ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εργασία 2 – Java RMI

Ονοματεπώνυμο: Σηφάκης Σπυρίδων

ΑΜ: 20390208

Τμήμα: ΣΤ3-Β (Τετάρτη 2-4)

Ημ παράδοσης: 4/6/23

Περιεχόμενα

Η εργασία υλοποιεί μία επικοινωνία μεταξύ client και server με την βοήθεια του RMIregistry. Για να το επιτύχουμε αυτό πρέπει να εκτελέσουμε σε έναν τερματικό την εντολή “rmiregistry 1099” ο οποίος είναι και το default port του “rmiregistry”. Παρακάτω θα αναπτυχθεί και θα σχολιαστεί ο κώδικας που έχει υλοποιηθεί για την επίλυση του προβλήματος.

THInterface.java

Αρχίζοντας από το αρχείο THInterface.java:

```
import java.rmi.*;
import java.util.ArrayList;

//αρχικοποίηση συναρτήσεων
public interface THInterface extends Remote{
    String clist() throws RemoteException;
    int book(String type, int num, String name) throws
RemoteException;
    ArrayList<ArrayList<String>> guest() throws RemoteException;
    ArrayList<String> cancel(String type,int num,String name) throws
RemoteException;
}
```

Το αρχείο THInterface.java είναι ο τρόπος ο οποίος επικοινωνεί ο Client με τον Server. Ουσιαστικά, έχει μέσα όλες τις συναρτήσεις τις οποίες θα χρειαστούν στον client και στον server. Το Interface έχει μόνο την δήλωση των συναρτήσεων και όχι και τις υλοποιήσεις τους. Οι υλοποιήσεις των συναρτήσεων γίνονται στο αρχείο THImpl.java όπως θα αναλυθεί παρακάτω.

THImpl.java

Για την καλύτερη εμπέδωση της Implementation συνάρτησης, δηλαδή της υλοποίησης των συναρτήσεων, θα αναλυθεί ο κώδικας σε κομμάτια με σκοπό και την ευκολότερη κατανόηση του προγράμματος αλλά και την ευκολότερη ανάγνωση του προγράμματος.

```
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.RemoteException;
import java.util.ArrayList;

//λειτουργίες συναρτήσεων
public class THImpl extends UnicastRemoteObject implements
THInterface{
```

Στο συγκεκριμένο μέρος του κώδικα εισάγονται οι κατάλληλες βιβλιοθήκες όπως οι java.rmi όπως και java.rmi.RemoteException και δημιουργεί την κλάση του THImpl που κληρονομεί από την UnicastRemoteObject με την γραμμή “extends UnicastRemoteObject”. Υλοποιεί τις συναρτήσεις της κλάσης THInterface με το “implements THInterface”.

Global μεταβλητές

Συνεχίζοντας, δημιουργούνται global μεταβλητές για να μπορούμε να τις χρησιμοποιούμε σε όλες τις συναρτήσεις που θα ασχοληθούμε στην συνέχεια. Δημιουργούμε 3 διαφορετικών ειδών μεταβλητών που θα προσφέρουν παρακάτω χρησιμότητα. Οι πρώτες πέντε μεταβλητές που περιέχουν την λέξη size ή s αναφέρονται στο αρχικό μέγεθος των πινάκων και είναι τύπου int . Μετέπειτα, οι μεταβλητές που αρχίζουν με closed αναφέρονται στις κλεισμένες θέσεις που έχουν αγοράσει οι χρήστες και είναι ακέραιοι. Τέλος, φτιάχνονται λίστες τύπου string που εκεί θα αποθηκευτούν τα στοιχεία των πελατών οι οποίοι αγόρασαν εισητήριο για θέσεις.

```
//public global μεταβλητές
    public static int sizeA=100;
    public static int sizeB=200;
    public static int sizeC=400;
    public static int scntr=225;
    public static int sside=75;

    public static int closedA=0;
    public static int closedB=0;
    public static int closedC=0;
    public static int closedcntr=0;
    public static int closedside=0;

    public static ArrayList<String> A=new ArrayList<>();
    public static ArrayList<String> B=new ArrayList<>();
    public static ArrayList<String> C=new ArrayList<>();
    public static ArrayList<String> cntr=new ArrayList<>();
    public static ArrayList<String> side=new ArrayList<>();
```

Κατασκευαστής της THImpl

Παρακάτω παρατηρούμε τον κατασκευαστή της κλάσης THImpl το οποίο καλεί την super(). Η super() καλεί τον «γονικό» κατασκευαστή και για κάθε έναν πίνακα γεμίζει τον πίνακα με κενούς χαρακτήρες (“”). Αρχίζει από το μεγαλύτερο σε μέγεθος πίνακα και μετά για τα στοιχεία τα οποία είναι όσα και οι άλλοι πίνακες εκχωρεί και εκεί το κενο χαρακτήρα.

```
public THImpl() throws RemoteException{
        super(0);
        int i;
        for(i=0;i<sizeC;i++)
            if(i<sizeA)
                A.add("");
            if(i<sizeB)
                B.add("");
            if(i<scntr)
                cntr.add("");
            if(i<sside)
                side.add("");
            C.add("");
    }
```

Υλοποίση της clist() (2ο ερώτημα)

Η επόμενη συνάρτηση είναι και η απάντηση στο 2ο ερώτημα της άσκησης. Η clist εμφανίζει τις ελεύθερες θέσεις και επιστρέφει ένα string με τις ελεύθερες θέσεις και τις λεπτομέρειες αυτών. Η συνάρτηση δημιουργεί 5 προσωρινές μεταβλητές ακεραίων από t1 – t5. Ακόμα ελέγχει αν οι θέσεις έχουν αρνητικό αριθμό. Αυτό μπορεί να συμβεί σε περίπτωση που κάποιος προσπαθήσει να αγοράσει εισιτήριο χωρίς να υπάρχουν αρκετά. Σε αυτή τη περίπτωση το κάνει 0 και το βάζει στην μεταβλητη t1. Παρομοίως και με τα άλλα 4. Από κάτω, αρχικοποιώ 5 μεταβλητές τύπου String με το ανάλογο t και την περιγραφή του από την εργασία. Τέλος επιστρέφει ένα String το οποίο συνδιάζει όλους τους πίνακες μαζί.

```
public String clist(){
        int t1,t2,t3,t4,t5;
        if(sizeA-closedA<0) t1=0;
        else t1=sizeA-closedA;

        if(sizeB-closedB<0) t2=0;
        else t2=sizeB-closedB;

        if(sizeC-closedC<0) t3=0;
        else t3=sizeC-closedC;
```

```
        if(scntr-closedcntr<0) t4=0;
        else t4=scntr-closedcntr;

        if(sside-closedside<0) t5=0;
        else t5=sside-closedside;

        String tmp1=t1+" θέσεις Πλατεία - Ζώνη Α (κωδικός: ΠΑ) - τιμή:
45 Ευρώ\n";
        String tmp2=t2+" θέσεις Πλατεία - Ζώνη Β (κωδικός: ΠΒ) - τιμή:
35 Ευρώ\n";
        String tmp3=t3+" θέσεις Πλατεία - Ζώνη Γ (κωδικός: ΠΓ) - τιμή:
25 Ευρώ\n";
        String tmp4=t4+" θέσεις Κεντρικός Εξώστης (κωδικός: ΚΕ) -
τιμή: 30 Ευρώ\n";
        String tmp5=t5+" θέσεις Πλαϊνά Θεωρεία (κωδικός: ΠΘ) - τιμή:
20 Ευρώ\n";
        return tmp1+tmp2+tmp3+tmp4+tmp5;
    }
```

Υλοποίηση της book (3ο ερώτημα)

Για να κάνει κράτηση θέσεων ο χρήστης χρησιμοποιώντας τα ορίσματα που έχει δώσει, εκτελεί η συνάρτηση μία switch σε σχέση με το type για να δει αν επιθυμεί πλατεία Α(ΠΑ), πλατεία Β(ΠΒ), πλατεία Γ(ΠΓ), Κεντρικό εξώστη(ΚΕ) και Πλαϊνά Θεωρεία(ΠΘ). Για κάθε περίπτωση από τις παραπάνω ελέγχει πρώτα αν το μέγεθος του πίνακα μείον το νούμερο που θέλει να αφαιρέσει. Αν το αποτέλεσμα είναι μεγαλύτερο ή ίσο του μηδενός τότε μειώνει τη μεταβλητή που αναφέρεται στο μέγεθος του πίνακα και αυξάνει κατά num τις κλεισμένες θέσεις. Μετά για κάθε θέση που εισάγει προσθέτει στην λίστα το όνομα του, την τιμή του και τον τύπο τον οποίο έκλεισε με την add. Βέβαια η λίστα περιέχει διπλάσιο μέγεθος γιατί στον κατασκευαστή της συνάρτησης κάναμε add() το κενό χαρακτήρα. Άρα θα έχει διπλάσιο μέγεθος λίστας. Το πιο σωστό ίσως να ήταν να γίνει με την συνάρτηση set αλλά δεν μου έτρεχε έτσι όπως θα ήθελα. Οπότε διατήρησα την add. Η συνάρτηση επιστρέφει το κόστος. Στην περίπτωση που ο χρήστης δεν μπορεί να κλείσει όσα εισιτήρια επιθυμεί επιστρέφεται το μέγεθος που υπάρχουν διαθέσιμες θέσεις αλλά με αρνητικό πρόσημο. Αυτό το σκέφτηκα διότι το κόστος δεν μπορεί να είναι ποτέ αρνητικό. Αν επιστρέψει αρνητικό σημαίνει ότι μπήκε στην περίπτωση που δεν επαρκούν και όπως θα δούμε πιο μετά στον client το ξαναστέλνει με το μέγεθος που επέστρεψε η συνάρτηση αλλά με αρνητικό πρόσημο. Θα εξηγηθεί περαιτέρω στον client.

```
public int book(String type,int num,String name){
        int i;
        switch(type){
            case "ПА":
                if(sizeA-num>=0){
                    sizeA-=num;
                    closedA+=num;
                    for(i=0;i<num;i++)
                        A.add("name = "+name+" price = "+num*45+" type
= "+type);
                    return num*45;
                } else{return -sizeA;}
            case "ПВ":
                if(sizeB-num>=0){
                    sizeB-=num;
                    closedB+=num;
                    for(i=0;i<num;i++)
                        B.add("name = "+name+" price = "+num*35+" type
= "+type);
                    return num*35;
                } else{return -sizeB;}
            case "ПГ":
                if(sizeC-num>=0){
                    sizeC-=num;
                    closedC+=num;
                    for(i=0;i<num;i++)
                        C.add("name = "+name+" price = "+num*25+" type
= "+type);
                    return num*25;
                } else{return -sizeC;}
            case "КЕ":
                if(scntr-num>=0){
                    scntr-=num;
                    closedcntr+=num;
                    for(i=0;i<num;i++)
                        cntr.add("name = "+name+" price = "+num*30+"
type = "+type);
                    return num*30;
                } else{return -scntr;}
            case "ПΘ":
                if(sside-num>=0){
                    sside-=num;
                    closedside+=num;
                    for(i=0;i<num;i++)
                        side.add("name = "+name+" price = "+num*35+"
type = "+type);
```

```
                    return num*20;
                } else{return -sside;}
        }
        return 1;
    }
```

Υλοποίηση της guest() (Ερώτημα 4ο)

Η παρακάτω συνάρτηση είναι τύπου ArrayList μέσα σε ArrayList. Ο λόγος που επιλέχθηκε αυτός ο τρόπος επιστροφής είναι για να επιστρέψει όλες τις λίστες που μπορεί να έχει κάνει κράτηση κάποιος χρήστης. Δημιουργείται ένα προσωρινό ArrayList<ArrayList<String>> για να εισάγουμε όλες τις λίστες σε αυτόν και μετά να τον κάνουμε return. Αν κάποια λίστα δεν είναι null τότε εισάγεται στην προσωρινή λίστα. Βέβαια έτσι όπως έχω διαμορφώσει την λίστα δεν θα είναι ποτέ null απλά μπορεί να έχει "" κενό χαρακτήρα. Τέλος, επιστρέφει την προσωρινή μεταβλητή.

```
public ArrayList<ArrayList<String>> guest(){
        ArrayList<ArrayList<String>> tmp=new ArrayList<>();
        if(A!=null)
            tmp.add(A);
        if(B!=null)
            tmp.add(B);
        if(C!=null)
            tmp.add(C);
        if(cntr!=null);
            tmp.add(cntr);
        if(side!=null)
            tmp.add(side);
        return tmp;
    }
```

Υλοποίηση της cancel (5ο ερώτημα)

Η cancel είναι η συνάρτηση, η οποία έχει πολλά κενά αλλά θα αναλυθεί ο τρόπος σκέψης. Αρχικοποιείται ένας μετρητής για να δειχθεί πόσες θέσεις έχει ο συγκεκριμένος χρήστης και το pos για την θέση. Μετά συγκρίνοντας τον τύπο που έχει δώσει ο χρήστης ψάχνει τον ανάλογο πίνακα και αν βρεθεί ο χρήστης αυξάνει τον μετρητή και την θέση στην οποία βρίσκεται. Αν το μέγεθος που θέλει να ακυρώσει ο χρήστης είναι μικρότερο από το μετρητή ο οποίος έχει το πόσες θέσεις έχει κλείσμενες τότε αυξάνει τις διαθέσιμες θέσεις που έχει η λίστα και μειώνει τις κλεισμένες θέσεις κατά num. Μετά, για τη θέση που βρέθηκε το στοιχείο μέχρι τη θέση συν το νούμερο των θέσεων που θέλει να ακυρώσει, κάνει remove από την λίστα τις παραπάνω θέσεις και επιστρέφει τον πίνακα A. Αλλιώς επιστρέφει «αποτυχία».

```
public ArrayList<String> cancel(String type,int num,String name){
        int count=0,pos;
        switch(type){
            case "ΠΑ":
                for (int i = 0; i < sizeA; i++)
                    if (A[i] == name){
                        pos=i;
                        count++; // μετράει πόσες φορές έχει κλείσει ο
χρήστης
                }
                if(count<=num){
                    sizeA+=num;
                    closedA-=num;
                    for(i=pos;i<pos+num;i++)
                        A.remove(i);
                    return A;
                }else return "αποτυχία";

            // case "ΠΒ":

        }
    }
```

THServer.java

Στον παρακάτω κώδικα γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών και μετά φτιάχνεται η class THServer. Υλοποιείται ο κατασκευαστής της και δημιουργείται ένα αντικείμενο τύπου THImpl, για να μπορεί να έχει πρόσβαση στις συναρτήσεις που δημιουργήσαμε παραπάνω, με όνομα in. Ακόμα, με το Naming.rebind() εισάγουμε την διεύθυνση, την πόρτα και το αντικείμενο. Αν δεν πετύχει, ο server εμφανίζει το είδος του λάθους στην οθόνη. Στην main καλείται ο κατασκευαστής της THServer. Η μεταβλητή reg τύπου Registry που μόλις δημιουργήθηκε δημιουργεί registry στην πόρτα "1099" αλλιώς εμφανίζει με exception τι είδους λάθος είναι.

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.*;
import java.rmi.*;

public class THServer{
    public THServer(){
        try{
            THImpl in= new THImpl();
            // Registry reg=LocateRegistry.createRegistry(1099);
            Naming.rebind("rmi://localhost:1099/TH",in);
        }catch (Exception e){
            e.printStackTrace(); //ektypwnei to eidos sfalmatos
        }
    }

    public static void main(String args[]) throws Exception{
        new THServer();
        try{
            //Δημιουργεί registry την πόρτα 1099
            Registry reg=LocateRegistry.createRegistry(1099);
        }catch(Exception e){
            e.printStackTrace();
        }

    }
}
```

THClient.java

Ο client καλεί όλες τις συναρτήσεις και επικοινωνεί με το rmi και τον server. Στον παρακάτω κώδικα φτιάχνουμε μία συνάρτηση η οποία διαβάζει με την scanner ένα string και αποθηκεύει μόνο τον πρώτο χαρακτήρα πχ από το "yes" αποθηκεύει το 'y'. Αυτός ο κώδικας θα χρειαστεί σε παρακάτω ερώτημα.

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.*;

public class THClient{
    public static char input() throws RemoteException{
        Scanner scan= new Scanner(System.in);
        System.out.println("give answer y:yes or n:no");
        char opt=scan.next().charAt(0);
        return opt;
    }
```

1ο Ερώτημα – allowed parameters

Δημιουργώντας την main του client ελέγχει αν έχουν δοθεί παράμετροι κατά την κλήση της client. Αν δεν έχουν δοθεί τότε εμφανίζονται στον τερματικό οι παρακάτω επιλογές που είναι επιτρεπτές και ολοκληρώνεται ο κώδικας.

```
if(args.length==0){
                System.out.println("\nYou can only use:\n");
                System.out.println("java THClient list <hostname>");
                System.out.println("java THClient book <hostname>
<type> <number> <name>");
                System.out.println("java THClient guests <hostname>");
                System.out.println("java THClient cancel <hostname>
<type> <number> <name>");
                return;
            }
```

2ο Ερώτημα - clist

Σε περίπτωση που έχουν δοθεί παράμετροι κατά την κλήση της THClient τότε δημιουργεί ένα αντικείμενο τύπου THInterface με lookup το "rmi://localhost:1099/TH". Ακόμα, μπαίνει σε μία switch αναλόγως με το πρώτο στοιχείο των ορισμάτων. Αν αυτό είναι "list" τότε ελέγχει αν είναι 2 τα ορίσματα και σε αυτή την περίπτωση εκτυπώνει στο τερματικό το clist για το αντικείμενο c.

```
THInterface c=(THInterface)Naming.lookup("rmi://Localhost:1099/TH");
            switch(args[0]){
                case "list":
                    if(args.length==2)
                       System.out.println(c.clist());
                    break;
```

3ο Ερώτημα- book

Αν είναι “book” το πρώτο όρισμα τότε ελέγχει αν έχουν δοθεί πέντε παράμετροι και σε αυτή τη περίπτωση δημιουργεί ένα opt τύπου char. Ανοίγει μία try ώστε να δούμε αν υπάρχει κάποιο exception. Ακριβώς από κάτω φτιάχνουμε δύο μεταβλητές ακέραιου τύπου και στο ένα, το οποίο είναι το num, αρχικοποιείται με τη 4η παράμετρο η οποία περιέχει τον αριθμό αλλά πρώτα την κάνει μετατροπή σε ακέραιο χρησιμοποιώντας την εντολή Integer.parseInt(). Ενώ, η μεταβλητή cost τύπου int καλεί την συνάρτηση book() την οποία φτιάξαμε στο THImpl και ορίσαμε στην THInterface. Αν το cost το οποίο επιστράφηκε είναι μικρότερο του μηδενός σημαίνει ότι δεν υπήρχαν αρκετές θέσεις και τότε εκτυπώνει στην οθόνη τα αντίστοιχα μηνύματα και του δίνει την επιλογή να διαλέξει με την χρήση της input() συνάρτησης που αναφέρθηκε παραπάνω και την τιμή της την επιστρέφει στην μεταβλητή opt. Αν η opt έχει την τιμή ‘y’ τότε σημαίνει ότι θέλει να κλείσει ο χρήστης όλες τις απομείναντες θέσεις και τότε καλεί ξανά την book με τα ίδια ορίσματα αλλά στο num αντί για τη παράμετρο που δίνει ο χρήστης βάζουμε το μέγεθος το οποίο έχει διαθέσιμο από την προηγούμενη κλήση της συνάρτησης ώστε να ξέρουμε το ακριβές μέγεθος των διαθέσιμων θέσεων. Η ιδιαιτερότητα εδώ βρίσκεται ότι η συνάρτηση έχει επιστρέψει αρνητικό αριθμό οπότε βάζοντας το μείον (“-“) μπροστά από το cost τότε καλείται σωστά και εμφανίζεται το κατάλληλο μήνυμα με το κόστος που έχει να πληρώσει ο πελάτης. Αλλιώς, επιστρέφει το αρχικό κόστος.

```
case "book":
      if(args.length==5){
           char opt;
           try{
                int num=Integer.parseInt(args[3]);
                int cost=c.book(args[2],num,args[4]);
                if(cost<0){
                      System.out.println("Δεν υπάρχουν διαθέσιμα
εισητήρια όσα ζητάτε");
                      System.out.println("θες να πάρεις όσα υπάρχουν
σε αυτή τη θέση?");
                      opt=input();
                      if(opt=='y')
                           cost=c.book(args[2],-cost,args[4]);
                      System.out.println("cost is = "+cost);
                }else{
                      System.out.println(cost);
```

```
                }
            }catch(Exception e){
                    e.printStackTrace();
            }
      }
      break;
```

4ο Ερώτημα – guests

Στην περίπτωση που είναι η πρώτη παράμετρος “guests” και οι παράμετροι είναι δύο τότε φτιάχνει έναν ArrayList μέσα σε ArrayList και αρχικοποιείται με τη συνάρτηση guest() και μετά εμφανίζει όλα τα περιεχόμενα δημιουργώντας ακόμα ένα ArrayList.

```
case "guests":
     if(args.length==2){
        ArrayList<ArrayList<String>> Ar=c.guest();
        for(int i=0;i<Ar.size();i++){
            ArrayList<String> esAr=Ar.get(i);
            for(int j=0;j<esAr.size();j++)
               System.out.println(esAr.get(j));
         }
      }
      break;
```

5ο Ερώτημα – Cancel

Στην περίπτωση της cancel μετατρέπει το 4ο όρισμα ακέραιο και εκτυπώνει τα αποτελέσματα της cancel.

```
Στην περίπτωση της cancel μετατρέπει το 4ο όρισμα ακέραιο και
εκτυπώνει τα αποτελέσματα της cancel.

case "cancel":
      int num=Integer.parseInt(args[3]);
      System.out.println(c.cancel(args[3],num,args[4]));
      break;
      }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Ενδεικτικές εκτελέσεις του κώδικα

Όλα τα τρεξίματα θα γίνουν δίχως την χρήση της cancel διότι εμφανίζει errors

Compile όλων των αρχείων:

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ javac *.java
```

Ενεργοποίηση του RMIRegistry:

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ rmiregistry 1099
```

Ενεργοποίηση του THServer:

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THServer
```

Κατά όλη την διάρκεια την εκτέλεσης του κώδικα στον client θα πρέπει ο THServer και ο Rmiregistry να παραμείνουν ενεργοποιημένοι άμα κλείσει ο server ή ο rmiregistry θα δημιουργήσει θέμα στα δεδομένα του θεάτρου.

1η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient

You can only use:

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

2η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient list localhost
100 θέσεις Πλατεία - Ζώνη Α (κωδικός: ΠΑ) - τιμή: 45 Ευρώ
200 θέσεις Πλατεία - Ζώνη Β (κωδικός: ΠΒ) - τιμή: 35 Ευρώ
400 θέσεις Πλατεία - Ζώνη Γ (κωδικός: ΠΓ) - τιμή: 25 Ευρώ
225 θέσεις Κεντρικός Εξώστης (κωδικός: ΚΕ) - τιμή: 30 Ευρώ
75 θέσεις Πλαϊνά θεωρεία (κωδικός: ΠΘ) - τιμή: 20 Ευρώ
```

3 Περίπτωση

3.1η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient book localhost ΠΒ 3 spiros
105
```

Κόστος = 105Ε

3.2η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient book localhost ΠΒ 300 spiros
Δεν υπάρχουν διαθέσιμα εισητήρια όσα ζητάτε
θες να πάρεις όσα υπάρχουν σε αυτή τη θέση?
give answer y:yes or n:no
y
cost is = 6895
```

3.3η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient book localhost ΠΑ 300 spiros
Δεν υπάρχουν διαθέσιμα εισητήρια όσα ζητάτε
θες να πάρεις όσα υπάρχουν σε αυτή τη θέση?
give answer y:yes or n:no
n
cost is = -100
```

(εμφανίζει -100 Ε αλλά δεν είχα χρονο να το διορθώσω. Είναι μια else στο opt στον client)

Οι κλεισμένες θέσεις :

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient list localhost
100 θέσεις Πλατεία - Ζώνη Α (κωδικός: ΠΑ) - τιμή: 45 Ευρώ
0 θέσεις Πλατεία - Ζώνη Β (κωδικός: ΠΒ) - τιμή: 35 Ευρώ
400 θέσεις Πλατεία - Ζώνη Γ (κωδικός: ΠΓ) - τιμή: 25 Ευρώ
225 θέσεις Κεντρικός Εξώστης (κωδικός: ΚΕ) - τιμή: 30 Ευρώ
75 θέσεις Πλαϊνά Θεωρεία (κωδικός: ΠΘ) - τιμή: 20 Ευρώ
```

4η περίπτωση

```
spirossif@DESKTOP-DPG2SGM:/mnt/d/OneDrive - University of West Attica/σχολη ΠΑΔΑ/Εξάμηνα/ΣΤ εξάμηνο/KATANEMHMENA/erg2$ java THClient guests localhost
name = spiros price = 105 type = ΠΒ
name = spiros price = 105 type = ΠΒ
name = spiros price = 105 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
name = spiros price = 6895 type = ΠΒ
```