Project Java 2020-2021

Έτος Σπουδών: 1ο Έτος

Συνεργάτης 1:
Γρημάνης Δαμιανός (ΑΜ: 1084571, dgrimanis@upnet.gr)

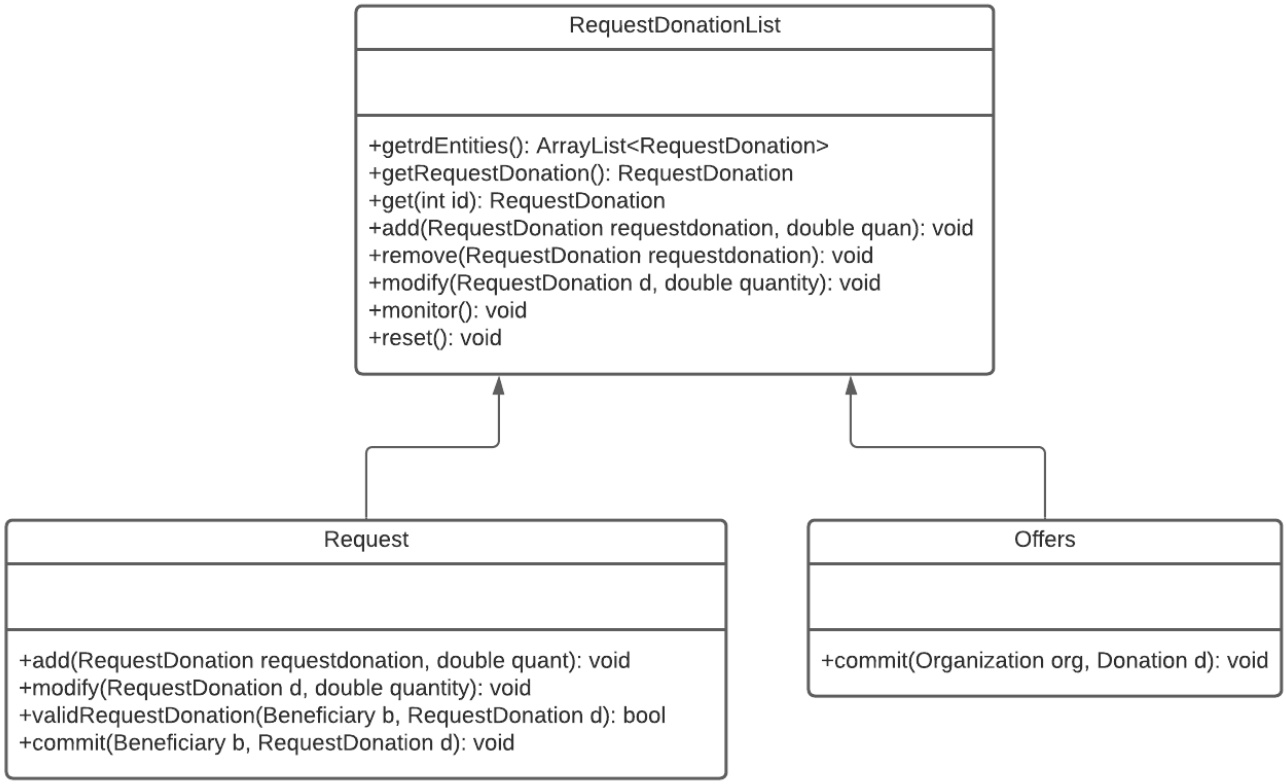
Συνεργάτης 2:
Κουρή Μαρία (ΑΜ: 1084526, up1084526@upnet.gr)

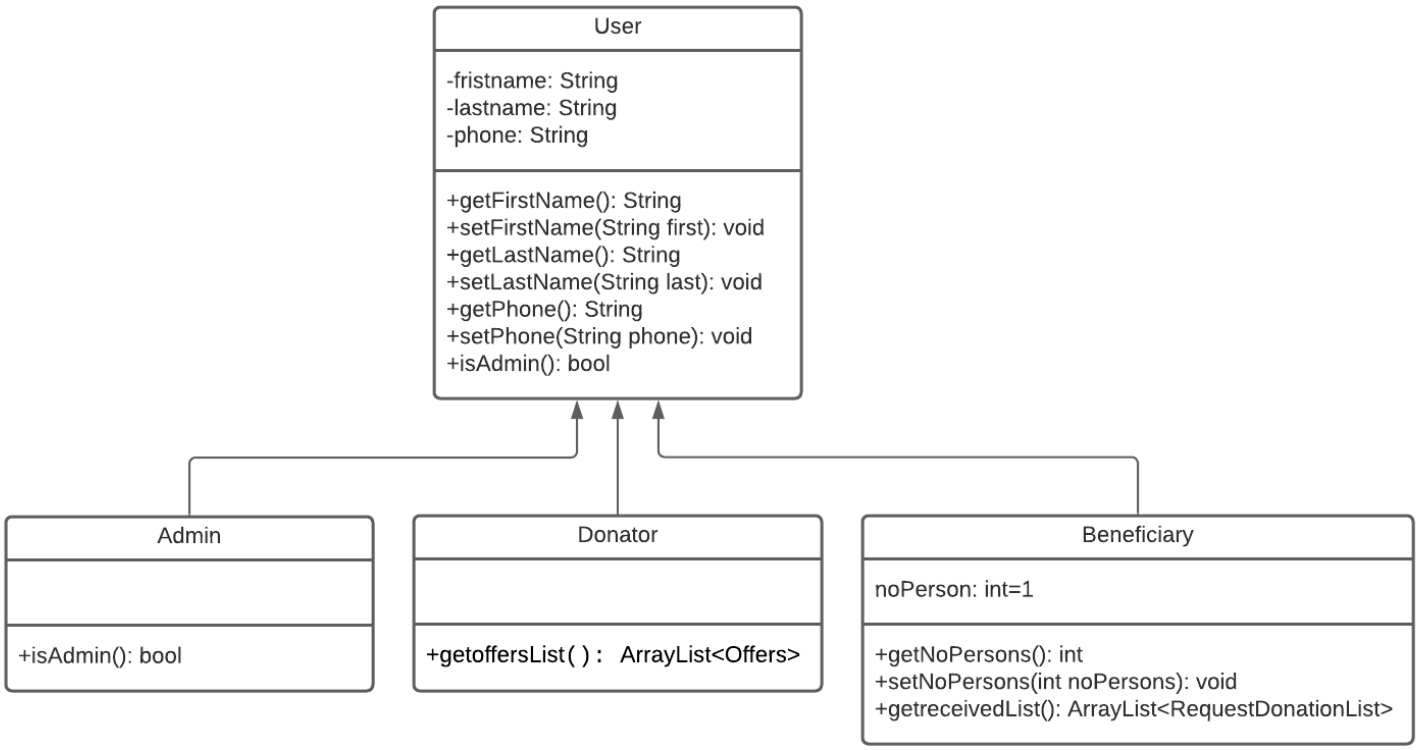
Συνεργάτης 3:
Πατέλη Χρυσαυγή (ΑΜ: 1084513, up1084513@upnet.gr)

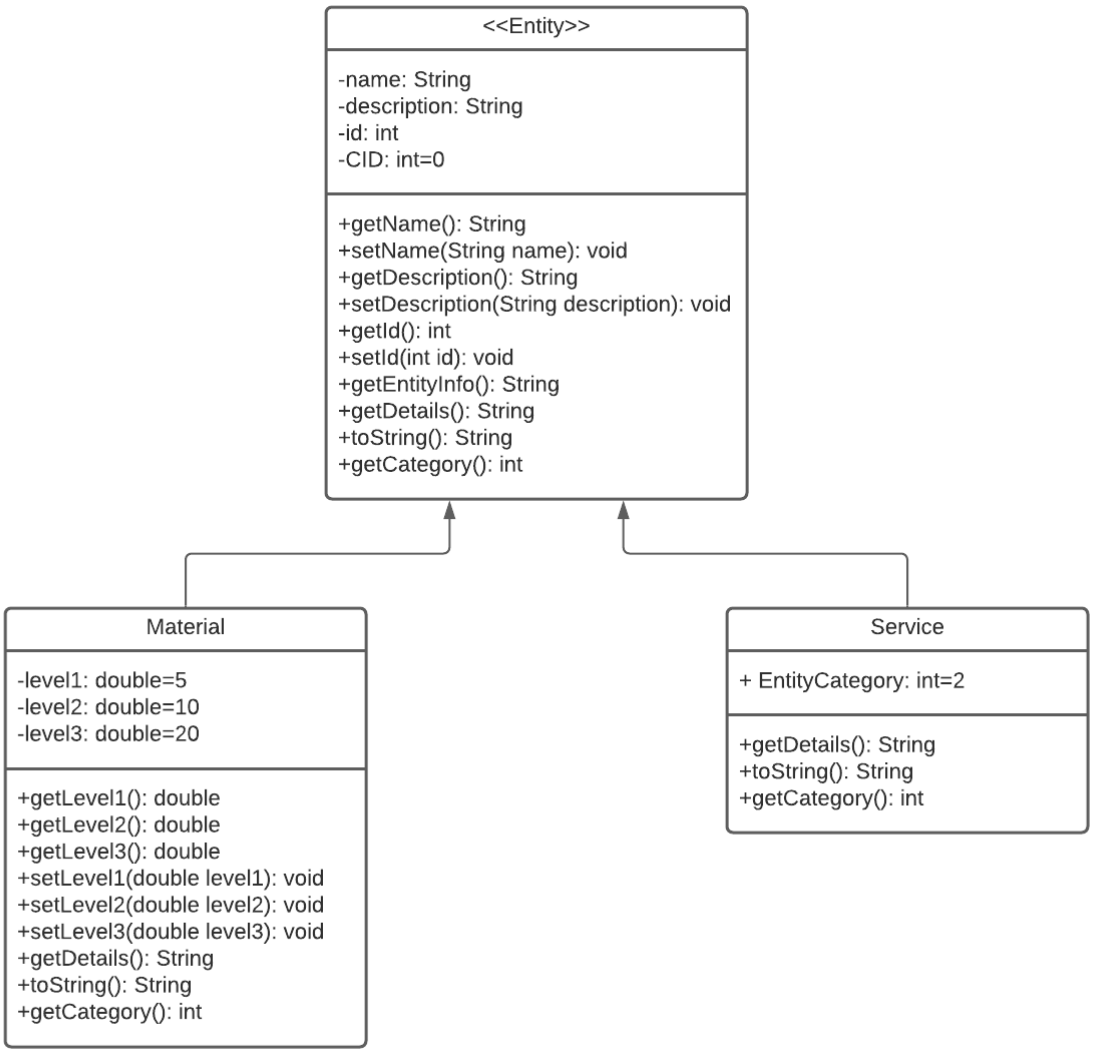
GitHub link: https://github.com/damianosgrim/Koinofeles_Idrima

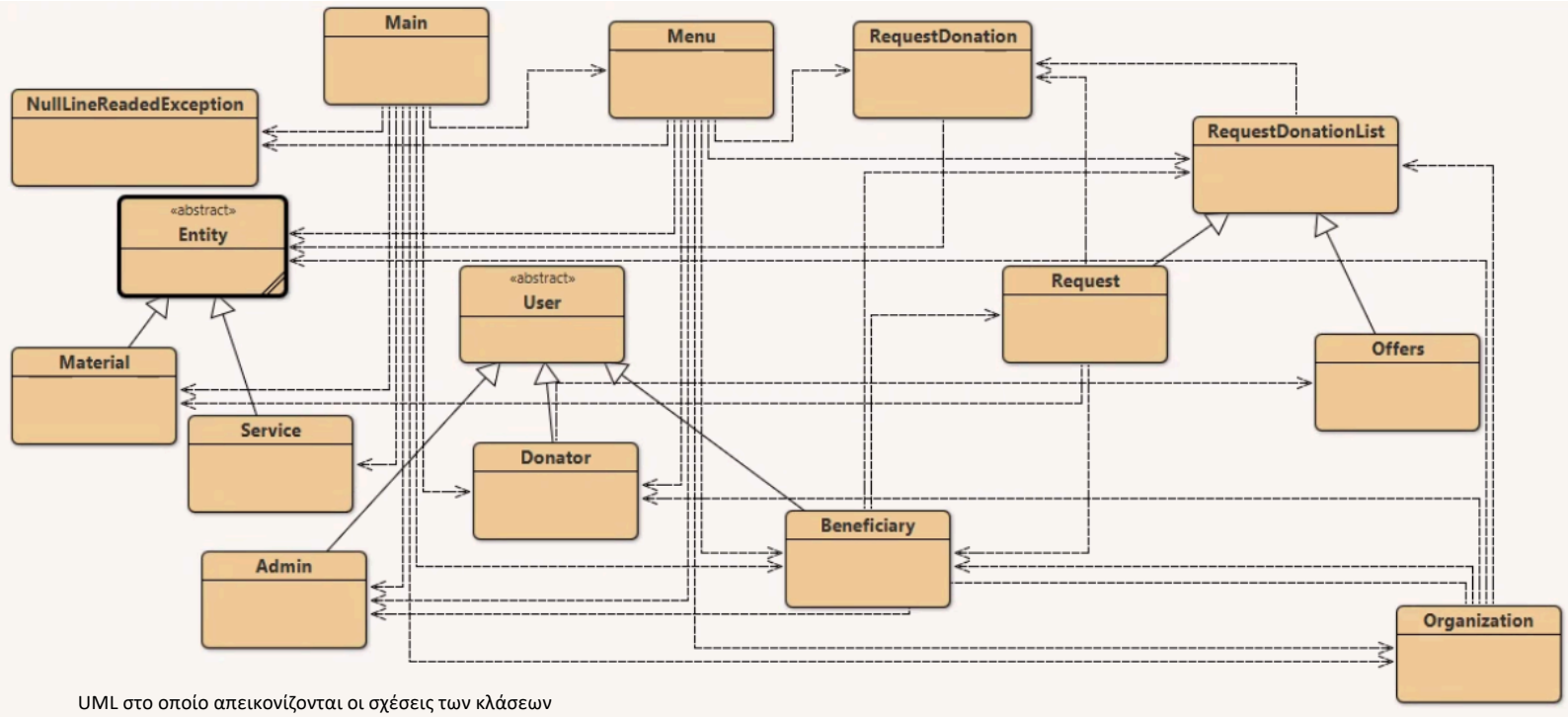
BlueJ Java

UML ΔΙΑΓΡΑΜΜΑ


RequestDonationList
+getrdEntities(): ArrayList<RequestDonation>
+getRequestDonation(): RequestDonation
+get(int id): RequestDonation
+add(RequestDonation requestdonation, double quan): void
+remove(RequestDonation requestdonation): void
+modify(RequestDonation d, double quantity): void
+monitor(): void
+reset(): void
Request
+add(RequestDonation requestdonation, double quant): void
+modify(RequestDonation d, double quantity): void
+validRequestDonation(Beneficiary b, RequestDonation d): bool
+commit(Beneficiary b, RequestDonation d): void
Offers
+commit(Organization org, Donation d): void


User
-fristname: String
-lastname: String
-phone: String
+getFirstName(): String
+setFirstName(String first): void
+getLastName(): String
+setLastName(String last): void
+getPhone(): String
+setPhone(String phone): void
+isAdmin(): bool
Admin
+isAdmin(): bool
Donator
+getoffersList( ) : ArrayList<Offers>
Beneficiary
noPerson: int=1
+getNoPersons(): int
+setNoPersons(int noPersons): void
+getreceivedList(): ArrayList<RequestDonationList>





UML στο οποίο απεικονίζονται οι σχέσεις των κλάσεων

Επεξήγηση του τρόπου υλοποίησης του Project

- Κατά την υλοποίηση του Project τηρούμε την αρχή απόκρυψης πληροφοριών και ενθυλάκωσης. Για αυτόν τον λόγο, οι μεταβλητές δηλώνονται private, οπότε για την ανάθεση τιμών σε αυτές χρησιμοποιούμε setters και getters για την πρόσβαση σε αυτές.
- Ακόμη, όλες οι μέθοδοι δηλώνονται ως public, ώστε να χρησιμοποιούνται σε άλλες κλάσεις, δηλαδή ο κώδικας να είναι επαναχρησιμοποιήσιμος.
- Ταυτόχρονα, σε μερικές υποκλάσεις οι μέθοδοι υπερκαλύπτουν αυτές τον υπερκλάσεων, είτε για προσθήκη παραπάνω πληροφοριών, είτε για αναπροσαρμογή του κώδικα, ώστε να επιτυγχάνεται ο τελικός στόχος.
- Επιπλέον, ορισμένες μέθοδοι δηλώνονται static για να μην καλούνται μέσω κάποιου αντικειμένου, αλλά μόνο με το όνομα της κλάσης, στην οποία ανήκουν και να γίνεται πιο απλή η διαδικασία προγραμματισμού.
- Τέλος, σε αρκετές κλάσεις προστίθενται constructors χωρίς ορίσματα, για να είναι δυνατή η δημιουργία αντικειμένων χωρίς να χρειάζεται ανάθεση τιμής σε κάθε όρισμα (όπως στους constructors με ορίσματα)

Κλάση User

Στην υλοποίηση της κλάσης προτιμήθηκε η χρήση first name και last name, αντί για name. Η κλάση αυτή δεν έχει στιγμιότυπα και για αυτό δηλώνεται ως abstract. Ακόμη έχουμε προσθέσει μία μέθοδο [isAdmin()] που επιστρέφει false για όλες τις υποκλάσεις εκτός από την υποκλάση Admin που έχουμε κάνει override, με αποτέλεσμα να επιστρέφει true.

Κλάση Admin

Είναι υποκλάση της User και έχει μία μέθοδο isAdmin(), η οποία κάνει override στην αντίστοιχη μέθοδο της υπερκλάσης. Ακόμη αποτελείται από έναν constructor με ορίσματα (String first, String last, String ph) για την δημιουργία στιγμιότυπων με αυτά τα χαρακτηριστικά.

Κλάση Donator

Είναι υποκλάση της User. Η κλάση αυτή αποτελείται από μία ArrayList τύπου Offers (λίστα δυναμικού μεγέθους- για την δημιουργία της κάνουμε import την βιβλιοθήκη java.util.*), στην οποία καταχωρούνται οι δωρεές που κάνει ο Donator και έναν getter για την λίστα ώστε να μπορούμε να καλούμε τα δεδομένα της και σε άλλες κλάσεις. Επιπλέον, υπάρχουν δύο constructors για την δημιουργία στιγμιότυπων. Ο πρώτος είναι χωρίς ορίσματα, έτσι ώστε το στιγμιότυπο να δημιουργείται χωρίς να καταχωρούνται πληροφορίες, και ο δεύτερος έχει ορίσματα String first, String last, String ph.

Κλάση Beneficiary

Η κλάση Beneficiary είναι υποκλάση της User και αναπαριστά τον επωφελούμενο του συστήματος. Διαθέτει μία μεταβλητή noPersons τύπου int, η οποία αρχικοποιείται με την τιμή 1, και αναπαριστά τον αριθμό των ατόμων της οικογένειας του Beneficiary. Ακόμη διαθέτει δύο constructors, έναν με ορίσματα (String firstname, String lastname, String phone, int noPersons) και έναν χωρίς (ώστε το στιγμιότυπο να δημιουργείται χωρίς να καταχωρούνται πληροφορίες. Επιπλέον υπάρχουν setters και getters για την τιμή noPersons για να τροποποιείται και να χρησιμοποιείται αντίστοιχα και στις άλλες κλάσεις. Τέλος δημιουργούμε δύο ArrayList (η μία τύπου RequestDonationList και η άλλη τύπου Request) που αναπαριστούν τις παροχές που έχει λάβει ο Beneficiary (receivedList) και αυτές που επιθυμεί να λάβει (requestsList) και έναν getter για την receivedList.

Κλάση Entity

Είναι κλάση η οποία αντιπροσωπεύει το είδος της δωρεάς. Αποτελείται από 3 private μεταβλητές, μία τύπου int id, δύο τύπου String name, String description και μία βοηθητική μεταβλητή για την ανάθεση id, η οποία δηλώνεται private και είναι τύπου int CID=0 (Η CID χρησιμοποιείται στους setters-getters του id, ώστε να τροποποιείται-αυξάνεται για κάθε παροχή). Ταυτόχρονα έχουμε setters και getters για τις μεταβλητές Name, Description και id για να τροποποιούμε τις private μεταβλητές και σε άλλες κλάσης (αρχή της ενθυλάκωσης).

Η κλάση διαθέτει και άλλες 3 μεθόδους:

i. public abstract String getDetails(): αφηρημένη μέθοδος -χωρίς σώμα- για χρήση της σε επόμενο στάδιο του κώδικα
ii. public String toString(): μέθοδος που εκτυπώνει τις πληροφορίες για το κάθε Entity
iii. public abstract int getCategory(): αφηρημένη μέθοδος -χωρίς σώμα. Δημιουργείται με δική μας πρωτοβουλία έτσι ώστε να χρησιμοποιηθεί στην main και να εκτυπώνει στον Admin του κάθε οργανισμού την ποσότητα της κάθε παροχής.

Κλάση Material

Η κλάση είναι υποκλάση της Entity και αναπαριστά όλα τα υλικά αγαθά του συστήματος. Διαθέτει 3 πεδία private double level1, level2 και level3, τα οποία αρχικοποιούνται με τις τιμές 5,10 και 20 αντίστοιχα και αναφέρονται στην ποσότητα που δικαιούνται 1, 2-4, 5+ άτομα. Επίσης, διαθέτει μια int EntityCategory=1 μεταβλητή, η οποία είναι public και αρχικοποιημένη, διότι χρησιμοποιείται για να εμφανίζει στην περιγραφή κάθε παροχής αν είναι Material. Ακόμη προσθέτουμε setters και getters για τα level1,2,3 και getters:

i. public String getDetails(): μέθοδος που δημιουργείται για να επιστρέφει επιπλέον πληροφορίες για το Material (levels και ότι το αντικείμενο είναι Material)
ii. public String toString(): εκτυπώνει τις πληροφορίες και τις λεπτομέρειες των Materials και υπερκαλύπτει την αντίστοιχη μέθοδο της υπερκλάσης.

iii. public int getCategory(): δημιουργείται με δική μας πρωτοβουλία έτσι ώστε να χρησιμοποιηθεί στην main και να εκτυπώνει στον Admin του κάθε οργανισμού την ποσότητα των Materials.

Κλάση Service

Είναι υποκλάση της Entity Class, όπως και η Material, και αναπαριστά το σύνολο των υπηρεσιών που διαθέτει ο οργανισμός. Διαθέτει ένα πεδίο intEntityCategory=2, το οποίοι είναι public και αρχικοποιημένο, διότι χρησιμοποιείται για να εμφανίζει στην περιγραφή κάθε παροχής αν είναι Service (αντίστοιχη λειτουργία με την κλάση Material). Ακόμη διαθέτει τις ακόλουθες μεθόδους:

i. public String getDetails(): μέθοδος που επιστρέφει το είδος του της δωρεάς-παροχής
ii. public String toString(): εκτυπώνει τις πληροφορίες και τις λεπτομέρειες των Services και υπερκαλύπτει την αντίστοιχη μέθοδο της υπερκλάσης.
iii. public int getCategory(): δημιουργείται με δική μας πρωτοβουλία έτσι ώστε να χρησιμοποιηθεί στην main και να εκτυπώνει στον Admin του κάθε οργανισμού την ποσότητα των Services.

Κλάση RequestDonation

Η κλάση αναπαριστά το αίτημα ή την παροχή ενός συγκεκριμένου Entity σε μία συγκεκριμένη ποσότητα. Αποτελείται από δύο πεδία Entity entity και double quantity. Ακόμη διαθέτει έναν constructor με ορίσματα Entity entity, double quantity, getters για το Entity και το Quantity, setter για το Quantity και μία μέθοδο get_id(), η οποία επιστρέφει το id του entity για ένα requestdonation. Τέλος διαθέτει έναν interface comparator, ο οποίος συγκρίνει δύο αντικείμενα με βάση το ID τους και εφόσον είναι ίδια επιστρέφει 0, αλλιώς 1.

Κλάση Organization

Η κλάση αυτή απεικονίζει τον οργανισμό που υποστηρίζει το σύστημα. Διαθέτει πεδία String name, Admin admin και μία ArrayLists (λίστες με δυναμικό μέγεθος). Οι λίστες αυτές είναι οι:

i. entityList τύπου Entity: λίστα με τα είδη που μπορούν να διανεμηθούν σε Beneficiaries
ii. donatorList τύπου Donator: λίστα με τους δωρητές
iii. beneficiaryList τύπου Beneficiary: λίστα με τους επωφελούμενους
iv. και συνοδεύονται από τους αντίστοιχους getters. Στη συνέχεια αποτελείται από έναν constructor με ορίσματα και έναν χωρίς ορίσματα, δημιουργείται ένα αντικείμενο currentDonations τύπου RequestDonationList με τους αντίστοιχους getters-setters και προστίθενται getters-setters για τον admin του οργανισμού και getter για το όνομα του οργανισμού. Ταυτόχρονα έχει μέθοδους:
v. addEntity(Entity entity): ελέγχει αν υπάρχει ήδη (εγείρει την κατάλληλη εξαίρεση) το entity στον οργανισμό και αν δεν υπάρχει το προσθέτει
vi. removeEntity(int entity): διαγράφει ένα entity

vii. insertDonator(Donator donator): προσθέτει έναν donator στην λίστα και εγείρει την κατάλληλη εξαίρεση εάν αυτός υπάρχει ήδη και εμφανίζει το αντίστοιχο μήνυμα
viii. removeDonator(Donator donator): αφαιρεί έναν donator από τον οργανισμό
ix. insertBeneficiary(Beneficiary beneficiary): προσθέτει έναν beneficiary στην λίστα και εγείρει την κατάλληλη εξαίρεση εάν αυτός υπάρχει ήδη και εμφανίζει το αντίστοιχο μήνυμα
x. removeBeneficiary(Beneficiary beneficiary): αφαιρεί έναν Beneficiary απο τον οργανισμό
xi. listEntities(): αναπαριστά τις υπάρχουσες κατηγορίες των entity και λίστα με όλα τα entity ανά συγκεκριμένη κατηγορία
xii. listDonators(): εμφανίζει τους donators αριθμημένους
xiii. listBeneficiary(): εμφανίζει τους beneficiary αριθμημένους
xiv. showCategory(String Category): μέθοδος που προστίτεια με δική μας πρωτοβουλία για τον προσδιορισμός του entity (service ή material) για την εκτύπωσή του στο Menu. Αν το Category είναι 1 τότε εμφανίζονται αριθμημένα τα Materials και αν είναι 2 εμφανίζονται τα Services, αλλιώε εκτυπώνεται το κατάλληλο μήνυμα. Για να υλοποιηθεί αυτό το στάδιο χρησιμοποιούμε switch statements με cases 1 και 2 για τις αντίστοιχες λειτουργίες (1-Materials, 2-Services)

Κλάση RequestDonationList

Είναι μία κλάση που αναπαριστά την συλλογή των αντικειμένων RequestDonation. Αποτελείται από μία ArrayList rdEntities τύπου RequestDonation και ένα αντικείμενο rdon τύπου RequestDonation. Προσθέτουμε getter για την διαχείριση της ArrayList, constructor χωρίς ορίσματα και getter τύπου RequestDonation που επιστρέφει το rdon. Στην συνέχεια ακολουθούν οι μέθοδοι:

i. get(int id): για την καταχώρηση ενός id του entity και επιστροφή του αντίστοιχου requestdonation
ii. add(RequestDonation requestdonation, double quant): χρησιμοποιείται για την προσθήκη requestdonation στην rdentities και την τροποποίση του quantity εάν αυτό υπάρχει ήδη
iii. remove(RequestDonation requestdonation): διαγράφει ένα requestdonation από την λίστα
iv. modify(RequestDonation d ,double quantity): μέθοδος για την τροποποίηση της ποσότητας ενός entity
v. monitor(): εκτυπώνει το περιεχόμενο της λίστας
vi. reset(): εκκαθάριση της λίστας

Κλάση Request

Είναι κλάση η οποί αναπαριστά το σύνολο των παροχών που ζητάει ο Beneficiary και είναι υποκλάση της RequestDonationList. Αποτελείται από τις ακόλουθες μεθόδους:

i. add(RequestDonation requestdonation, double quant): ελέγχει εάν ένα entity είναι διαθέσιμο στον οργανισμό και υπερκαλύπτει την αντίστοιχη μέθοδο της

υπερκλάσης. Εάν το entity είναι ήδη διαθέσιμο στον οργανισμό ανανεώνει την υπάρχουσα κατάσταση, αλλιώς δημιουργεί καινούργιο entity (if-else)

ii. modify(RequestDonation d ,double quantity): η μέθοδος αυτή ελέγχει εάν ο beneficiary δικαιούται την ποσότητα που ζητάει

iii. boolean validRequestDonation(Beneficiary b, RequestDonation d): ελέγχει την ποσότητα ενός RequestDonation και αν είναι επιτρεπτεί για τω αριθμό των μελών που έχει δηλώσει ο beneficiary. Εάν μετά τον έλεγχο υπάρχουν οι ζητούμενες ποσότητες διαθέσιμες επιστρέφει true.

iv. commit(Beneficiary b, RequestDonation d): η μέθοδος commit δεν έχει ολοκληρωθεί και υπάρχει σε σχόλια, καθώς υπήρχε θέμα στο compile με το RequestDonation d

Κλάση Offers

Κλάση η όποια είναι υποκλάση της RequestDonationList και αναπαριστά το σύνολο των παροχών που θέλει να προσφέρει ένας donator στο σύστημα. Αποτελείται από μία μέθοδο commit() η οποία ενημερώνει τα CurrentDonation του οργανισμού με το περιεχόμενο της λίστας rdEntities και στην συνέχεια διαγράφει το περιεχόμενο της rdEntities. Για την εκτέλεση αυτής της λειτουργίας δοκιμάσαμε αντιγραφή της λίστας με χρήση της εντολής copy και δύο ελέγχους με την χρήση for για προσθήκη του κάθε entity στα CurrentDonation, αλλά δεν καταφέραμε να εκτελέσουμε τον κώδικα χωρίς error.

Κλάση main

Στην main κλάση δημιουργούμε τα ακόλουθα αντικείμενα:

i. **Αντικείμενο τύπου Organization**
   Organization org = new Organization("KoinofelesIdrimaCEID", ad);

ii. **Αντικείμενο τύπου Menu**
   Menu menu = new Menu();

iii. **Αντικείμενο τύπου Admin**
   Admin ad = new Admin("Chryssa", "Pateli", "13033");

iv. **Αντικείμενο τύπου Material**
   Material mlk = new Material();
   Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό
   mlk.setName("Milk");
   mlk.setDescription("Dairy product 1lt ");
   mlk.getId();
   org.addEntity(mlk);

v. **Αντικείμενο τύπου Material**
   Material sug = new Material();
   Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό
   sug.setName("Sugar");

sug.setDescription("Sweetener 1kg ");
sug.getId();
org.addEntity(sug);

vi. **Αντικείμενο τύπου Material**

Material rc = new Material();

Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό

rc.setName("Rice");
rc.setDescription("Cereal grain 500gr ");
rc.getId();
org.addEntity(rc);

vii. **Αντικείμενο τύπου Service**

Service mds = new Service();

Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό

mds.setName("MedicalSupport");
mds.setDescription("Health service");
mds.getId();
org.addEntity(mds);

viii. **Αντικείμενο τύπου Service**

Service nrs = new Service();

Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό

nrs.setName("NurserySupport");
nrs.setDescription("Health service");
nrs.getId();
org.addEntity(nrs);

ix. **Αντικείμενο τύπου Service**

Service bs = new Service();

Και κλήση setter για το Name, Description και getter για το Id για καταχώρηση για το συγκεκριμένο αντικείμενο και τέλος προσθήκη του στον Οργανισμό

bs.setName("BabySitting");
bs.setDescription("Housekeeping ");
bs.getId();
org.addEntity(bs);

x. **Αντικείμενα τύπου Beneficiary**

Beneficiary b1 = new Beneficiary("Maria", "Kouri", "13032", 4);
Beneficiary b2 = new Beneficiary("Damianos", "Grimanis", "13031", 2);

xi. **Αντικείμενα τύπου Donator**

Donator d = new Donator("Eleni", "Vogiatzaki", "13030");

Τέλος προσθέτουμε τους Beneficiaries και Donator στον Οργανισμό προσθήκη στον οργανισμό

```
org.insertBeneficiary(b1);
org.insertBeneficiary(b2);
org.insertDonator(d);
```

και την ακόλουθη εξαίρεση:

```
try
{
   menu.Scan(); //καλεί την scan της menu
}
catch (NullLineReadedException nlre) //εξαίρεση για κενή γραμμή
{
   nlre.printStackTrace();
}
```

Κλάση Menu

Στην κλάση Menu εισάγουμε τις βιβλιοθήκες java.util.*, java.io.* και java.lang.* για την χρήση της εντολής ParseInt, ώστε να μετατρέψουμε string σε int. Διαθέτει έναν Scanner για τις απαντήσεις του χρήστη, δύο πεδία τύπου string ans και phone και μία Boolean μεταβλητή havelogout, η οποία γίνεται true by default ή όταν κάποιος χρήστης κάνει logout και false όταν εμφανιστεί μήνυμα χαιρετισμού. Στη συνέχεια υπάρχουν οι εξής μέθοδοι:

i. Scan(): Εκκινεί το πρόγραμμα και καλωσορίζει τον χρήστη. Ελέγχει εάν το τηλέφωνο που εισάγει ο χρήστης ανήκει σε κάποιον Beneficiary, Donator ή στον Admin και αν δεν είναι καταχωρημένο δίνει την επιλογή εγγραφής στο σύστημα.
ii. SignB(): Δημιουργείται ένα καινούργιου αντικείμενο τύπου Beneficiary και ζητάει από τον χρήστη να εισάγει τα στοιχεία του.
iii. SignD(): Δημιουργείται ένα καινούργιου αντικείμενο τύπου Donator και ζητάει από τον χρήστη να εισάγει τα στοιχεία του.
iv. MenuB(Beneficiary b): Εμφανίζει το menu πλοήγησης του Beneficiary
v. MenuD(Donator d): Εμφανίζει το menu πλοήγησης του Donator
vi. MenuA(Admin a): Εμφανίζει το menu πλοήγησης του Admin
vii. ListBen(): Μέθοδος υπό-menu για τον Admin ώστε να μπορεί διαχειριστεί τους εγγεγραμμένους Beneficiary
viii. ListDon(): Μέθοδος υπό-menu για τον Admin ώστε να μπορεί διαχειριστεί τους εγγεγραμμένους Donator
ix. LogOut(): Μέθοδος για την αποσύνδεση χρήστη και την επιστροφή στην αρχική σελίδα
x. ExitFromProgram(): Μέθοδος που τερματίζει το πρόγραμμα

ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Κλάση Admin

```
public class Admin extends User {

   //constuctor με ορίσματα όνομα, επώνυμο, τηλέφωνο
   public Admin (String first, String last, String ph)
   { super(first, last, ph); }

   public boolean isAdmin() { return true; }//Override
}
```

Κλάση Beneficiary

```
import java.util.*;
public class Beneficiary extends User {
   int noPersons=1; //αριθμός ατόμων της οικογένειας

   //constructor χωρίς ορίσματα
   public Beneficiary(){}

   //constructor με ορίσματα όνομα, επώνυμο, τηλέφωνο και αριθμό ατόμων
   public Beneficiary(String firstname, String lastname, String phone, int noPersons)
   { super(firstname, lastname, phone);
     this.noPersons=noPersons; }

   //getter-setter για τον αριθμό ατόμων
   public int getNoPersons() { return noPersons; }
   public void setNoPersons(int noPersons) {this.noPersons=noPersons;}

   //λίστα με τις παροχές που έχει λάβει ο Beneficiary
   static ArrayList<RequestDonationList> receivedList = new
ArrayList<RequestDonationList>();
    public static ArrayList<RequestDonationList> getreceivedList(){
      return receivedList;
   }
   //λίστα με τις παροχές που επιθυμεί να λάβει ο Beneficiary
   ArrayList<Request> requestsList = new ArrayList<Request>();
}
```

Κλάση Donator

```
import java.util.*;
public class Donator extends User {

   //δημιουργία λίστας για την καταχώρηση όσων επιθυμεί ο Donator
   static ArrayList<Offers> offersList = new ArrayList<Offers>();

    public static ArrayList<Offers> getoffersList(){
      return  offersList;
   }
   //constructor χωρίς ορίσματα
   public Donator(){}
```

```
   //constructor με ορίσματα όνομα, επώνυμο, τηλέφωνο
   public Donator(String first, String last, String ph)
   { super(first, last, ph); }
}
```

Κλάση Entity

```
import java.util.*;
public abstract class Entity{
   private String name; //ονομα δωρεάς
   private String description; //συντομη περγραφή
   private int id; //κωδικός είδους
    private static int CID = 0; // βοηθητικη μεταβλητη για αναθεση id

   //getter-setter για όνομα δωρεάς
   public String getName(){return name;}
   public void setName(String name){this.name=name;}

   //getter-setter για περιγραφή δωρεάς
   public String getDescription(){return description;}
   public void setDescription(String description){this.description=description;}

   //getter-setter για ID δωρεάς
   public int getId(){
       ++CID;
      id = CID;
      return id;
   }
   public void setId(int id){this.id=id;}

   //επιστροφή πληροφοριών
   public String getEntityInfo(){
      String cat="" ; // μεταβλητη για διαχωρισμο Id εμφανιζει M:material. S:service
      if(getCategory()==1){ cat="M" ;}
      else if(getCategory()==2){ cat="S";}
      return "Name of donation: " +name + " and is " + description + " ID:  " + cat  + id;
   }

   //αφηρημένη μέθοδος
   public abstract String getDetails();

   //εκτύπωση πληροφοριών
   public String toString(){
      return "info: " + getEntityInfo() + "details "+ getDetails();
   }

   //μέθοδος που δηλώνεται για να χρησιμοποιηθεί στην main ώστε να εκτυπώνει στον Admin την ποσότητα των Materials
   public abstract int getCategory();
}
```

Κλάση Main

```
import java.util.*;
public class Main {
public static void main(String[] args) {
 //δημιουργία αντικειμένου τύπου admin
Admin ad = new Admin("Chryssa", "Pateli", "13033");
//δημιουργία αντικειμένου τύπου Organization
Organization org = new Organization("KoinofelesIdrimaCEID", ad);

//δημιουργία αντικειμένου τύπου Menu
Menu menu = new Menu();

//δημιουργία αντικειμένου τύπου Material
Material mlk = new Material();
mlk.setName("Milk");
mlk.setDescription("Dairy product 1lt ");
mlk.getId();
org.addEntity(mlk);

//δημιουργία αντικειμένου τύπου Material
Material sug = new Material();
sug.setName("Sugar");
sug.setDescription("Sweetener 1kg ");
sug.getId();
org.addEntity(sug);

//δημιουργία αντικειμένου τύπου Material
Material rc = new Material();
rc.setName("Rice");
rc.setDescription("Cereal grain 500gr ");
rc.getId();
org.addEntity(rc);

//δημιουργία αντικειμένου τύπου Service
Service mds = new Service();
mds.setName("MedicalSupport");
mds.setDescription("Health service");
mds.getId();
org.addEntity(mds);

//δημιουργία αντικειμένου τύπου Service
Service nrs = new Service();
nrs.setName("NurserySupport");
nrs.setDescription("Health service");
nrs.getId();
org.addEntity(nrs);

//δημιουργία αντικειμένου τύπου Service
Service bs = new Service();
bs.setName("BabySitting");
bs.setDescription("Housekeeping ");
```

```
bs.getId();
org.addEntity(bs);

//δημιουργία αντικειμένου τύπου Beneficiary
Beneficiary b1 = new Beneficiary("Maria", "Kouri", "13032", 4);
Beneficiary b2 = new Beneficiary("Damianos", "Grimanis", "13031", 2);

//δημιουργία αντικειμένου τύπου Donator
Donator d = new Donator("Eleni", "Vogiatzaki", "13030");
//προσθήκη στον οργανισμό
org.insertBeneficiary(b1);
org.insertBeneficiary(b2);
org.insertDonator(d);


try
{
   menu.Scan(); //καλεί την scan της menu
}
catch (NullLineReadedException nlre) //εξαίρεση για κενή γραμμή
{
   nlre.printStackTrace();
}
}
}
```

Κλάση Material

```
import java.util.*;
public class Material extends Entity{

   private static double level1=5; //η ποσοτητα που δικαιουται 1 ατομο
   private static double level2=10; //η ποσοτητα που δικαιουνται 2-4 ατομα
   private static double level3=20; //η ποσοτητα που δικαιουνται 5+ ατομα
   public int EntityCategory=1;

   //getters για levels
   public static double  getLevel1(){return level1;}
   public static double getLevel2(){return level2;}
   public static double getLevel3(){return level3;}

   ////setters για levels
   public void setLevel1(double level1){this.level1=level1;}
   public void setLevel2(double level2){this.level2=level2;}
   public void setLevel3(double level3){this.level3=level3;}

   //getters για επιστροφή των ποσοτήτων του κάθε level και του είδους
   public String getDetails(){

      return ("level1: "+ level1 + " level2 " + level2 + " level3 " + level3 + " this object is
material");
```

```
    }


    //μέθοδος για την εκτύπωση πληροφοριών και λεπτομερειών //@override
    public String toString(){
        return "info: " + getEntityInfo() + "details "+ getDetails();
     }

    //μέθοδος που δηλώνεται για να χρησιμοποιηθεί στην main ώστε να εκτυπώνει στον Admin την ποσότητα των Materials
    public int getCategory(){
        return EntityCategory;
    }
}
```

Κλάση Menu

```
import java.util.*;
import java.io.*;
import java.lang.*; //Για την ParseInt μετατροπή string σε int
public class Menu {
    static Scanner scan = new Scanner(System.in);
    private static String ans;
    private static String phone;
    //μεταβλητή που γίνεται true by default ή όταν κάποιος χρήστης κάνει logout και false όταν εμφανιστεί μήνυμα χαιρετισμού
    private static boolean havelogout=true;


public static void Scan() throws NullLineReadedException{
    System.out.println("Welcome to " + Organization.getOrgName() + "!");
    System.out.println("Please enter a phone number: ");
    int search = 0; //Εκχωρηση τιμής 1,2,3 ανάλογα με τον τύπο του user
     Beneficiary b; //Βοηθητική μεταβλητή
     Donator d; //Βοηθητική μεταβλητή
     ans=scan.nextLine();
     phone=ans; //καταχώρηση της απάντησης σε μεταβλητή για διατήρηση της και να μην ξαναζητείται από τον χρήστη
      boolean exit=false; //μεταβλητή για αμυντικό προγραμματισμό

    /*μέθοδος η οποία ελέγχει αν το νούμερο που έχουμε βάλει ανήκει σε κάποιον Donator που είναι
    ήδη εγγεγραμμένος στο σύστημα*/
     for(int i = 0; i<Organization.getdonatorList().size(); i++){
         if(Organization.getdonatorList().get(i).getPhone().equals(ans)){

           search=1; //γίνεται 1 αν είναι don
           d=Organization.getdonatorList().get(i);
           System.out.println("The phone number " + ans + " has been already added in the system as donator");
           //αν το τηλέφωνο βρεθέι ρωτάει αν θέλει να συνδεθεί ως ο συγκεκριμένος donator
           do{
```

```
        System.out.println("Do you want to sign in as: " + d.getFirstName() + " " + d.getLastName() +"? (y/n)");
        ans=scan.nextLine();
        if ((ans.equals("n") || ans.equals("no") )) {exit=true; LogOut();} //αν η πει όχι επιστρέφει στο αρχικό menu
        else if(ans.equals("y") || ans.equals("yes") ) {
          exit=true;
         Menu.MenuD(d); //Καλέι την μέθοδο με όρισμα την βοηθητική μεταβλητή d τύπου Donator με βάση το τηλέφωνο που βρέθηκε
        }
       }while(!exit);
     }
     }

   //κάνει τον ίδιο έλεγχο αλλά για έναν Beneficiary
   for(int i = 0; i<Organization.getbeneficiaryList().size(); i++){
      if(Organization.getbeneficiaryList().get(i).getPhone().equals(ans)){
         search=2; //γίνεται 2 αν είναι ben
         b=Organization.getbeneficiaryList().get(i);
         System.out.println("The phone number " + ans + " has been already added in the system as beneficiary");
         do{
         System.out.println("Do you want to sign in as: " + b.getFirstName() + " " + b.getLastName() +"? (y/n)");
         ans=scan.nextLine();
          if ((ans.equals("n") || ans.equals("no") )) {exit=true; LogOut();}
          else if(ans.equals("y") || ans.equals("yes") ) {
             exit=true;
          Menu.MenuB(b);  //Καλέι την μέθοδο με όρισμα την βοηθητική μεταβλητή b τύπου Benefiaciary με βάση το τηλέφωνο που βρέθηκε
          }
         }while(!exit);
      }
    }

   //κάνει τον ίδιο έλεγχο για έναν Admin
   if (Organization.getAdmin().getPhone().equals(ans)) {
      search = 3; //γίνεται 3 αν είναι adm
      System.out.println("The phone number " + ans + " has been already added in the system as admin");
       do{
      System.out.println("Do you want to sign in as: " + Organization.getAdmin().getFirstName() + " " + Organization.getAdmin().getLastName() +"? (y/n)");
       ans=scan.nextLine();
       if ((ans.equals("n") || ans.equals("no") )) {exit=true; LogOut();}
      else if(ans.equals("y") || ans.equals("yes") ) {
         exit=true;
       Menu.MenuA(Organization.getAdmin());
      }
     }while(!exit);
```

```
    }

    //στην περίπτωση που δεν υπάρχει το τηλέφωνο ρωτάει εάν θέλει να εγγραφεί
    if(search==0) { //παραμένει 0 εάν δεν υπάρχει το τηλ
        boolean ex= false;
        do{
       System.out.println("Do you want to sign up in our system? (y/n)");
       ans=scan.nextLine();
       if ((ans.equals("n") || ans.equals("no") )) {ex=true; ExitFromProgram();}
       else if(ans.equals("y") || ans.equals("yes") ) {
          ex=true;
          exit=false;
          //ρωτάει αν θέλουμε να εγγραφούμε ως ben ή ως don
          do{
          System.out.println("Do you want to sign up as (1)Beneficiary or (2)Donator? (1/2)");
          ans=scan.nextLine();
          switch (ans) {
             case "1":
             Menu.SignB();
             exit=true;
             break;

             case "2":
             Menu.SignD();
             exit=true;
             break;

             default:
                System.out.println("Invalid command!"); //λάθος επιλογή
             break;
          }
       }while(!exit);

    }
    }while(!ex);
  }
}

//Μέθοδος εγρραφής χρήστη ως Beneficiary
 public static void SignB() {
    Beneficiary b = new Beneficiary();
    b.setPhone(phone);

    System.out.println("Enter your First Name: ");
    ans=scan.nextLine();
     while (ans.equals("")) {
      System.out.println("Enter your First Name: ");
      ans=scan.nextLine(); }
    b.setFirstName(ans);

    System.out.println("Enter your Last Name: ");
```

```
    ans=scan.nextLine();
     while (ans.equals("")) {
      System.out.println("Enter your Last Name: ");
      ans=scan.nextLine(); }
    b.setLastName(ans);

    System.out.println("Insert members of your family: ");
    ans=scan.nextLine();
     while (ans.equals("")) {
      System.out.println("Insert members of your family: ");
      ans=scan.nextLine();
    }
    b.setNoPersons(Integer.parseInt(ans));
    Organization.insertBeneficiary(b);
    System.out.println("You sign up successfully as Beneficiay!");
    System.out.println("Press Enter key to continue...");
    scan.nextLine();
    Menu.MenuB(b);
}


//Μέθοδος εγγραφής χρήστη ως Beneficiary
 public static void SignD() {
    Donator d = new Donator();
    d.setPhone(phone);

    System.out.println("Enter your First Name: ");
    ans=scan.nextLine();
     while (ans.equals("")) {
      System.out.println("Enter your First Name: ");
      ans=scan.nextLine(); }
    d.setFirstName(ans);

    System.out.println("Enter your Last Name: ");
    ans=scan.nextLine();
     while (ans.equals("")) {
      System.out.println("Enter your Last Name: ");
      ans=scan.nextLine(); }
    d.setLastName(ans);

    Organization.insertDonator(d);
    System.out.println("You sign up successfully as Donator!");
    System.out.println("Press Enter key to continue...");
    scan.nextLine();
    MenuD(d);
}

//Menu Beneficiary
public static void MenuB(Beneficiary b) {
   //η μεταβλητή havelogout είναι true και εμφανίζεται μήνυμα χαιρετισμού και στην συνέχεια γίνεται false οπότε όταν κάνουμε back δεν εμφανίζει τον χαιρετισμό
```

```
   if (havelogout==true) {
   System.out.println("Welcome " + b.getFirstName() + " " + b.getLastName() + ", your phone is " + b.getPhone() + ", you're benefiaciary and the number of your family members are: " + b.getNoPersons() + ".");
   havelogout=false; }
   boolean exit=false;
   do{
   System.out.println("1.Add Request  2.Show Requests  3.Commit  4.Back  5.Logout  6.Exit (Enter the number of the option you want)");
   ans=scan.nextLine();
      switch (ans) {
        case "1":
       boolean Case=false;
       boolean ex=false;
       do{
         System.out.println("1.Material  2.Services (Press b to back)"  );
         ans=scan.nextLine();
         switch(ans){
            case "1":
                  Organization.showCategory(ans); //καλείται η μέθοδος που εκτυπώνει τα Materials αριθμημένα
                  do{
                  System.out.print("Type the name of a material for more details:");
                  ans=scan.nextLine();
                  boolean found=false;
                  //ψάχνει το Material και αν το βρει εμφανίζει τις πληροφορίες
                  for(Entity i:Organization.getentityList()){
                     if(ans.equals(i.getName())){
                        for(int j=0;j<Organization.getentityList().size();j++){
                           if(Organization.getentityList().get(j).getName().equals(ans)){
                              System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                           }
                        found=true;
                        //ρωτάει τον χρηστη αν θελει να λαβει το Material
                        System.out.println("Do you want to receive this material? (y/n)");
                        ans=scan.nextLine();
                        if (ans.equals("n") || ans.equals("no") ) {break;}
                        else if (ans.equals("y") || ans.equals("yes")) {
                           //ρωταει την ποσοτητα που θελει να λαβει
                           System.out.println("Enter the quantity you want to receive");
                           ans=scan.nextLine();
                           //Κατασκευή αντικειμένου Request Donation για δαιγραφη του απο την λιστα αν το λαβει καποιος
                           RequestDonation rq = new RequestDonation(i, Integer.parseInt(ans));
                           RequestDonationList.remove(rq);
                           System.out.println("You succesfully recieved " + ans + " " + i.getName());
                           break;
                        }
                     }
```

```
                }
            }
            Case=true;
            //αν δεν βρεθει το Material εμφανιζει μηνυμα
            if(found==false){System.out.println("The material doesn't exist!");}

             }while(!Case);

        ex=false; //Για να ξανατρέξει στο προηγούμενο μενου

         break;

        case "2":
            //ααντιστοιχα με το service
                Organization.showCategory(ans);
                do{
                System.out.print("Type the name of a service for more details:");
               ans=scan.nextLine();
                boolean found=false;
                for(Entity i:Organization.getentityList()){
                   if(ans.equals(i.getName())){
                      for(int j=0;j<Organization.getentityList().size();j++){
                          if(Organization.getentityList().get(j).getName().equals(ans)){
                            System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                          }
                         found=true;
                         System.out.println("Do you want to receive this service? (y/n)");
                         ans=scan.nextLine();
                         if (ans.equals("n") || ans.equals("no") ) {break;}
                         else if (ans.equals("y") || ans.equals("yes")) {
                            System.out.println("Enter the hours of that service you want to
recieve");
                            ans=scan.nextLine();
                            //Κατασκευή αντιεκιμένου Request Donation
                            RequestDonation rq = new RequestDonation(i,
Integer.parseInt(ans));
                            RequestDonationList.remove(rq);
                            //(An υπαρχει ήδη request don?)
                            System.out.println("You succesfully received " + ans + " hours of " +
i.getName());
                            break;

                         }
                      }
                   }
                }
                     Case= true;
                if(found==false){System.out.println("The service doesn't exist!");}

               }while(!Case);
```

```
            ex= false; //Για να ξανατρέξει στο αμέσως προηγούμενο menu
             break;

            case "b":
            MenuB(b);
            break;

            default:
                    System.out.println("Invalid command");
            break;
          }
        }while(!ex);
        exit=true;
         break;



          case "2":

          break;

          case "3":

          break;

          case "4":
          ex=false;
          do{
          System.out.println("Do you want to sign in as: " + b.getFirstName() + " " +
b.getLastName() +"? (y/n)");
           ans=scan.nextLine();
            if ((ans.equals("n") || ans.equals("no") )) {LogOut();}
            else if(ans.equals("y") || ans.equals("yes") ) {
            Menu.MenuB(b);
            ex=true;
           }
          }while(!ex);
          exit=true;
          break;

          case "5":
          LogOut();
          exit=true;
          break;

          case "6":
          ExitFromProgram();
          exit=true;
          break;

          default:
```

```
          System.out.println("Invalid command!");
          break;
       }
   }while(!exit);
}

//Menu Donator λειτουργεί με τον ιδιο τροπο με το menu Ben
public static void MenuD(Donator d) {
   if (havelogout==true) {
   System.out.println("Welcome " + d.getFirstName() + " " + d.getLastName() + ", your phone
is " + d.getPhone() + ", you're donator.");
   havelogout=false; }
   boolean exit=false;
   do{
      System.out.println("1.Add Offer  2.Show Offers  3.Commit  4.Back  5.Logout  6.Exit  (Enter
the number of the option you want)");
      ans=scan.nextLine();
      switch (ans) {
      case "1":
      boolean Case=false;
       boolean ex=false;
       do{
         System.out.println("1.Material  2.Services (Press b to back)"  );
         ans=scan.nextLine();
         switch(ans){
            case "1":

                    Organization.showCategory(ans);
                   do{
                    System.out.print("Type the name of a material for more details:");
                   ans=scan.nextLine();
                    boolean found=false;
                    for(Entity i:Organization.getentityList()){
                       if(ans.equals(i.getName())){
                          for(int j=0;j<Organization.getentityList().size();j++){
                              if(Organization.getentityList().get(j).getName().equals(ans)){
                                 System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                              }
                          found=true;
                          System.out.println("Do you want to donate this material? (y/n)");
                          ans=scan.nextLine();
                          if (ans.equals("n") || ans.equals("no") ) {break;}
                          else if (ans.equals("y") || ans.equals("yes")) {
                             System.out.println("Enter the quantity you want to donate");
                             ans=scan.nextLine();
                             //Κατασκευή αντιεκιμένου Request Donation
                             RequestDonation rq = new RequestDonation(i, Integer.parseInt(ans));
                             RequestDonationList.add(rq, 0.0);
```

```
                        System.out.println("You succesfully donated " + ans + " " +
i.getName());
                        break;
                    }
                  }
                }
              }
              Case=true;
              if(found==false){System.out.println("The material doesn't exist!");}

               }while(!Case);

          ex=false; //Για να ξανατρέξει

           break;

          case "2":
                   Organization.showCategory(ans);
                   do{
                   System.out.print("Type the name of a service for more details:");
                  ans=scan.nextLine();
                   boolean found=false;
                   for(Entity i:Organization.getentityList()){
                      if(ans.equals(i.getName())){
                         for(int j=0;j<Organization.getentityList().size();j++){
                             if(Organization.getentityList().get(j).getName().equals(ans)){
                               System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                             }
                            found=true;
                            System.out.println("Do you want to donate this service? (y/n)");
                            ans=scan.nextLine();
                            if (ans.equals("n") || ans.equals("no") ) {break;}
                            else if (ans.equals("y") || ans.equals("yes")) {
                               System.out.println("Enter the hours of that service you want to
donate");
                               ans=scan.nextLine();
                               //Κατασκευή αντιεκιμένου Request Donation
                               RequestDonation rq = new RequestDonation(i,
Integer.parseInt(ans));
                               RequestDonationList.add(rq, 0.0);

                               System.out.println("You succesfully donated " + ans + " hours of " +
i.getName());
                               break;

                            }
                         }
                      }
                   }
                        Case= true;
```

```
                    if(found==false){System.out.println("The service doesn't exist!");}

                  }while(!Case);
              ex= false; //Για να ξανατρέξει
               break;

              case "b":
              MenuD(d);
              break;

              default:
                    System.out.println("Invalid command");
              break;
            }
          }while(!ex);
          exit=true;
           break;


            case "2":

            break;

            case "3":

            break;

            case "4":
            ex=false;
            do{
            System.out.println("Do you want to sign in as: " + d.getFirstName() + " " +
d.getLastName() +"? (y/n)");
             ans=scan.nextLine();
              if ((ans.equals("n") || ans.equals("no") )) {LogOut();}
              else if(ans.equals("y") || ans.equals("yes") ) {
              Menu.MenuD(d);
              ex=true;
             }
            }while(!ex);
             exit=true;
            break;

            case "5":
            LogOut();
            exit=true;
            break;

            case "6":
            ExitFromProgram();
            exit=true;
            break;
```

```
        default:
        System.out.println("Invalid command!");
        break;
      }
   }while(!exit);
}

//Menu Admin
public static void MenuA(Admin a)  {
    if (havelogout==true) {
   System.out.println("Welcome " + a.getFirstName() + " " + a.getLastName() + ", your phone
is " + a.getPhone() + ", you're admin.");
   havelogout=false; }

   boolean exit=false;
   do{
   System.out.println("1.View  2.Monitor Organization  3.Back  4.Logout  5.Exit  (Enter the
number of the option you want)");
   ans=scan.nextLine();

      switch (ans) {
         case "1":
          boolean Case=false;
          boolean ex=false;
          do{
            System.out.println("1.Material  2.Services (Press b to back)"  );
            ans=scan.nextLine();
            switch(ans){
               case "1":
                        Organization.showCategory(ans);
                       do{
                        System.out.print("Type the name of a material for more details:");
                       ans=scan.nextLine();
                        boolean found=false;
                        for(Entity i:Organization.getentityList()){
                           if(ans.equals(i.getName())){
                              for(int j=0;j<Organization.getentityList().size();j++){
                                 if(Organization.getentityList().get(j).getName().equals(ans)){
                                   System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                                 }
                              }
                              found=true;
                              Case= true;
                           }
                        }
                        if(found==false){System.out.println("The material doesn't exist!");}

                       }while(!Case);
```

```
                ex=false; //Για να ξανατρέξει

                 break;

                case "2":
                        Organization.showCategory(ans);
                        do{
                        System.out.print("Type the name of a service for more details:");
                       ans=scan.nextLine();
                        boolean found=false;
                        for(Entity i:Organization.getentityList()){
                           if(ans.equals(i.getName())){
                              for(int j=0;j<Organization.getentityList().size();j++){
                                 if(Organization.getentityList().get(j).getName().equals(ans)){
                                   System.out.println(Organization.getentityList().get(j).
getEntityInfo());
                                 }
                              }
                              found=true;
                              Case= true;
                           }
                        }
                        if(found==false){System.out.println("The service doesn't exist!");}

                       }while(!Case);
                ex= false; //Για να ξανατρέξει
                 break;

                case "b":
                MenuA(a);
                break;

                default:
                        System.out.println("Invalid command");
                break;
              }
           }while(!ex);
           exit=true;
            break;

           case "2":
           ex=false;
           do{
           System.out.println("1.List Beneficiaries  2.List Donators  3.Reset Beneficiaries Lists
(Press b to back)");
           ans=scan.nextLine();
           switch(ans){
              case"1":
              ListBen();
              ex=false;
              break;
```

```
            case"2":
            ListDon();
            ex=false;
            break;

            case"3":
            for(int j=0;j<Organization.getbeneficiaryList().size();j++){
              Organization.getbeneficiaryList().get(j).receivedList.clear();}
              System.out.println("All receivedList have been cleared");
            ex=false;
            break;

            case"b":
            MenuA(a);
            break;

            default:
             System.out.println("Invalid command");
             break;
           }
         }while(!ex);

         exit=true;
         break;

         case "3":
         ex=false;
         do{
         System.out.println("Do you want to sign in as: " + a.getFirstName() + " " +
a.getLastName() +"? (y/n)");
          ans=scan.nextLine();
           if ((ans.equals("n") || ans.equals("no") )) {LogOut();}
           else if(ans.equals("y") || ans.equals("yes") ) {
           Menu.MenuA(a);
           ex=true;
          }
         }while(!ex);
          exit=true;
         break;

         case "4":
         LogOut();
         exit=true;
         break;

         case "5":
         ExitFromProgram();
         exit=true;
         break;
```

```
        default:
        System.out.println("Invalid command!");
        break;
      }
   }while(!exit);
}

//Μέθοδος υπο-menu του Admin για την διαχείρηση των Beneficiary
public static void ListBen() {
boolean e =false;
boolean e1=false;
boolean e2=false;
 Organization.listBeneficiary();
 do{
 System.out.println("Choose one Beneficiary by last name (Press b to go back)");
 ans=scan.nextLine();
 if (ans.equals("b")) {break;}
for(Beneficiary i:Organization.getbeneficiaryList()){
            if(ans.equals(i.getLastName())){
               for(int j=0;j<Organization.getbeneficiaryList().size();j++){
               if(Organization.getbeneficiaryList().get(j).getLastName().equals(ans)){
                  e=true;

System.out.println(Organization.getbeneficiaryList().get(j).getreceivedList());
                     do{
                     System.out.println("Do you want to delete the list? (y/n)");
                     ans=scan.nextLine();
                      if ((ans.equals("n") || ans.equals("no") )) {
                         e1=true;
                         do{
                         System.out.println("Do you want to delete this Beneficiary? (y/n)");
                         ans=scan.nextLine();
                         if ((ans.equals("n") || ans.equals("no") )) {e2=true; break;} // pame piso
                        else if(ans.equals("y") || ans.equals("yes") ){

Organization.removeBeneficiary(Organization.getbeneficiaryList().get(j));
                         System.out.println("Beneficiary successfully Deleted ");
                         e2=true;
                         break;
                          }
                          else{e2=false; System.out.println("Invalid command");}
                      } while(!e2);
                     }

                     else if(ans.equals("y") || ans.equals("yes") ){
                        e1=true;
                     Organization.getbeneficiaryList().get(j).receivedList.clear();
                      System.out.println("ReceivedList successfully Deleted ");
                      do{
                      System.out.println("Do you want to delete this Beneficiary? (y/n)");
                       ans=scan.nextLine();
```

```
                    if ((ans.equals("n") || ans.equals("no") )) {e2=true; break;} // pame piso
                    else if(ans.equals("y") || ans.equals("yes") ){
                     Organization.removeBeneficiary(Organization.getbeneficiaryList().get(j));
                       System.out.println("Beneficiary successfully Deleted ");
                       e2=true;
                       break;
                    }
                     else{e2=false; System.out.println("Invalid command");}
                    }while(!e2);
                    }

                  }while(!e1);
               }
        }
     }
  }
}while(!e);
}

//Μέθοδος υπο-menu του Admin για την διαχείρηση των Donator
public static void ListDon() {
boolean e=false;
boolean e1=false;
boolean e2=false;
Organization.listDonators();
do{
    System.out.println("Choose one Donator by last name (Press b to go back)");
    ans=scan.nextLine();
  if (ans.equals("b")) {break;}
 for(Donator i:Organization.getdonatorList()){
          if(ans.equals(i.getLastName())){
             for(int j=0;j<Organization.getdonatorList().size();j++){
                e=true;
                do{
                 if(Organization.getdonatorList().get(j).getLastName().equals(ans)){
                   System.out.println(Organization.getdonatorList().get(j).getoffersList());
                   e1=true;
                   do{
                       System.out.println("Do you want to delete this Donator? (y/n)");
                       ans=scan.nextLine();
                       if ((ans.equals("n") || ans.equals("no") )) {e2=true; break;} // pame piso
                      else if(ans.equals("y") || ans.equals("yes") ){
                         e2=true;
                       Organization.removeDonator(Organization.getdonatorList().get(j));
                       System.out.println("Donator successfully Deleted ");
                       break;
                        }
                        else{e2=false; System.out.println("Invalid command");}
                      }while(!e2);
                    }
                }while(!e1);
```

```
                }
            }
        }
} while(!e);
}


//Αποσύνδεση χρήστη και επιστροφή στην αρχική σελίδα
public static void LogOut() {
   System.out.println("You have log out and returned to home page");
   havelogout=true;
   try
   {
   Menu.Scan();
   }
   catch (NullLineReadedException nlre)
   {
    nlre.printStackTrace();
    }
   }

//Μέθοδος που τερματίζει το πρόγραμμα
public static void ExitFromProgram(){
   System.exit(0);
}
}
```

Κλάση NullLineReadedException

```
public class NullLineReadedException extends Exception
//εξαίρεση για την περίπτωση κενής γραμμής
{
   public NullLineReadedException(String message){
      super(message);
   }
}
```

Κλάση Offers

```
import java.util.*;

public class Offers extends RequestDonationList{
   /* public static void commit(Organization org, Donator d){
   for (RequestDonation rd:d.getoffersList().getrdEntities())
   {
      org.getCurrentDon().add(rd,org);
   }

   Donator.getoffersList().clear(); //εκαθάριση της rdEntities
   } */
  /*  public static void commit(){
      for (int i=0; i<getrdEntities().size(); i++) {
         addCuDon(get(i).getrdEntities()) ; } */
```

```
}
```

Κλάση Organization

```
import java.util.*;
public class Organization {
      private static String name;
      private static Admin admin;

      //λίστα με τα είδη που μπορούν να διανεμηθούν σε Beneficiaries
      static ArrayList<Entity> entityList = new ArrayList<Entity>();
      //λίστα με τους δωρητές
      static ArrayList<Donator> donatorList = new ArrayList<Donator>();
      //λίστα με τους επωφελούμενους
      static ArrayList<Beneficiary> beneficiaryList = new ArrayList<Beneficiary>();

      public static ArrayList<Entity> getentityList(){
      return entityList;
   }

      public static ArrayList<Donator> getdonatorList(){
         return donatorList;
      }

      public static ArrayList<Beneficiary> getbeneficiaryList(){
         return beneficiaryList;
      }

      //constructor χωρίς ορίσματα
      public Organization(){}

      //constructor με ορίσματα όνομα και admin
      public Organization(String name, Admin admin)
      {this.name=name;
       this.admin=admin; }

      //δημιουργία αντικειμένου currentDonations τύπου RequestDonationList
      public static RequestDonationList currentDonations;
      public static RequestDonationList getCurrentDon(){return currentDonations;}
      public static void setCurrentDon(RequestDonationList currDon)
{currentDonations=currDon;}

      //getter-setter για admin
      public static Admin getAdmin() {return admin;}
      public void setAdmin(Admin admin) {this.admin=admin;}

      //getter για το όνομα του οργανισμού
      public static String getOrgName() {return name;}

      //ελέγχει αν υπάρχει ήδη το entity που βάζουμε και αν δεν υπάρχει το προσθέτει
      public void addEntity(Entity entity) {
          if(entityList.contains(entity)){
```

```
            throw new IllegalArgumentException("This entity is already in the list."); //εξαίρεση εάν υπάρχει ήδη
            }
        else{
            entityList.add(entity);}
    }

    /* public void addCuDon(RequestDonation rd){
        currentDonations.add(rd);  } */                    //μέθοδος που θα προσέθετε ένα requestdonation στην currentdonation

    //διαγράφει ένα entity
    public void removeEntity(int entity) {entityList.remove(entity);}

    //προσθήκη donator στην donator list
    public static void insertDonator(Donator donator) {
        if(donatorList.contains(donator)){
            //εξαίρεση ελέγχει αν υπάρχει ήδη ο donator που βάζουμε
            throw new IllegalArgumentException("This donator is already in the list.");
            }
        else{
            donatorList.add(donator);}
        }

    //αφαιρεί έναν donator απο τον οργανισμό
    public static void removeDonator(Donator donator) {donatorList.remove(donator);}


    //προσθήκη benenficiary στην beneficiarylist
    public static void insertBeneficiary(Beneficiary beneficiary) {
        if(beneficiaryList.contains(beneficiary)){
            //εξαίρεση ελέγχει αν υπάρχει ήδη ο beneficiary που βάζουμε
            throw new IllegalArgumentException("This beneficiary is already in the list.");
            }
        else{
            beneficiaryList.add(beneficiary);}
     }

     //αφαιρεί έναν Beneficiary απο τον οργανισμό
    public static void removeBeneficiary(Beneficiary beneficiary) {beneficiaryList.remove(beneficiary);}

    //υπάρχουσες κατηγορίες των entity και λίστα με όλα τα entity ανά συγκεκριμένη κατηγορία
    public static void listEntities() {for (Entity entity : entityList) { System.out.println(entity.getDetails()); }}

    //εμφανίζει τους δωρητές Donators
    public  static  void listDonators() {
        int count=1;
        for (Donator donator : donatorList)
```

```
        { System.out.println(count + ". " + donator.getFirstName() + " " +
donator.getLastName() );
        count++;}
      }

      //εμφανίζει τους Beneficiary
      public static void listBeneficiary() {
        int count=1;
        for (Beneficiary beneficiary : beneficiaryList)
        { System.out.println(count + ". " + beneficiary.getFirstName() + " " +
beneficiary.getLastName());
          count++; }
      }

      //προσδιορισμός entity (service-material) για εκτύπωση στο menu
      public static void showCategory(String Category){
        int counter=1;
        switch(Category){
          //αν το Category ειναι 1 τότε εμφανίζει τα Materials αριθμημένα
          case "1":
            for(Entity i: entityList){
              if(i.getCategory()==1){
                System.out.println(""+ counter + " " + i.getName());
                counter++;
              }
            }
            break;

          //αν το Category ειναι 2 τότε εμφανίζει τα Services αριθμημένα
          case "2":
            for(Entity i: entityList){
              if(i.getCategory()==2){
                System.out.println(""+ counter + " " + i.getName());
                counter++;
              }
            }
            break;

          default:
          System.out.println("Invalid command!");
            break;
        }
      }
}
```

Κλάση Request

```
public class Request extends RequestDonationList{

   //ελέγχει αν το entity είναι διαθέσιμο //@override
   public static void add(RequestDonation requestdonation, double quant){
       for (int i = 0; i<rdEntities.size(); i++)
```

```
    {
        //εάν είναι ανανεώνει την ποσότητα
    if(rdEntities.get(i).getEntity().getId()== requestdonation.getEntity().getId())
    {
       requestdonation.setQuantity(requestdonation.getQuantity()+quant);
       rdEntities.get(i).setQuantity(requestdonation.getQuantity()+quant);
    }
        //αλλιώς δημιουργεί καινούργιο
    else if (rdEntities.get(i).getEntity().getId() != requestdonation.getEntity().getId()){
    rdEntities.add(requestdonation);}
  }
  }

  //ελέγχει αν ο Beneficiary δικαιούται την ποσότητα
  public void modify(RequestDonation d ,double quantity){
    if(rdEntities.contains(d)){
    d.setQuantity(quantity);
    rdEntities.get(rdEntities.indexOf(d)).setQuantity(quantity);
  }
  }

  //ελέγχει την ποσότητα ενός RequestDonation και αν είναι επιτρεπτή για των αριθμό μελών που δηλώνει ο Beneficiary
  public boolean validRequestDonation(Beneficiary b, RequestDonation d){  // έλεγχος ποσοτητας
      boolean l=false;
     if(b.getNoPersons()==1) {
        if(d.getQuantity()<=Material.getLevel1()){
          l = true;
     }
     }
     else if(b.getNoPersons()>=2 && b.getNoPersons()<=4) {
         if(d.getQuantity()<=Material.getLevel2()){
          l = true;
     }
     }
     else if(b.getNoPersons()>=5) {
         if(d.getQuantity()<=Material.getLevel3()){
          l = true;
     }
  }
     return l;
  }

  /* public void commit(Beneficiary b, RequestDonation d){
     try{
        if(validRequestDonation(b,d)){
           rdEntities.remove(d);
           b.receivedList.add(d); }
        }
        catch(Exception e){
```

```
            System.out.println("You can't make a request :( ");
        }
    }  */
}
```

Κλάση RequestDonation

```
import java.util.*;
public class RequestDonation
{
    Entity entity;
    double quantity;

    //consuctor με ορίσματα entity και quantity
    public RequestDonation(Entity entity, double quantity){
    this.entity=entity;
    this.quantity=quantity;
    }

    //επιστρέφει το id για ένα requestdonation
    public int get_id(){return entity.getId();}

    //getter για το entity
    public Entity getEntity(){return entity;}

    //setter-getter για το Quantity
    public void setQuantity(double quantity){this.quantity=quantity;}
    public double getQuantity(){return quantity;}

    //interface comparator για την σύγκριση δύο αντικειμένων με βάση το ID
    private int compare(Entity d1, Entity d2)
        {
            if (d1.getId()==d2.getId()){return 0;}
            else {return 1;}
        }
}
```

Κλάση RequestDonationList

```
import java.util.*;
public class RequestDonationList{
    public  static  ArrayList<RequestDonation>  rdEntities=new  ArrayList<RequestDonation>();
//δημιουργία λίστας
    static RequestDonation rdon;

    public static ArrayList<RequestDonation> getrdEntities(){
        return rdEntities;
    }

    //constructor χωρίς ορίσματα
    public RequestDonationList(){}

    //getter για την RequestDonation
```

```
   public static RequestDonation getRequestDonation(){return rdon;}

   //καταχώρηση ενός id του entity και επιστροφή του αντίστοιχου requestdonation
   public static RequestDonation get(int id){
      RequestDonation r =null;
      for (int i = 0; i<rdEntities.size(); i++){
         if (id==rdEntities.get(i).getEntity().getId()){ r= rdEntities.get(i);}
       }
       return r;
    }

   //προσθήκη requestdonation στην rdentities και τροποποίση του quantity εάν ήδη υπάρχει
   public static void add(RequestDonation requestdonation, double quant){
       for (int i = 0; i<rdEntities.size(); i++)
      {
      if(rdEntities.get(i).getEntity().getId()== requestdonation.getEntity().getId()) {
         requestdonation.setQuantity(requestdonation.getQuantity()+quant);
         rdEntities.get(i).setQuantity(requestdonation.getQuantity()+quant);
      }
      else if (rdEntities.get(i).getEntity().getId() != requestdonation.getEntity().getId()){
      rdEntities.add(requestdonation);}
   }
   }

   //διαγραφή ενός requestdonation από την λίστα
    public static void remove(RequestDonation requestdonation){
      rdEntities.remove(requestdonation);
   }

   //αλλαγή της ποσότητας ενός product
   public void modify(RequestDonation d ,double quantity){
      if(rdEntities.contains(d)){
      d.setQuantity(quantity);
      rdEntities.get(rdEntities.indexOf(d)).setQuantity(quantity);
   }
   }

   //εκτύπωση των στοιχείων της λίστας
   public void monitor(){
      for (RequestDonation requestdon : rdEntities) {
         System.out.println("name of donation " + requestdon.getEntity().getName() + ",
quantity " +requestdon.getQuantity()); }
      }

   //εκαθάριση της λίστας
   public void reset(){
   rdEntities.clear();
   }
}
```

Κλάση Service

```
import java.util.*;
public class Service extends Entity{
   public int EntityCategory=2;

   //μέθοδος που επιστρέφει το είδος της δωρεάς
   public String getDetails(){
        return ("the object is service");
   }

   //@override
   public String toString(){
   return "info: " + getEntityInfo() + "details "+ getDetails();
   }

   //μέθοδος που δηλώνεται για να χρησιμοποιηθεί στην main ώστε να εκτυπώνει στον Admin την ποσότητα των Services
   public int getCategory(){
      return EntityCategory;
   }
}
```

Κλάση User

```
public abstract class User {
   private  String firstname;  //Επιλέγουμε να έχουμε και όνομα και επώνυμο αντί για σκέτο name
   private  String lastname;
   private String phone;

   //constructor χωρίς ορίσματα
   public User(){}

   //constructor με ορίσματα (προσθήκη πληροφοριών για χρήστη)
   public User (String first, String last, String ph)
   {   firstname = first;
      lastname = last;
      phone = ph; }

   //getter-setter για το όνομα
   public  String getFirstName() { return firstname; }
   public  void setFirstName(String first) {firstname=first;}

   //getter-setter για επίθετο
   public  String getLastName() { return lastname; }
   public  void setLastName(String last) { lastname=last;}

   //getter-setter για τηλέφωνο
   public String getPhone() { return phone; }
   public void setPhone(String phone) { this.phone=phone;}
```

```
   //μέθοδος επιστρέφει false για όλες τις υποκλάσεις εκτός από το Admin που έχουμε κάνει override.
   public boolean isAdmin() { return false; }
}
```