Πανεπιστήμιου Δυτικής Αττικής
Σχολή Μηχανικών
Κατανεμημένα Συστήματα Εργατστήριο

Ονοματεπώνυμο: Γάγγας Ιωάννης
Αριθμός Μητρώου: 19390038
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Ημερομηνία παράδωσης εργασίας: 03/06/2024

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη μιας απλής Java RMI client/server εφαρμογής η οποία θα επιτρέπει την σύνδεση πολλαπλών χρηστών σε ένα σύστημα ξενοδοχείου, όπου θα μπορούν να κλείσουν δωμάτια, να δούνε την λίστα πελατών και δωματίων και να ακυρώσουν κρατήσεις. Για την ομαλή λειτουργία της εφαρμογής χρειάζονται τα 4 αρχεία που ζητούνται στην εκφώνηση το HRServer.java που θα αποτελεί τον remote server, το HRClient.java, που είναι ο client, το remote interface HRInterface.java και η υλοποίηση του interface αυτού που είναι το HRImpl.java. Επιπλέον, προστέθηκαν και 2 επιπλέον αρχεία για την διευκόλυνση των λειτουργιών το Hotel.java και το Customer.java.

Πρώτα έχουμε το αρχείο HRInterface.java, το οποίο περιέχει τις μεθόδους που καλούνται από τον server μαζί με το Exception handling τους.

```java
package com.HotelBooking.shared;

import java.rmi.RemoteException;
// Interface code for RMI-server
public interface HRInterface extends java.rmi.Remote {
    // Creation of the needed calls.
    public String list() throws RemoteException;

    public String book(String type, int number, String name) throws RemoteException;

    public String guests() throws RemoteException;

    public String cancel(String type, int number, String name) throws RemoteException;

}
```

Στην συνέχεια το αρχείο HRImpl.java είναι αυτό που εφαρμόζει το interface που δημιουργήσαμε παραπάνω και ταυτόχρονα εξυπηρετεί ως default contructor για τον server.

```java
package com.HotelBooking.HotelBooking.server;


import java.util.ArrayList;
import java.util.List;

import com.HotelBooking.shared.HRInterface;
// Code to implement the interface.
public class HRImpl extends java.rmi.server.UnicastRemoteObject implements HRInterface {
    // We crate a list that will store every guest that has made a reservation in our hotel.
    private final List<Customer> custList = new ArrayList<>();

    // Creation of "Hotel".
    Hotel MyHotel = new Hotel();

    public HRImpl()
            throws java.rmi.RemoteException {
        super();
    }
    // Prints a list of available room at the moment.
    @Override
    public String list()throws java.rmi.RemoteException{
        String out = null;

        out = "For the rooms that belong in category A, there are " + MyHotel.getQuant(roomType:"A") + " rooms available with a price of " + MyHotel.getprice(roomType:"A") + " for each
        out = out + "For the rooms that belong in category B, there are " + MyHotel.getQuant(roomType:"B") + " rooms available with a price of " + MyHotel.getprice(roomType:"B") + " fo
        out = out + "For the rooms that belong in category C, there are " + MyHotel.getQuant(roomType:"C") + " rooms available with a price of " + MyHotel.getprice(roomType:"C") + " fo
        out = out + "For the rooms that belong in category D, there are " + MyHotel.getQuant(roomType:"D") + " rooms available with a price of " + MyHotel.getprice(roomType:"D") + " fo
        out = out + "For the rooms that belong in category E, there are " + MyHotel.getQuant(roomType:"E") + " rooms available with a price of " + MyHotel.getprice(roomType:"E") + " fo
        return out;
    }
```

```java
// Returns the list of guests that have made a reservation.
@Override
public String guests() throws java.rmi.RemoteException{
    String out = null;
    // Variable to store the size of guestlist.
    int guest = custList.size();
    out = "There are "+guest+" People that have booked a room in our hotel.\n";
    System.out.println(out);
    // For loop that print the guests and their bookings.
    for(Customer d: custList){
        out = out + d.getName()+" has "+d.getRoomCount()+" rooms, category: "+d.getRoomType()+".\n";
        System.out.println(d.getName()+" has "+d.getRoomCount()+" rooms, category:  "+d.getRoomType()+".\n");
    }
    return out;
}
// Cancels a booking that has been made by a guest.
@Override
public String cancel(String type, int number, String name)throws java.rmi.RemoteException{
    String answer = null;
    boolean exist = false;
    int pos = 0;
    for(Customer d : custList){
        if(d.getName()!=null && d.getName().equalsIgnoreCase(name)){
            exist = true;
            pos = custList.indexOf(d);
        }
    }
    if(!exist){
        answer = "There is no Customer with that name.";
        return answer;
    }
    Customer MyCustomer = custList.get(pos);
    if(MyHotel.cancel(MyCustomer, type, number)){
        MyHotel.updateAvail(type, -number);
        MyCustomer.setRoomCount(MyCustomer.getRoomCount() - number);
    }
    // Prints the rooms that the guest has booked after the cancelation.
    answer = "Customers remaining rooms "+MyCustomer.getRoomCount()+".";
    return answer;
}
```

Παρακάτω παρουσιάζεται το αρχείο του server HRServer.java. Με την χρληση του registry γίνοται τα απαιτούμενα imports ενώ ο server συνδέεται στο default port 1099.

```java
package com.HotelBooking.HotelBooking.server;

import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.UnknownHostException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

import com.HotelBooking.shared.HRInterface;

public class HRServer {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) throws RemoteException{
        try {
            HRInterface server = new HRImpl();
            Registry registry = LocateRegistry.createRegistry(1099);
            String url = "rmi://" + InetAddress.getLocalHost().getHostAddress() + ":1099/HRImpl";
            Naming.rebind(url, server);  //create rmi server
            System.out.println("Server is running.");
        } catch (MalformedURLException | UnknownHostException | RemoteException e){
        System.out.println("Trouble: " + e );}
    }
}
```

Έπειτα περνάμε στο αρχείο του client HRClient.java. Αρχικά, ο client συνδέεται στον διακομιστή μέσω του String το οποίο δίνεται στην naming.lookup για την κλήση των συναρτήσεων που δημιουργούνται παρακάτω.

```
public class HRClient extends UnicastRemoteObject   {
    protected HRClient() throws RemoteException{
    }

    Run main | Debug main | Run | Debug
    public  static void main(String[] args){
        String error = "INVALID NUMBER OF ARGUMENTS!";
        try{
            String url = "rmi://" + InetAddress.getLocalHost().getHostAddress() + ":1099/HRImpl";
            Remote remoteObject = Naming.lookup(url);
            HRInterface Rserver = (HRInterface) remoteObject;

            System.out.println("Welcome to our Hotel Booking System.");
```

Η επιλογή των ενεργειών του χρήστη αναγνωρίζεται από τον αριθμό των παραμέτρων που δίνονται από τον χρήστη κάθε φορά που τρέχει το πρόγραμμα. Στην περίπτωση που δεν δωθεί καμία παράμετρος εκτιπώνονται οι επιλογές που έχει ο χρήστης. Αν δωθεί έστω μία παράμετρος τότε ενεργοποιείται το case το οποίο αναγνωρίζει τις εντολές “list”, “book”, “guests” και “cancel” με τις υπόλοιπες μεταβλητές να καταχωρούνται στις αντίστοιχες παραμέτρους args[1], args[2], args[3].

```
} else {
    switch (args[0]) {
        case "list" -> {
            if (args.length == 1) {
                System.out.println(Rserver.list());
            } else {
                System.err.println(error);
                System.exit(1);
            }
            }
        case "book" -> {
            if (args.length == 4) {
                String answer = null;
                //Integer.valueOf(args[2]),
                answer = Rserver.book(args[1], Integer.parseInt(args[2]), args[3]);
                System.out.println(answer);
                if (answer.contains("available")) {
                    try (Scanner myObj = new Scanner(System.in)) {
                        String answerscan = myObj.nextLine();
                        if (answerscan.equalsIgnoreCase("Y")) {
                        }
                    }
                }
            } else {
                System.err.println(error);
                System.exit(1);
            }
            }
        case "guests" -> {
            if (args.length == 1) {
                System.out.print(Rserver.guests());
            } else {
                System.err.println(error);
                System.exit(1);
            }
            }
        case "cancel" -> {
            if (args.length == 4) {
                System.out.print(Rserver.cancel(args[1], Integer.parseInt(args[2]), args[3]));
            } else {
                System.err.println(error);
                System.exit(1);
            }
            }
        default -> printOptions();
```

```
    public static void printOptions() { //usage
        System.out.println("""
                           ~Menu~
                           1. java HRClient list: Display all available rooms with information on them
                           2. java HRClient book:<type> <number> <name>: Makes a reservation of <numbers> rooms that are of type <type> in the name of <name>
                           3. java HRClient guests: List of clients and their bookings
                           4. java HRClient cancel: <type> <number> <name>: Cancels the booking of <name> that are type <type> and add up to <number. Also returns a list of available r
                           """);
    }
}
```

Επιπρόσθετα υπάρχουν τα δύο αρχεία Customer.java και Hotel.java. Το πρώτο χρησιμοποιείται για την επιστροφή των κραστήσεων αν δωθεί η επιλογή “guests” από τον χρήστη. Το δεύτερο αρχείο περιέχει τις συναρτήσεις που καλούνται από τον client. Η κλάση Hotel έχει τα δεδομένα για κάθε τύπο δωματίου, και αποθηκεύει τις κρατήσεις που έχουν γίνει. Οι getprice και getQuant αντιστοιχούν τις τιμές στα δωμάτια κια την εναπομήναντα διαθεσιμότητα. Η διαχείρηση των πινάκων είναι αναθετημένη στην updateAvail σε συνεργασία με την book που ελέγχει αν η κράτηση είναι εφικτή λόγω διαθεσημότητας. Τέλος η cancel αφαιρεί μία κράτηση από το αντίστοιχο δωμάτιο (κομμάτι του πίνακα).

```
package com.HotelBooking.HotelBooking.server;

public class Customer {
    private String name;
    private String roomType;
    private int roomCount;

    public Customer(String name, String roomType, int roomCount){
        this.name = name;
        this.roomCount = roomCount;
        this.roomType = roomType;

    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getRoomType(){
        return this.roomType;
    }

    public void setRoomType(String roomType){
        this.roomType = roomType;
    }

    public int getRoomCount(){
        return this.roomCount;
    }

    public void setRoomCount(int roomCount){
        this.roomCount = roomCount;
    }

}
```

```
protected Hotel(){
    // Price for each category A,B,C,D,E
    price[0] = 60;
    price[1] = 80;
    price[2] = 90;
    price[3] = 115;
    price[4] = 140;

    // Quantity for each category A,B,C,D,E
    // Single rooms
    quantity[0] = 25;
    // Double rooms
    quantity[1] = 40;
    // Twin rooms
    quantity[2] = 20;
    // Triple rooms
    quantity[3] = 15;
    // Quad rooms
    quantity[4] = 10;

}
```

```
public class Hotel {
    public int getprice(String roomType){
        int pri = 0;
        switch (roomType) {
            case "A" -> pri = this.price[0];
            case "B" -> pri = this.price[1];
            case "C" -> pri = this.price[2];
            case "D" -> pri = this.price[3];
            case "E" -> pri = this.price[4];
            default -> {
            }
        }
        return pri;
    }

    public int getQuant(String roomType){
        int quant = 0;
        switch (roomType) {
            case "A" -> quant = this.quantity[0];
            case "B" -> quant = this.quantity[1];
            case "C" -> quant = this.quantity[2];
            case "D" -> quant = this.quantity[3];
            case "E" -> quant = this.quantity[4];
            default -> {
            }
        }
        return quant;
    }
```

```java
public void updateAvail(String roomType, int roomCount){
        case "A":
            this.quantity[0] = quantity[0] - roomCount;
            break;
        case "B":
            this.quantity[1] = quantity[1] - roomCount;
            break;
        case "C":
            this.quantity[2] = quantity[2] - roomCount;
            break;
        case "D":
            this.quantity[3] = quantity[3] - roomCount;
            break;
        case "E":
            this.quantity[4] = quantity[4] - roomCount;
            break;
        default:
            throw new IllegalStateException("UNEXPECTED VALUE: " + roomType);
    }
}

protected boolean book(String roomType, int roomCount){
    boolean answer = false;
    switch (roomType) {
        case "A" -> {
            if (this.quantity[0] >= roomCount) answer = true;
        }
        case "B" -> {
            if (this.quantity[1] >= roomCount) answer = true;
        }
        case "C" -> {
            if (this.quantity[2] >= roomCount) answer = true;
        }
        case "D" -> {
            if (this.quantity[3] >= roomCount) answer = true;
        }
        case "E" -> {
            if (this.quantity[4] >= roomCount) answer = true;
        }
        default -> answer = false;
    }
    return answer;
}

public boolean cancel(Customer customer,String roomType,int roomCount){
    // This class checks if the customers has booked the booking he is trying to cancel
    return customer.getRoomCount() >= roomCount && customer.getRoomType().equals(roomType);
}
```

Στην εκτέλεση του προγράμματος αντιμετώπισα κάποια προβλήματα κατά την δήλωση των παραμέτρων. Για τον λόγο αυτόν έχω δημιουργήσει ένα αρχείο .json μέσω του οποίου τρέχω το πρόγραμμα. Κάθε φορά που θέλω να κάνω κλήση των book, cancel αλλάζω απλά τις παραμέτρους εντός της αντισοτιχης περίπτωσης τρεξήματος στο αρχείο.

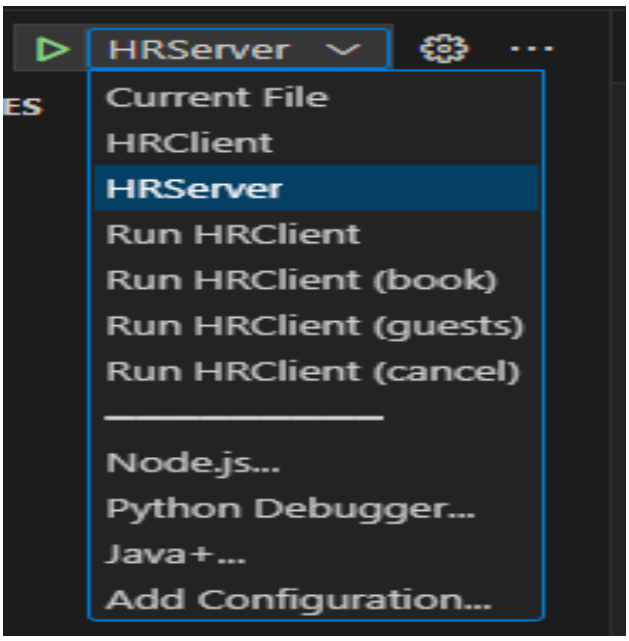
```json
        {
            "type": "java",
            "name": "Run HRClient",
            "request": "launch",
            "mainClass": "com.HotelBooking.HotelBooking.client.HRClient",
            "args": "list"
        },
        {
            "type": "java",
            "name": "Run HRClient (book)",
            "request": "launch",
            "mainClass": "com.HotelBooking.HotelBooking.client.HRClient",
            "args": "book A 1 John"
        },
        {
            "type": "java",
            "name": "Run HRClient (guests)",
            "request": "launch",
            "mainClass": "com.HotelBooking.HotelBooking.client.HRClient",
            "args": "guests"
        },
        {
            "type": "java",
            "name": "Run HRClient (cancel)",
            "request": "launch",
            "mainClass": "com.HotelBooking.HotelBooking.client.HRClient",
            "args": "cancel A 1 John"
        }
    ]
}
```

Ενδεικτικά τρεξήματα

- Πρώτα τρέχουμε τον διακομιστή

```
Server is running.
```

- Στην συνέχεια τρέχουμε ενδεικτικά κάθε μία από τις εντολές που ζητούνται στην εργασία μέσω του drop-menu του αρχείου json.



- Επιλογή “HRClient” που αντιστοιχεί στο τρέξημο του client χωρίς παράμετρο

```
Welcome to our Hotel Booking System.
~Menu~
1. java HRClient list: Display all available rooms with information on them
2. java HRClient book:<type> <number> <name>: Makes a reservation of <numbers> rooms that are of type <type> in the name of <name>
3. java HRClient guests: List of clients and their bookings
4. java HRClient cancel: <type> <number> <name>: Cancels the booking of <name> that are type <type> and add up to <number. Also returns a list of available rooms>
```

- Επιλογή “Run HRClient” που αντιστοιχεί στην επιλογή “list”

```
Welcome to our Hotel Booking System.
For the rooms that belong in category A, there are 25 rooms available with a price of 60 for each one.
For the rooms that belong in category B, there are 40 rooms available with a price of 80 for each one.
For the rooms that belong in category C, there are 20 rooms available with a price of 90 for each one.
For the rooms that belong in category D, there are 15 rooms available with a price of 115 for each one.
For the rooms that belong in category E, there are 10 rooms available with a price of 140 for each one.
```

- Επιλογή “Run HRClient (book)” που αντιστοιχεί στην επιλογή “book”, με μεταβλητές “A 1 John”

```
Welcome to our Hotel Booking System.
BOOKING SUCCESFULL, YOUR BILL IS: 60 ?
```

- Επιλογή “Run HRClient (guests)” που αντιστοιχεί στην επιλογή “guests”

```
Welcome to our Hotel Booking System.
There are 1 People that have booked a room in our hotel.
John has 1 rooms, category: A.
```

- Επιλογή “Run HRClient (cancel)” που αντιστοιχεί στην επιλογή “cancel”, με

μεταβλητές “A 1 John”

```
Welcome to our Hotel Booking System.
Customers remaining rooms 0.
```

- Καταχώρηση τυχαίων πελατών με τυχαία δεδομένα

```
PS D:\Σχολή\Κατανεμημένα Συστήματα (6ο εξάμηνο)\19390038_ΓΑΓΓΑΣ_ΙΩΑΝΝΗΣ_ΧΠ_DS2>
.

Welcome to our Hotel Booking System.
BOOKING SUCCESFULL, YOUR BILL IS: 540 ?
PS D:\Σχολή\Κατανεμημένα Συστήματα (6ο εξάμηνο)\19390038_ΓΑΓΓΑΣ_ΙΩΑΝΝΗΣ_ΧΠ_DS2>
PS D:\Σχολή\Κατανεμημένα Συστήματα (6ο εξάμηνο)\19390038_ΓΑΓΓΑΣ_ΙΩΑΝΝΗΣ_ΧΠ_DS2>

Welcome to our Hotel Booking System.
BOOKING SUCCESFULL, YOUR BILL IS: 420 ?
PS D:\Σχολή\Κατανεμημένα Συστήματα (6ο εξάμηνο)\19390038_ΓΑΓΓΑΣ_ΙΩΑΝΝΗΣ_ΧΠ_DS2>
PS D:\Σχολή\Κατανεμημένα Συστήματα (6ο εξάμηνο)\19390038_ΓΑΓΓΑΣ_ΙΩΑΝΝΗΣ_ΧΠ_DS2>

Welcome to our Hotel Booking System.
BOOKING SUCCESFULL, YOUR BILL IS: 60 ?
```

- Υλοποιήση της “guests”

```
Welcome to our Hotel Booking System.
There are 16 People that have booked a room in our hotel.
John has 0 rooms, category: A.
John has 1 rooms, category: A.
Raul has 5 rooms, category: E.
Ilias has 2 rooms, category: E.
Jim has 3 rooms, category: C.
Garen has 2 rooms, category: B.
Eleni has 5 rooms, category: A.
Marileni has 7 rooms, category: D.
Maria has 1 rooms, category: B.
Lampros has 6 rooms, category: C.
Jamal has 3 rooms, category: E.
Panos has 1 rooms, category: A.
Bryan has 2 rooms, category: D.
Ben has 7 rooms, category: A.
Camilla has 8 rooms, category: C.
Niki has 5 rooms, category: D.
```

- Υλοποιήση της “lists”

```
Welcome to our Hotel Booking System.
For the rooms that belong in category A, there are 11 rooms available with a price of 60 for each one.
For the rooms that belong in category B, there are 37 rooms available with a price of 80 for each one.
For the rooms that belong in category C, there are 3 rooms available with a price of 90 for each one.
For the rooms that belong in category D, there are 1 rooms available with a price of 115 for each one.
For the rooms that belong in category E, there are 0 rooms available with a price of 140 for each one.
```

- Διαγραφή 6 δωματίων απο “Camilla”

```
Welcome to our Hotel Booking System.
Customers remaining rooms 2.
```

- Διαγραφή μη καταχωρημένου πελάτη “George”

```
Welcome to our Hotel Booking System.
There is no Customer with that name.
```