ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
Distributed Systems

ΕΡΓΑΣΙΑ 2

Όνομα Μαθητή : Άγγελος Κωνσταντίνος

Επίθετο Μαθητή : Μεντζέλος

Αριθμός Μητρώου : 21390132

Ημερομηνία ολοκλήρωσης :

16/06/2024

Περιεχόμενα

1. Σχολιασμός Κώδικα

1.1 BookRoomsServer.java

Στον server BookRoomsServer.java αρχικά τοποθετώ τις βιβλιοθήκες τις οποίες χρησιμοποιώ ώστε να γίνει η RMI σύνδεση με τον client. Στην κλάση BookRoomsServer έχω δύο μεθόδους την public static void main η οποία είναι η main συνάρτηση της κλάσης και την public BookRoomsServer η οποία είναι ο constructor της κλάσης. Αρχικά στην συνάρτηση public static void main εκτυπώνω ένα μήνυμα και αμέσως μετά με ένα try catch exception δηλώνω το port του RMI server μου και εκτυπώνω άλλο ένα μήνυμα. Αν δεν συμβεί αυτό καταλήγει σε ένα RemoteException που θα εκτυπώσει ότι υπάρχει ήδη ο RMI server. Μετά το try catch καλώ τον constructor της κλάσης. Εκεί πάλι μέσα σε ένα try catch καλώ τον constructor της κλάσης BookRoomsImpl και κάνω bind στην τοπική μου ip (localhost), το port που δήλωσα και το αντικείμενο BookRoomsImpl. Αν δεν γίνει το bind πετάει ένα exception και τερματίζει το πρόγραμμα.

1.2 BookRooms.java

Στο αρχείο BookRooms.java υπάρχει το interface της κλάσης BookRoomsImpl. Αρχικά υπάρχουν οι βιβλιοθήκες που χρησιμοποιώ για τις μεθόδους του interface. Αρχικά η πρώτη μέθοδος είναι η public ArrayList<Room> list() throws RemoteException. Αυτή η μέθοδος υπάρχει ώστε να εκτυπώνει τα διαθέσιμα δωμάτια που υπάρχουν για κάθε τύπου δωματίου. Τα επιστρέφει όλα σε μία λίστα τύπου ArrayList<Room> όπου το Room είναι μια κλάση που θα αναλύσω παρακάτω. Αν υπάρξει κάποιο πρόβλημα στην σύνδεση πετάει ένα RemoteException. Η δεύτερη μέθοδος είναι η public double book(String type , int num, String name) throws RemoteException. Αυτή η μέθοδος υπάρχει ώστε ο πελάτης να κλίνει τον αριθμό των δωματίων αλλά και τον τύπο του με βάση το όνομα του. Για αυτόν τον λόγο υπάρχουν και τα τρία ορίσματα. Επιστρέφει έναν double αριθμό και αν υπάρχει κάποιο πρόβλημα στην συνάρτηση όπως και στην προηγούμενη πετάει RemoteException. Πηγαίνοντας τώρα στην τρίτη μέθοδο η οποία είναι η public ArrayList<Customer> guests() throws RemoteException επιστρέφει όπως και στην πρώτη μια λίστα τύπου ArrayList<Customer> όπου Customer είναι μία κλάση η οποία θα την αναλύσω παρακάτω. Η μέθοδος αυτή χρησιμοποιείτε με σκοπό να μπορούμε να δούμε ποιοι χρήστες έχουν κλείσει δωμάτια, τι τύπου είναι και τον αριθμό των δωματίων. Η τέταρτη συνάρτηση public ArrayList<Room> cancel (String type , int num , String name) throws Remote Exception χρησιμοποιείτε με σκοπό ο πελάτης να ακυρώσει μια κράτηση δίνοντας τον τύπο, τον αριθμό αλλά και το όνομα του πελάτη. Αυτά είναι και τα ορίσματα που δέχεται η συνάρτηση ενώ επιστρέφει και αυτή μια λίστα τύπου ArrayList<Room>. Επίσης και αυτή πετάει RemoteException. Τέλος η μέθοδος public void registerForCallBack (BookRoomsClient callbackObj, String type) throws RemoteException χρησιμοποιείτε με σκοπό να ενημερώνει τον πελάτη σε περίπτωση που έχει επιλέξει να περιμένει να βρεθούν τα δωμάτια που επιθυμεί. Παίρνει σαν είσοδο ένα αντικείμενο τύπου BookRoomsClient και τον τύπο του δωματίου.

1.3 Customer.java

Η κλάση Customer χρησιμοποιείτε με σκοπό να αποτυπώνω κάθε πελάτη με τα εξής χαρακτηριστικά: το όνομα του (String name), τον τύπου δωμάτιο που έχει επιλέξει (String roomType), τον αριθμό των δωματίων που έχει κλείσει (int numberOfRooms) και το συνολικό κόστος των δωματίων που έχει κλείσει (double totalCost). Η κλάση περιέχει έναν constructor που αρχικοποιεί αυτά τα χαρακτηριστικά και μεθόδους τύπου getters setters για το κάθε χαρακτηριστικό. Τέλος η κλάση έχει implements το Serializable επειδή τα αντικείμενα μπαίνουν σε ArrayList.

1.4 Room.java

Αντίστοιχα λοιπόν και η κλάση Room η οποία αποτυπώνει το κάθε δωμάτιο με τα εξής χαρακτηριστικά: το τύπο του δωματίου (String type), το κόστος του δωματίου (int cost) και αν είναι διαθέσιμο (int availability). Η κλάση περιέχει έναν constructor που αρχικοποιεί αυτά τα χαρακτηριστικά και μεθόδους τύπου getters setters για το κάθε χαρακτηριστικό. Τέλος η κλάση έχει implements το Serializable επειδή τα αντικείμενα μπαίνουν σε ArrayList.

1.5 BookRoomsClientInter.java

Αυτό το αρχείο είναι για το interface του client ώστε να ενημερώνει τον χρήστη για την διαθεσιμότητα των δωματίων. Αρχικά χρησιμοποιώ την βιβλιοθήκη που θα χρειαστώ για το exception της μεθόδου και για το extends του interface. Η κλάση περιέχει μια μέθοδο η οποία είναι void notifyA (String type) throws RemoteException. Η μέθοδος αυτή υλοποιεί αυτό που εξήγησα στην αρχή. Παίρνει ως όρισμα τον τύπο του δωματίου ώστε να ενημερώσει για το συγκεκριμένο τύπο δωματίου.

1.6 BookRoomsImpls.java

Σε αυτή την κλάση περιέχει την υλοποίηση των μεθόδων που περιέχει το interface BookRooms. Αρχικά τοποθετώ τις βιβλιοθήκες τις οποίες θα χρησιμοποιήσω. Η κλάση BookRoomsImpl έχει extends το UnicastRemoteObject και ως implements περιέχει το BookRooms ώστε να υλοποιήσω τις συναρτήσεις. Στην κλάση αυτή έχω κάποια χαρακτηριστικά τα οποία θα μου χρειαστούν για της μεθόδους τα οποία είναι τα εξής: ένα ArrayList<Room> rooms που θα είναι η λίστα με τα δωμάτια, ένα ArrayList<Customer> customers που θα είναι η λίστα με τους πελάτες και ένα Map<String,ArrayList<BookRoomsClientInter>> clients που θα είναι ένα map με τον τύπο του δωματίου και τους πελάτες που χρησιμεύει για την ενημέρωση. Πέρα από τις μεθόδους του Interface BookRooms έχει έναν constructor όπου δεσμεύει δυναμικά τις δύο λίστες και το map και βάζει στην λίστα των δωματίων το κάθε δωμάτιο που είναι τύπου Room δηλαδή αρχικοποιώ τον τύπο τους, το κόστος και την διαθεσιμότητα. Ο constructor πετάει RemoteException. Η μέθοδος ArrayList<Room> list() επιστρέφει την λίστα rooms στον client. Η μέθοδος book που ανάλυσα στο interface BookRooms ελέγχει επαναληπτικά για κάθε δωμάτιο που υπάρχει στην λίστα αν ο τύπος δωματίου υπάρχει στην λίστα και αν ισχύει ελέγχει ή η διαθεσιμότητα των δωματίων που υπάρχουν είναι μεγαλύτερη από τον αριθμό που δόθηκε ή αν είναι ίση με το 0 ή αν είναι μικρότερη. Αν ισχύει η πρώτη περίπτωση τότε στον συγκεκριμένο τύπο δωματίου μειώνει την διαθεσιμότητα, υπολογίζει το συνολικό κόστος, τοποθετεί στην λίστα τον πελάτη και επιστρέφει το κόστος. Αν ισχύει η δεύτερη περίπτωση τότε επιστρέφει 0 και αν ισχύει η τρίτη περίπτωση επιστρέφει την διαθεσιμότητα των δωματίων. Η μέθοδος guests επιστρέφει την λίστα με τους πελάτες. Αν δεν υπάρχει δωμάτιο του τύπου που έδωσε ο πελάτης επιστρέφει 0. Τώρα στην μέθοδο cancel που υλοποίησα ελέγχει επαναληπτικά για κάθε δωμάτιο που υπάρχει στην λίστα αν υπάρχει ο τύπος δωματίου που δόθηκε αυξάνει την διαθεσιμότητα των δωματίου του συγκεκριμένου τύπου και για κάθε πελάτη αν το όνομα που δόθηκε είναι στην λίστα τότε βάζει στην λίστα τον ίδιο πελάτη αφαιρώντας του τα δωμάτια που ακύρωσε και το κόστος των δωματίων που ακυρώθηκαν. Μετά βγάζει τον ίδιο πελάτη που δεν του έχουν αφαιρεθεί τα συγκεκριμένα στοιχεία από την λίστα και καλεί την συνάρτηση notf(type) που θα αναλύσω σε λίγο πως υλοποιείτε. Μόλις επιστρέψει από το κάλεσμα της συνάρτησης, σταματάει η επαναληπτική μέθοδος και επιστρέφει την λίστα των δωματίων. Όπως είπα και προηγουμένως έχω μία συνάρτηση η οποία είναι η public void notf(String type) throws RemoteException. Σε αυτή την συνάρτηση ελέγχω αν το χαρακτηριστικό clients που είναι ένα map όπως είπα και στην αρχή, περιέχει τον τύπο του δωματίου που αυτό σημαίνει ότι αν ισχύει αυτή η συνθήκη τότε υπάρχουν πελάτες που περιμένουν για δωμάτιο. Αν ισχύει η συνθήκη τότε σε μία λίστα τύπου ArrayList<BookRoomsClientInter> βάζω τους τύπους δωματίου του κάθε πελάτη. Μετά ενημερώνει κάθε πελάτη ότι υπάρχουν διαθέσιμα δωμάτια με την συνάρτηση του interface BookRoomsClientInter παίρνοντας ως παράμετρο τον τύπο του δωματίου. Μόλις τελειώσει η επανάληψη βγάζει από το map τον συγκεκριμένο τύπο δωματίου. Τέλος στην τελευταία μέθοδο του interface BookRooms η registerForCallBack δημιουργεί μια λίστα τύπου ArrayList<BookRoomsClientInter> και βάζει στην λίστα τους τύπους δωματίων των πελατών. Αμέσως μετά ελέγχει αν έχει δημιουργηθεί η λίστα. Αν δεν έχει γίνει αυτό δηλαδή είναι null τότε κάνει δυναμική δέσμευση της λίστας και βάζει στο map τον τύπο και την λίστα που δέσμευσα. Μετά τον έλεγχο τοποθετώ στην λίστα τον πελάτη που θα είναι σε αναμονή για δωμάτιο.

1.7 BookRoomsClient.java

Στην κλάση αυτή υλοποιείται όλη η μεριά του πελάτη. Αρχικά τοποθετώ τις βιβλιοθήκες τις οποίες χρειάζομαι για τα exceptions , την RMI σύνδεση και για άλλα πράγματα που χρειάστηκα στον κώδικα. Η κλάση έχει ένα χαρακτηριστικό private static Boolean flag=false και τρείς μεθόδους όπου η πρώτη είναι ο protected constructor που απλά καλεί την super, την public static void main που θα αναλύσω σε λίγο και την μέθοδο από το interface BookRoomsClientInter την void notifyA που χρησιμοποιώ για την ενημέρωση των πελάτων όταν υπάρχει διαθέσιμο δωμάτιο. Η μέθοδος notifyA εκτυπώνει ένα μήνυμα και κάνει την τιμή flag ίση με true(ότι βρέθηκε δωμάτιο, θα εξηγήσω σε λίγο γιατί). Η συνάρτηση main λοιπόν βρίσκεται όλη μέσα σε ένα try catch που πιάνει τα εξής exceptions : NotBoundException, RemoteException και MalformedURLException. Αρχικά κάνω την RMI σύνδεση με τον server παίρνοντας ως όρισμα την ip από την γραμμή εντολών και δημιουργώ ένα αντικείμενο τύπου BookRoomsClient. Αμέσως μετά παίρνω το δεύτερο όρισμα που δίνει ο χρήστης από την γραμμή εντολών και ελέγχω αν τα ορίσματα που έδωσε ο χρήστης είναι λιγότερα από δύο ώστε να σταματήσω την λειτουργία του κώδικα αν ισχύει αυτή η περίπτωση. Στην συνέχεια ελέγχω αν η επιλογή που έδωσε ο χρήστης είναι “list” τότε δημιουργώ μια λίστα τύπου ArrayList<Room> και την γεμίζω με το αποτέλεσμα την συνάρτησης list του BookRoomsImpl. Εκτυπώνω το κάθε δωμάτιο της λίστας και τερματίζω το πρόγραμμα. Αν η επιλογή που έδωσε ο χρήστης είναι “book” τότε κάνω έναν έλεγχο αν έχει δώσει ο χρήστης λιγότερα από πέντε ορίσματα και αν έχει γίνει αυτό τερματίζει το πρόγραμμα αλλιώς παίρνω τα ορίσματα από τον χρήστη. Μετά καλώ την συνάρτηση book του BookRoomsImpl με τα ορίσματα που δόθηκαν από τον χρήστη και κάνω έλεγχο για την επιστροφή του κόστους. Αν το κόστος είναι 0 τότε δεν υπάρχουν καθόλου δωμάτια του τύπου που ζητήθηκαν και έτσι ρωτάει τον χρήστη αν θέλει να περιμένει. Αν η επιλογή είναι ναι τότε καλεί την συνάρτηση registerForCallBack και ο χρήστης περιμένει αλλιώς ακυρώνεται η κράτηση και τερματίζεται το πρόγραμμα. Αν το κόστος είναι μικρότερο από τον αριθμό των δωματίων που δόθηκαν τότε εκτυπώνει το συνολικό υπόλοιπο των δωματίων αυτού του τύπου και ρωτάει τον χρήστη αν τα θέλει. Αν τα θέλει τότε τα κλείνει και τερματίζει το πρόγραμμα. Αλλιώς ρωτάει αν θέλει να περιμένει και γίνεται πάλι η ίδια διαδικασία με την περίπτωση του κόστους όταν είναι 0. Τέλος αν το κόστος είναι μεγαλύτερο από τον αριθμό των δωματίων που ζητήθηκαν τότε γίνεται η κράτηση και τερματίζει το πρόγραμμα. Αν τώρα η επιλογή του χρήστη είναι “guests” τότε καλεί την συνάρτηση guests του BookRoomsImpl και το αποτέλεσμα της το βάζει σε μια λίστα τύπου ArrayList<Customer>. Έτσι εκτυπώνει τον κάθε πελάτη και μετά τερματίζει το πρόγραμμα. Αν η επιλογή είναι “cancel” τότε ελέγχει όπως και στο book για τα πέντε ορίσματα, παίρνει τα ορίσματα αν ισχύει και καλεί την συνάρτηση cancel του BookRoomsImpl που επιστρέφει μια λίστα τύπου ArrayList<Room> όπου θα έχει τα καινούργια διαθέσιμα δωμάτια. Έτσι εκτυπώνω τα διαθέσιμα δωμάτια και τερματίζω το πρόγραμμα. Τέλος με επαναληπτικό τρόπο ελέγχω αν το flag είναι false που σημαίνει ότι ο πελάτης που περιμένει για διαθέσιμο δωμάτιο δεν έχει πάρει απάντηση ότι υπάρχει δωμάτιο. Τότε περιμένει μέχρι να γίνει cancel από κάποιον.

2. Ενδεικτικά τρεξίματα

2.1 Μεταγλώττιση

Για να μεταγλωττίσουμε τους κώδικες θα πρέπει να γίνει όπως φαίνεται παρακάτω:

```
javac *.java
```

2.2 Εκτέλεση

Τώρα για να τρέξουμε τον server πρέπει να γίνει όπως φαίνεται παρακάτω:

```
java BookRoomsServer
```

Τώρα για να τρέξουμε τον client πρέπει να γίνει όπως φαίνεται παρακάτω (ένας από τους τέσσερις τρόπους):

```
java BookRoomsClient localhost book A 5 aggelos
```

```
java BookRoomsClient localhost cancel A 5 aggelos
```

java BookRoomsClient localhost list

java BookRoomsClient localhost guests

2.3 Τρεξίματα

Αρχικά τρέχουμε τον RMI Server (BookRoomsServer).

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsServer.java && java BookRoomsServer
RMI server started
java RMI registry created.
```

Μετά τρέχουμε τον BookRoomsClient για να δούμε τα δωμάτια πρώτα.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient list localhost
40 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

Στην συνέχεια τρέχουμε πάλι το BookRoomsClient για να κλείσουμε κάποια δωμάτια.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 30  aggelos
Κράτηση 30 δωματίων τύπου A για τον/την aggelos με συνολικό κόστος 2250.0
```

Αμέσως μετά βλέπουμε τους πελάτες που έκλεισαν δωμάτιο τρέχουμε πάλι το BookRoomsClient.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient guests localhost
Ο/Η aggelos έκλεισε 30 δωμάτια τύπου A με συνολικό κόστος 2250.0
```

Αν ξανατρέξουμε το BookRoomsClient με list θα δούμε ότι τα δωμάτια μειώθηκαν.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient list localhost
10 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

Αν θέλουμε να ακυρώσουμε δωμάτια τότε θα βάλουμε την επιλογή cancel στο BookRoomsClient. Αποτέλεσμα αυτού να εκτυπώσει τον καινούργιο αριθμό δωματίων.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient cancel localhost A 10  aggelos
20 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

Όσον αναφορά το Β μέρος θα ζητήσω περισσότερα δωμάτια από αυτά που υπάρχουν με αποτέλεσμα ο πελάτης να έχει την επιλογή να πάρει όσα είναι σε υπόλοιπο. Αν πατήσει αγγλικό Y τότε θα πάρει τα 20 δωμάτια. Αν όχι θα περιμένει όπως φαίνεται στο δεύτερο τρέξιμο μέχρις ότου να ακυρωθούν τα 30 δωμάτια που ζητήθηκαν. Όταν ακυρωθούν τα 10 τότε θα βγάλει στον χρήστη ότι βρέθηκαν δωμάτια. Αν δεν θέλει να περιμένει τότε θα ακυρωθεί η κράτηση.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 30  aggelos
Δεν υπάρχουν αρκετά δωμάτια τύπου A
Υπάρχουν μόνο 20 δωμάτια διαθέσιμα
Θες να κάνεις κράτηση για 20 δωμάτια(Y/N)
Y
Κράτηση 20 δωματίων τύπου A για τον/την aggelos με συνολικό κόστος 1500.0
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 30  aggelos
Δεν υπάρχουν αρκετά δωμάτια τύπου A
Υπάρχουν μόνο 20 δωμάτια διαθέσιμα
Θες να κάνεις κράτηση για 20 δωμάτια(Y/N)
N
Θες να περιμενεις (Y/N)
Y
Περιμένεις για δωμάτια τύπου A
Ενημέρωση για διαθέσιμα δωμάτια τύπου A
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient cancel localhost A 10  aggelos
30 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

Παραπάνω είδαμε ότι ο πελάτης κλείνει δωμάτιο αν υπάρχουν έστω και 1 του βγάζει αν το θέλει. Αν επιλέξει να το πάρει θα τελειώσει το πρόγραμμα. Τώρα θα δείξω την περίπτωση να μην υπάρχει κανένα δωμάτιο που ουσιαστικά κάνει την ίδια λειτουργία με το από πάνω στην περίπτωση που θέλει να περιμένει. Ξανατρέχω από την αρχή τον server άρα ξεκινάνε όλα από την αρχή.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient list localhost
40 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 40  aggelos
Κράτηση 40 δωματίων τύπου A για τον/την aggelos με συνολικό κόστος 3000.0
```

Αφού ο πελάτης Άγγελος έκλεισε 40 δωμάτια δεν θα υπάρχουν άλλα δωμάτια τέτοιου τύπου. Αν λοιπόν πάει να κλείσει κάποιος δωμάτιο ίδιου τύπου ο Γιάννης δεν θα βρει με σκοπό αν θέλει να περιμένει. Αν αποφασίσει να περιμένει θα πρέπει ο Άγγελος να ακυρώσει δωμάτια όσα έχει ζητήσει ο Γιάννης.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 40  aggelos
Κράτηση 40 δωματίων τύπου A για τον/την aggelos με συνολικό κόστος 3000.0
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 10 giannis
Δεν υπάρχουν διαθέσιμα δωμάτια τύπου A
Θες να περιμενεις (Y/N)
Y
Περιμένεις για δωμάτια τύπου A
Ενημέρωση για διαθέσιμα δωμάτια τύπου A
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient cancel localhost A 10  aggelos
10 δωμάτια A - τιμή: 75 Ευρώ την βραδιά
35 δωμάτια B - τιμή: 110 Ευρώ την βραδιά
25 δωμάτια C - τιμή: 120 Ευρώ την βραδιά
30 δωμάτια D - τιμή: 150 Ευρώ την βραδιά
20 δωμάτια E - τιμή: 200 Ευρώ την βραδιά
```

Μόλις ακυρώθηκαν 10 δωμάτια από τον Άγγελο τότε του έβγαλε του Γιάννη μήνυμα ότι βρέθηκαν δωμάτια οπότε μπορεί να τα κλείσει. Ο Γιάννης κλείνει τα 10 δωμάτια που ήθελε και φαίνεται στην λίστα των πελατών.

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient book localhost A 10 giannis
Κράτηση 10 δωματίων τύπου Α για τον/την giannis με συνολικό κόστος 750.0
```

```
filegeiasou@filegeiasou:/mnt/c/users/fileg/OneDrive - University of West Attica/6 Semester/Distributed Systems/ΑΠ - ΕΡΓΑΣΤΗΡΙΟ/java_rmi_exercise$  javac BookRoomsClient.java && java BookRoomsClient guests localhost
Ο/Η aggelos έκλεισε 30 δωμάτια τύπου Α με συνολικό κόστος 2250.0
Ο/Η giannis έκλεισε 10 δωμάτια τύπου Α με συνολικό κόστος 750.0
```

Τέλος σε όλες τις επιλογές αν δεν δεχθεί να περιμένει δεν κλείνει δωμάτιο και η κράτηση ακυρώνεται.

3. Επίλογος

Κλείνοντας το σχολιασμό μου στο κώδικα τον οποίο υλοποίησα, έχω δώσει σωστές απαντήσεις στα ερωτήματα της άσκησης. Γίνεται σωστά η σύνδεση του RMI και υλοποιούνται όλες οι συναρτήσεις. Στα ενδεικτικά τρεξίματα βλέπουμε ότι οι κώδικες οι οποίοι έχω υλοποιήσει μεταγλωττίζονται και τρέχουν με σωστά αποτελέσματα.