 pa 3 "padelis protos"
cancel
+-----+
|Number    |Type |Price |
+-----+
|6/200     |pa   |50.0  |
|300/300   |pb   |40.0  |
|500/500   |pg   |30.0  |
|100/100   |ke   |25.0  |
|50/50     |pth  |20.0  |
+-----+
```

Σχήμα 8.3: Ενδεικτικό τρέξιμο του client Γ

```
1 import java.rmi.Naming;
2 import java.rmi.RemoteException;
3 import java.rmi.server.UnicastRemoteObject;
```

```
4 import java.util.ArrayList;
5 import java.util.Scanner;
6
7 /**
8  * @author padelis
9  *
10 * Η κλάση TheaterClient επεκτείνει την UnicastRemoteObject διότι
11 * περιμένει μήνυμα ενημέρωσης σε κάποιες περιπτώσεις
12 *
13 */
14 public class TheaterClient extends UnicastRemoteObject implements ClientInterface
15 {
16
17     /**
18     * Καλή τον constructor της UnicastRemoteObject
19     *
20     * @throws RemoteException
21     */
22     public TheaterClient() throws RemoteException {
23         super();
24     }
25
26     /**
27     * @param
28     * @param list <hostname>
29     * @param book <hostname> <type> <number> <name>
30     * @param guests <hostname>
31     * @param cancel <hostname> <type> <number> <name>
32     */
33     public static void main(String[] args) {
34
35         /*
36         * Ελέγχει τον αριθμό των ορισμάτων και αν είναι λιγότερα
37         * από 2 τότε εμφανίζει την χρήση του client και κλήνει
38         */
39         checkArgs(args.length,2);
40         try {
41             // Κατασκευάζει ένα instance TheaterClient
42             TheaterClient tc = new TheaterClient();
43             // Καλή την μέθοδο clientServices
44             tc.clientServices(args,tc);
45         } catch (RemoteException e) {
46             System.out.println("Error: RemoteException " + e.getMessage());
47         }
```

```
48 }
49
50 /**
51  *
52  * @param args τα arguments της main
53  * @param tc η ίδια η κλάση TheaterClient
54  */
55 public void clientServices(String[] args, TheaterClient tc) {
56
57     Scanner sc = null;
58     String line = null;
59     int ret;
60
61     try {
62         // Το url του remote server
63         String url = "rmi://"
64             + args[1]
65             + ":5044/TheaterServer";
66
67         // Αναζήτηση του remote server αντικειμένου βάση του url
68         ServerInterface srv = (ServerInterface) Naming.lookup(url);
69
70         int number;
71         String type;
72         String name;
73         boolean flag = false;
74
75         switch(args[0]) {
76
77             case "list":
78
79                 // καλή την μέθοδο printList
80                 this.printList(srv);
81                 System.exit(0);
82                 break; // δεν χρειάζεται
83
84             case "book":
85
86                 // ελέγχει αν υπάρχουν 5 ή παραπάνω args
87                 checkArgs(args.length, 5);
88
89                 try {
90
91                     type = args[2];
```



```
92     number = Integer.parseInt(args[3]);
93     name = args[4];
94
95     // Σε περίπτωση που ο αριθμός δεν είναι πάνω από 0
96     if(number < 1) {
97         System.out.println("Number must be 1 or more");
98         System.exit(0);
99     }
100
101     sc = new Scanner(System.in);
102
103     ret = number;
104
105     do {
106         // κράτηση
107         ret = makeABook(srv, type, ret, name);
108
109         if ( ret == 1) {
110
111             flag = true;
112             break;
113         }
114         else if ( ret < 0) {
115             flag = true;
116             System.out.print("Do you want to make a book if it's possible[yes/else]: ");
117             line = sc.nextLine();
118             if(line.equals("yes")) {
119                 ret = -1 * ret;
120             }
121             else {
122                 break;
123             }
124         }
125         else if(ret == 2) {
126             flag = false;
127         }
128     }while(ret != 2);
129
130     // Ερώτηση για ενημέρωση των θέσεων που ζήτησε
131     if(flag) {
132
133         System.out.print("Do you want to get notified, if there is "
134             + number+" of seats available[yes/else]: ");
135         line = sc.nextLine();
```

```
136     flag = false;
137     if(line.equals("yes")) {
138         //κλήση της μεθόδου του remote server
139         // για προσθήκη στην λίστα αναμονής
140         srv.addQueue(tc, type, number);
141         flag = true;
142     }
143 }
144
145 if(sc != null)
146     sc.close();
147
148 if(flag)
149     break;
150
151
152 } catch(NumberFormatException e) {
153     System.out.println("Error: arg4 is not a number " + e.getMessage());
154 }
155
156 System.exit(0);
157 break; // δεν χρειάζεται
158
159 case "guests":
160
161     // Η μέθοδος εμφανίζει τους χρήστες που έχουν κάνει κράτηση
162     printGuests(srv);
163     System.exit(0);
164     break;
165
166 case "cancel":
167
168     System.out.println("cancel");
169
170     // Έλεγχος των arguments
171     checkArgs(args.length,5);
172
173     try {
174
175         type = args[2];
176         number = Integer.parseInt(args[3]);
177         name = args[4];
178
179         // Αν ο αριθμός δεν είναι πάνω από 0
```

```
180     if(number < 1) {
181         System.out.println("Number must be 1 or more");
182         System.exit(0);
183     }
184
185     // κλήση μεθόδου για ακύρωση κράτησης
186     cancelABook(srv,type,number,name);
187
188     }catch(NumberFormatException ex) {
189         System.out.println("Error: arg4 is not a number " + ex.getMessage());
190     }
191     System.exit(0);
192     break; // δεν χρειάζεται
193
194     default:
195         usage();
196         System.exit(0);
197     }
198 } catch (Exception e) {
199     System.out.println(e.toString());
200 }
201 }
202
203 /**
204  * Εμφανίζει την λίστα των ζωνών και την διαθεσιμότητα τους
205  *
206  * @param srv, o remote server
207  * @throws RemoteException
208  */
209 public void printList(ServerInterface srv) throws RemoteException{
210
211     int maxCapacity;
212     int number;
213     double price;
214     String type;
215
216     ArrayList<Zone> arr = srv.list();
217
218     System.out.println("-----+-----+-----+");
219     System.out.printf("|%-20s|%-6s|%-10s|\n", "Number", "Type", "Price");
220     System.out.println("-----+-----+-----+");
221     for(Zone tmp : arr) {
222         number = tmp.getNumber();
223         maxCapacity = tmp.getMaxCapacity();
```

```
224     price = tmp.getPrice();
225     type = tmp.getType();
226
227     System.out.printf("%-20s", number + "/" + maxCapacity);
228     System.out.printf("%-6s", type);
229     System.out.printf("%-10s\n", price);
230
231 }
232 System.out.println("+-----+-----+");
233 }
234
235 /**
236  * Προσπαθεί να κάνει κράτηση
237  *
238  * @param srv, ο remote server
239  * @param type, ο τύπος ζώνης
240  * @param number, ο αριθμός θέσεων
241  * @param name, το όνομα κράτησης
242  * @return 0, unknown type
243  * @return 1, μη διαθέσιμες θέσεις
244  * @return 2, η κράτηση έγινε με επιτυχία
245  * @return < 0, οι διαθέσιμες θέσεις
246  * @throws RemoteException
247  */
248 public int makeABook(ServerInterface srv, String type, int number, String name)
249     throws RemoteException{
250     String ret;
251     String[] inputs;
252     ret = srv.book(type, number, name);
253     int returnVal = 0;
254
255     inputs = ret.split(";");
256
257     switch(inputs[0]) {
258     case "unknown":
259         System.out.println("Type \"'"+type+"\" is not valid");
260         break;
261
262     case "zero":
263         System.out.println("There are no empty seats");
264         returnVal = 1;
265         break;
266
267     case "avail":
```

```
268     System.out.println("There are "+inputs[1]
269         +" seats only");
270
271     returnVal = -1 * Integer.parseInt(inputs[1]);
272     break;
273     case "price":
274         System.out.println("The book is confirmed for");
275         System.out.println("Name: " + name);
276         System.out.println("Type: " + type);
277         System.out.println("Number: " + number);
278         System.out.println("Price: " + inputs[1]);
279         returnVal = 2;
280         break;
281     }
282
283     return returnVal;
284 }
285
286 /**
287  * Εμφανίζει αυτούς που έχουν κάνει κράτηση
288  *
289  * @param srv, ο remote server
290  * @throws RemoteException
291  */
292 public void printGuests(ServerInterface srv) throws RemoteException{
293
294     int number;
295     double price;
296     String type;
297     String name;
298
299     ArrayList<Ticket> arrT = srv.guests();
300
301     System.out.println("-----+-----+-----+-----");
302     System.out.printf("|%-20s|%-10s|%-6s|%-10s|\n", "Name", "Number", "Type", "Price");
303     System.out.println("-----+-----+-----+-----");
304
305     for(Ticket tmpT : arrT) {
306         number = tmpT.getNumber();
307         name = tmpT.getName();
308         type = tmpT.getType();
309         price = tmpT.getPrice();
310
311         System.out.printf("|%-20s", name);
```

```
312     System.out.printf("%-10s", number);
313     System.out.printf("%-6s", type);
314     System.out.printf("%-10s\n", price);
315 }
316 System.out.println(" +-----+-----+-----+");
317 }
318
319 /**
320  * Προσπαθεί να ακυρώσει την κράτηση
321  *
322  * @param srv, ο remote server
323  * @param type, ο τύπος ζώνης
324  * @param number, ο αριθμός θέσεων
325  * @param name, το όνομα κράτησης
326  * @throws RemoteException
327  */
328 public void cancelABook(ServerInterface srv, String type, int number, String name)
329     throws RemoteException{
330
331     if (srv.cancel(type, number, name))
332         printList(srv);
333     else
334         System.out.println("There are no books with this info");
335 }
336
337 /**
338  * Εμφανίζει την χρήση
339  */
340 public static void usage() {
341     System.out.println("Usage:\n");
342     System.out.println("list <hostname>");
343     System.out.println("\tDisplays the available seats in each zone");
344     System.out.println("book <hostname> <type> <number> <name>");
345     System.out.println("\tBook <number> tickets in zone <type> by the <name>");
346     System.out.println("guests <hostname>");
347     System.out.println("\tDisplays a list of all viewers");
348     System.out.println("cancel <hostname> <type> <number> <name>");
349     System.out.println("\tCancels a book");
350 }
351
352 /**
353  * Ελέγχει το μέγεθος των arguments και αν είναι λάθος
354  * τότε εμφανίζει την χρήση του client και τερματίζει
355  */
```

```
356 * @param argsNum, το μέγεθος των arguments
357 * @param num, το ελάχιστο μέγεθος που πρέπει να είναι
358 */
359 public static void checkArgs(int argsNum, int num) {
360     if(argsNum < num) {
361         usage();
362         System.exit(0);
363     }
364 }
365
366 /**
367  * Εμφανίζει την ενημέρωση για τις διαθέσιμες θέσεις
368  * όταν το καλέσει ο server
369  */
370 @Override
371 public void notifyCustomer(int number) throws RemoteException {
372     System.out.println("There are " + number + " available seats");
373 }
374
375 }
```

Κώδικας 8.1: Κώδικας TheaterClient