Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Υπολογιστών

Εργαστήριο Κατανεμημένων Συστημάτων - Εργασία 2

Χρήστος Μαργιώλης – 19390133

Μάιος 2022

Περιεχόμενα

1 Δομή αρχείων

- `HRInterface`: Interface που περιέχει τις δηλώσεις των μεθόδων του server.
- `HRImpl`: Υλοποίηση του interface.
- `Room`: Βοηθητική κλάση για την υλοποίηση των δωματίων του ξενοδοχείου.
- `HRServer`: Ο server.
- `HRClient`: Ο client.

2 Εκτέλεση κώδικα

Κάνουμε compile τον κώδικα:

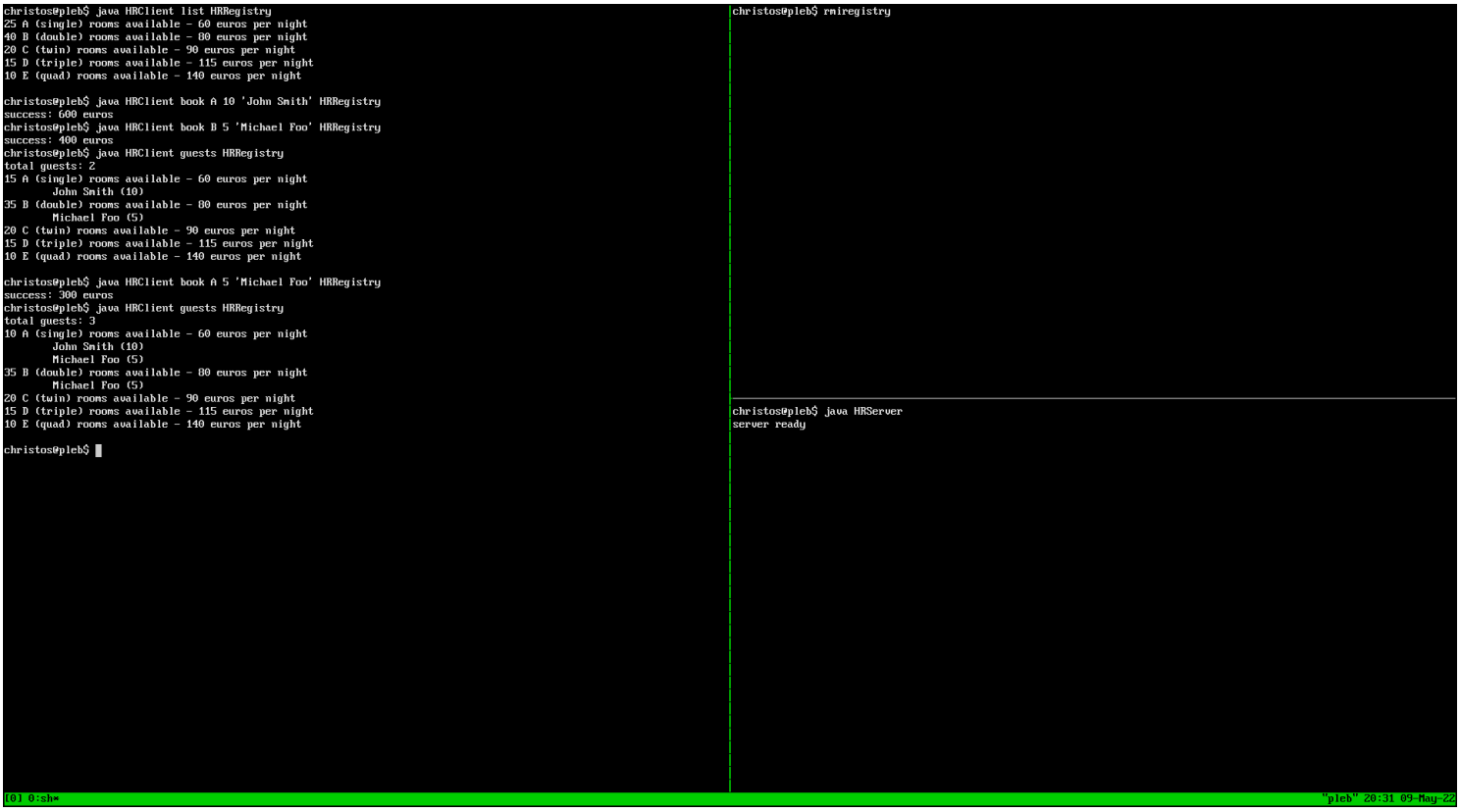
```
$ javac *.java
```

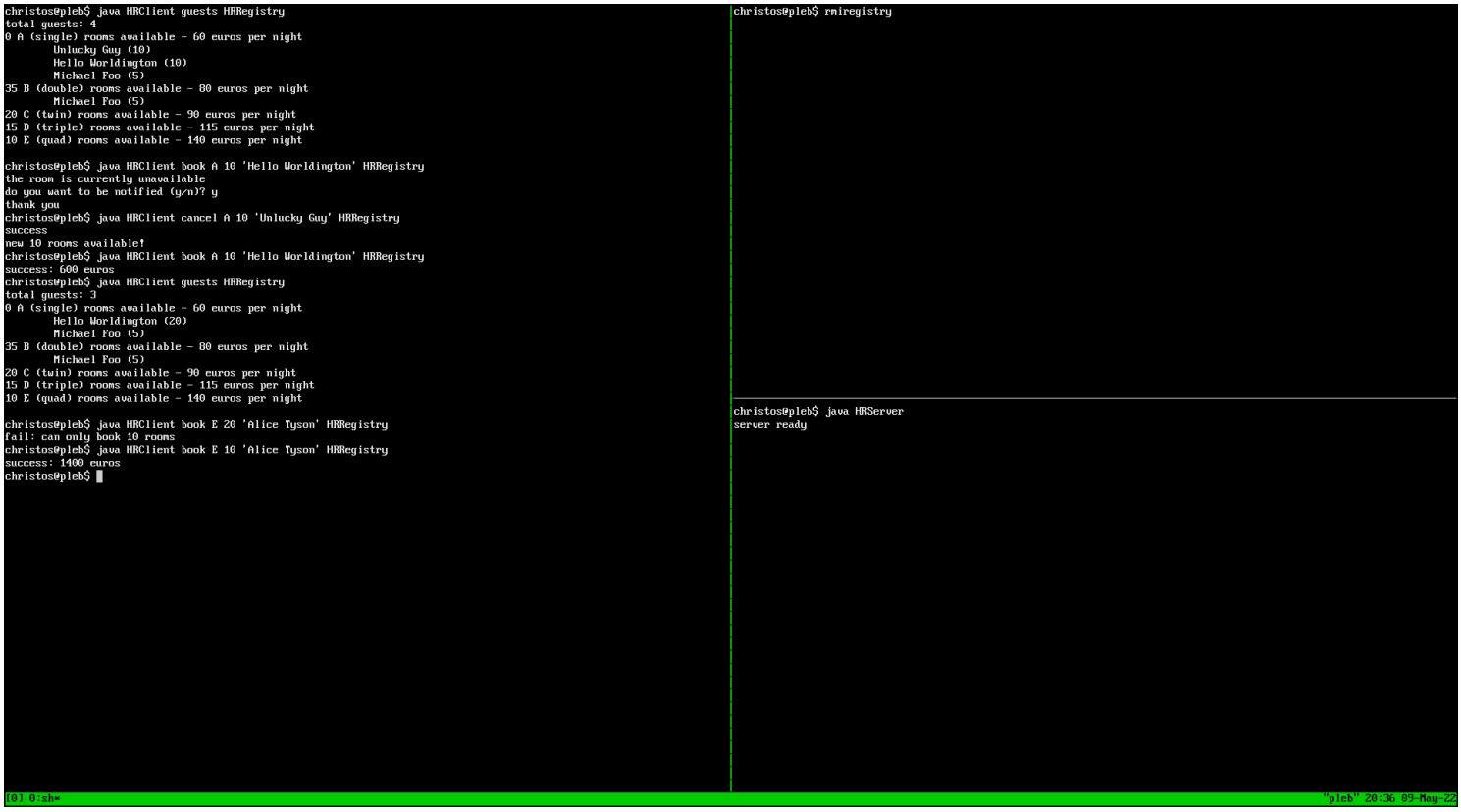
Ανοίγουμε δύο terminals. Στο ένα ξεκινάμε το `rmiregistry` και στο άλλο τον server. Σε τρίτο terminal εκτελούμε τον client, ο οποίος μπορεί να εκτελεστεί με έναν από τους 4 τρόπους:

```
java HRClient list <hostname>
java HRClient book <type> <number> <name> <hostname>
java HRClient guests <hostname>
java HRClient cancel <type> <number> <name> <hostname>
```

3 Ενδεικτικά τρεξίματα

Τα παρακάτω τρεξίματα δείχνουν τους 4 διαφορετικούς τρόπους εκτέλεσης του client, καθώς και τους χειρισμούς περιπτώσεων που μπορούν να προκείψουν (π.χ δεν υπάρχουν ελεύθερα δωμάτια, υπάρχουν λιγότερα δωμάτια από όσα θέλει να κρατήσει ο πελάτης)


christos@pleb$ java HRClient list HRRegistry
25 A (single) rooms available - 60 euros per night
40 B (double) rooms available - 80 euros per night
20 C (twin) rooms available - 90 euros per night
15 D (triple) rooms available - 115 euros per night
10 E (quad) rooms available - 140 euros per night
christos@pleb$ java HRClient book A 10 'John Smith' HRRegistry
success: 600 euros
christos@pleb$ java HRClient book B 5 'Michael Foo' HRRegistry
success: 400 euros
christos@pleb$ java HRClient guests HRRegistry
total guests: 2
15 A (single) rooms available - 60 euros per night
John Smith (10)
35 B (double) rooms available - 80 euros per night
Michael Foo (5)
20 C (twin) rooms available - 90 euros per night
15 D (triple) rooms available - 115 euros per night
10 E (quad) rooms available - 140 euros per night
christos@pleb$ java HRClient book A 5 'Michael Foo' HRRegistry
success: 300 euros
christos@pleb$ java HRClient guests HRRegistry
total guests: 3
10 A (single) rooms available - 60 euros per night
John Smith (10)
Michael Foo (5)
35 B (double) rooms available - 80 euros per night
Michael Foo (5)
20 C (twin) rooms available - 90 euros per night
15 D (triple) rooms available - 115 euros per night
10 E (quad) rooms available - 140 euros per night
christos@pleb$
christos@pleb$ rmiregistry
christos@pleb$ java HRServer
server ready
[0] 0:sh*
"pleb" 20:31 09-May-22



4 Κώδικας

Ο κώδικας είναι σχολιασμένος στα σημεία που θεωρώ ότι μπορεί να υπάρξει σύχγηση, και όχι ακόμα και σε σημεία που είναι λίγο-πολύ ξεκάθαρο το τι συμβαίνει.

4.1 HRInterface.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface HRInterface extends Remote {
        String list() throws RemoteException;
        String book(String type, int num, String name) throws RemoteException;
        String guests() throws RemoteException;
        String cancel(String type, int num, String name) throws RemoteException;
        void addListSub(String name, String type) throws RemoteException;
}
```

4.2 HRImpl.java

```
public class HRImpl implements HRInterface {
        Room[] rooms = {
                new Room("A", "single", 60, 25),
                new Room("B", "double", 80, 40),
                new Room("C", "twin", 90, 20),
                new Room("D", "triple", 115, 15),
                new Room("E", "quad", 140, 10),
        };

        public HRImpl() {
                super();
        }

        public String list() {
                String str = "";

                for (Room r : rooms)
                        str += r.toString() + "\n";
                return str;
        }

        public String book(String type, int num, String name) {
                for (Room r : rooms) {
                        if (type.equals(r.type))
                                return r.addGuest(name, num);
                }
                return fail();
        }

        public String guests() {
                String str = "";
                int n = 0;

                for (Room r : rooms)
                        n += r.totalGuests();
                str += "total guests: " + n + "\n";
                for (Room r : rooms) {
                        str += r.toString() + "\n";
                        if ((n = r.totalGuests()) != 0)
                                str += r.listGuests();
                }
                return str;
        }
```

```
        public String cancel(String type, int num, String name) {
                for (Room r : rooms) {
                        if (type.equals(r.type))
                                return r.removeReserv(name, num);
                }
                return fail();
        }

        public void addListSub(String name, String type) {
                for (Room r : rooms) {
                        if (type.equals(r.type)) {
                                r.addSub(name);
                                break;
                        }
                }
        }

        private String fail() {
                return "fail: room doesn't exist. available rooms types:\n" +
                    list();
        }
}
```

4.3 Room.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class Room {
        public String type;
        public String desc;
        public int price;
        public int avail;
        private HashMap<String, Integer> guests = new HashMap<String, Integer>();
        private ArrayList<String> listsubs = new ArrayList<String>();

        public Room(String type, String desc, int price, int avail) {
                this.type = type;
                this.desc = desc;
                this.price = price;
                this.avail = avail;
        }

        public String addGuest(String name, int num) {
                if (num <= 0)
                        return "fail: cannot book <= 0 rooms";
                else if (avail - num >= 0) {
                        guests.put(name, guests.getOrDefault(name, 0) + num);
                        avail -= num;
                        return "success: " + num * price + " euros";
                } else if (avail > 0)
                        return "fail: can only book " + avail + " rooms";
                else
                        return "fail: no rooms available";
        }

        public String removeReserv(String name, int num) {
                String str = "";
                int n;

                if (!guests.containsKey(name))
                        return "guest '" + name + "' doesn't exist";

                /* make sure he doesn't cancel more than he has booked */
                if ((n = guests.get(name)) >= num) {
                        guests.put(name, n - num);
                        avail += num;
                        n -= num;
```

```
                /* remove guest if he has 0 reservations */
                if (n == 0) {
                        guests.remove(name);
                        str += "success";
                } else
                        str += "success: " + n + " reservations left";
                str += notifySubs(name);
                return str;
        } else
                return "fail: cannot remove more (" + num +
                    ") than booked (" + n + ")";
    }

    public String listGuests() {
        String str = "";

        for (Map.Entry<String, Integer> g : guests.entrySet())
                str += "\t" + g.getKey() + " (" + g.getValue() + ")\n";
        return str;
    }

    public int totalGuests() {
        return guests.size();
    }

    public void addSub(String name) {
        listsubs.add(name);
    }

    public String notifySubs(String name) {
        if (listsubs.contains(name))
                return "\nnew " + avail + " rooms available!";
        else
                return "";
    }

    public String toString() {
        return avail + " " + type + " (" + desc +
            ") rooms available - " + price + " euros per night";
    }
}
```

4.4 HRServer.java

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class HRServer extends HRImpl {
        public static void main(String[] args) {
                try {
                        HRImpl impl = new HRImpl();
                        HRInterface stub = (HRInterface)UnicastRemoteObject.exportObject(impl, 0);
                        Registry reg = LocateRegistry.getRegistry();

                        reg.bind("HRRegistry", stub);
                        System.out.println("server ready");
                } catch (Exception e) {
                        System.err.println("server exception: " + e.toString());
                        e.printStackTrace();
                }
        }
}
```

4.5 HRClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.NotBoundException;
import java.util.Scanner;

public class HRClient {
        public static void usage() {
                System.err.println("usage: java HRClient list <hostname>");
                System.err.println("       java HRClient book <type> <number> <name> <hostname>");
                System.err.println("       java HRClient guests <hostname>");
                System.err.println("       java HRClient cancel <type> <number> <name> <hostname>");
                System.exit(1);
        }

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                String str = "";

                try {
                        if (args.length != 2 && args.length != 5)
                                usage();
                        String host = args[args.length-1];
                        HRInterface stub = (HRInterface)Naming.lookup(host);

                        if (args[0].equals("list") && args.length == 2)
                                str = stub.list();
                        else if (args[0].equals("book") && args.length == 5) {
                                str = stub.book(args[1], Integer.parseInt(args[2]), args[3]);
                                if (str.equals("fail: no rooms available")) {
                                        System.out.println("the room is currently unavailable");
                                        System.out.print("do you want to be notified (y/n)? ");
                                        if (sc.next().equals("y")) {
                                                stub.addListSub(args[3], args[1]);
                                                System.out.println("thank you");
                                                return;
                                        }
                                }
                        } else if (args[0].equals("guests") && args.length == 2)
                                str = stub.guests();
                        else if (args[0].equals("cancel") && args.length == 5)
                                str = stub.cancel(args[1], Integer.parseInt(args[2]), args[3]);
                        else
                                usage();
                } catch (RemoteException e) {
```

```
                System.err.println("RemoteException: " + e.toString());
            } catch (NotBoundException e) {
                System.err.println("NotBoundException: " + e.toString());
            } catch (IndexOutOfBoundsException e) {
                usage();
            } catch (Exception e) {
                System.err.println("Exception: " + e.toString());
            }

            System.out.println(str);
        }
}
```