PROJECT

ΟΝΤΟΚΕΝΤΡΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 2021 (JAVA)

Στοιχεία μελών ομάδας:

ΧΑΛΛΑΣ ΧΑΡΑΛΑΜΠΟΣ-ΜΑΡΙΟΣ – 1084589 – up1084589@upnet.gr

ΚΙΖΙΛΗΣ ΦΩΤΙΟΣ – 1084588 – up1084588@upnet.gr

ΒΙΛΛΙΩΤΗΣ ΑΧΙΛΛΕΑΣ – 1084567 – up1084567@upnet.gr

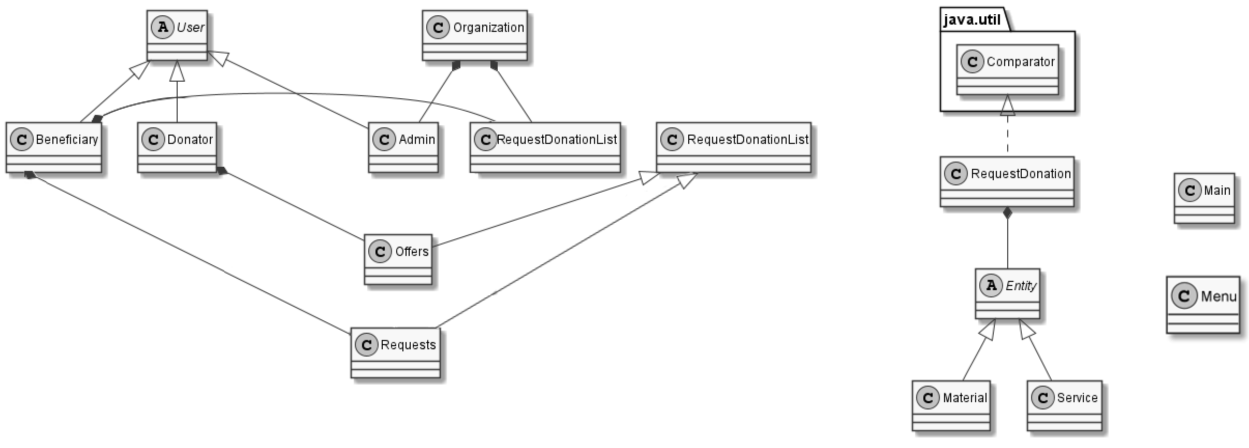
github repo: http://github.com/AxillV/Project-Java

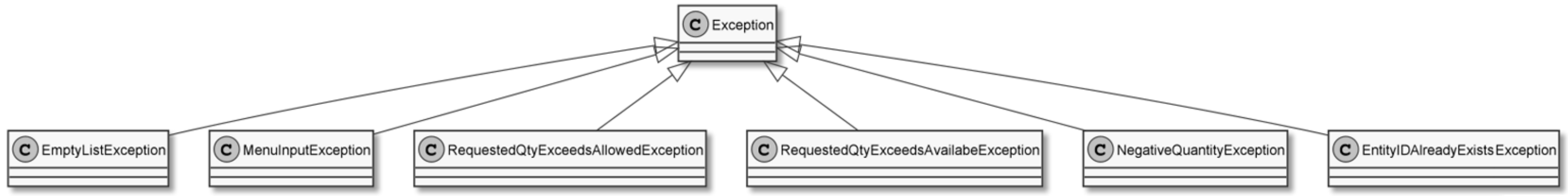
UML Class Diagram σε μορφή αρχείου PUML (PlantUML):

```
@startuml
class Offers {
}
class Menu {
}
class EmptyListException {
}
class Material {
}
class Donator {
}
class Main {
}
class Requests {
}
class Admin {
}
abstract class Entity {
}
class MenuInputException {
}
class RequestedQtyExceedsAllowedException {
}
class RequestDonationList {
}
abstract class User {
}
class Organization {
}
class RequestedQtyExceedsAvailabeException {
}
```

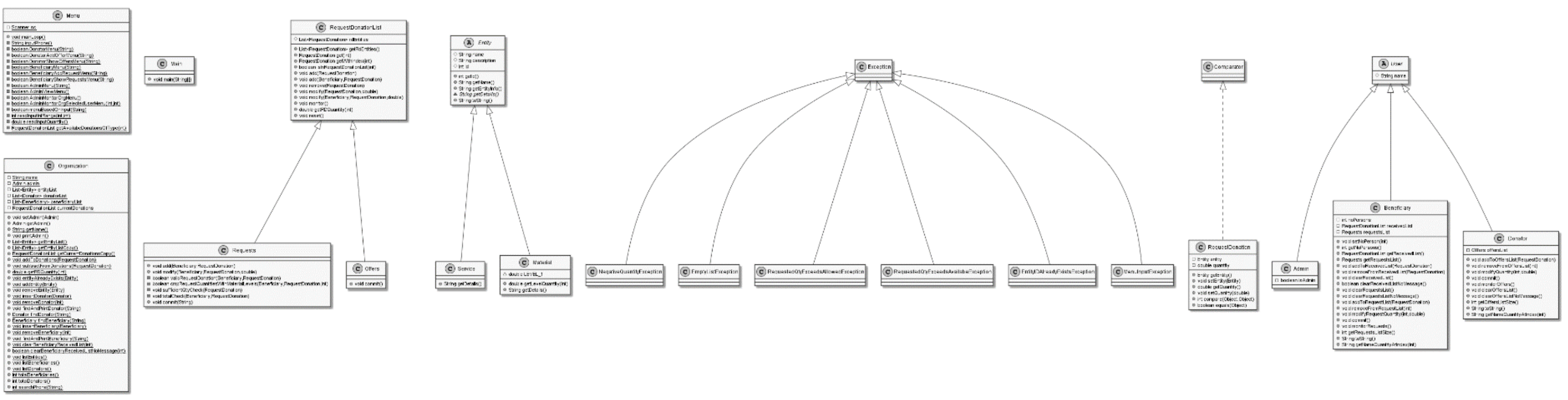
```
class Beneficiary {
}
class Service {
}
class NegativeQuantityException {
}
class RequestDonation {
}
class EntityIDAlreadyExist {
}

Exception <|-- EntityIDAlreadyExistsException
RequestDonationList <|-- Offers
Exception <|-- EmptyListException
Entity <|-- Material
User <|-- Donator
RequestDonationList <|-- Requests
User <|-- Admin
Exception <|-- MenuInputException
Exception <|-- RequestedQtyExceedsAllowedException
Exception <|-- RequestedQtyExceedsAvailabeException
User <|-- Beneficiary
Entity <|-- Service
Exception <|-- NegativeQuantityException
Beneficiary*--RequestDonationList
Beneficiary*--Requests
Donator*--Offers
Organization*--Admin
Organization*--RequestDonationList
RequestDonation*--Entity
java.util.Comparator <|.. RequestDonation
@enduml
```

UML Class Diagram σε μορφή εικόνας (png):




Exception
EmptyListException
MenuInputException
RequestedQtyExceedsAllowedException
RequestedQtyExceedsAvailabeException
NegativeQuantityException
EntityIDAlreadyExistsException

UML Class Diagram με μεθόδους:



Κατηγορίες χρηστών

Λόγω της έκτασης του μενού και των πολλαπλών ενεργειών που ενδέχεται να εκτελεθούν απο τον χρήστη, το έχουμε «χωρίσει» σε 4 νοητά κομμάτια: beneficiary, donator, admin και μη-εγγεγραμένος. Το κάθε κομμάτι έχει το δικό του σετ απο μεθόδους για την εμφάνιση των κατάλληλων μενού.

Ένας beneficiary μπορεί να δει το μενού επιλογών των beneficiary (BeneficiaryMenu) είσαγοντας τον αριθμό τηλεφώνου του. Μέσω αυτού μπορεί να δει και τις προσωπικές λίστες όσων έχει λάβει και όσων ζητάει, receivedList, και requestsList αντιστοίχως. Ένας donator έχει μόνο μια προσωπική λίστα, που είναι πράγματα που τίθεται να δωρίσει στον οργανισμό (offersList).

Ειδική περίπτωση αποτελεί ο μη-εγγεγραμένος (κάποιος που ο αριθμός τηλεφώνου του δεν αντιστοιχεί σε αντικείμενο donator, beneficiary ή admin στις λίστες χρηστών του οργανισμού. Ένας μη-εγγεγραμένος χρήστης έχει την δυνατότητα να εγγραφεί στον οργανισμό είτε ως επωφελούμενος ή δωρητής. Εισάγοντας το όνομά του προστίθεται στην λίστα χρηστών που επέλεξε να εγγραφεί μετά απο αυτό ανοίγει κανονικά το προσωποποιημένο μενού του με τις λίστες του.

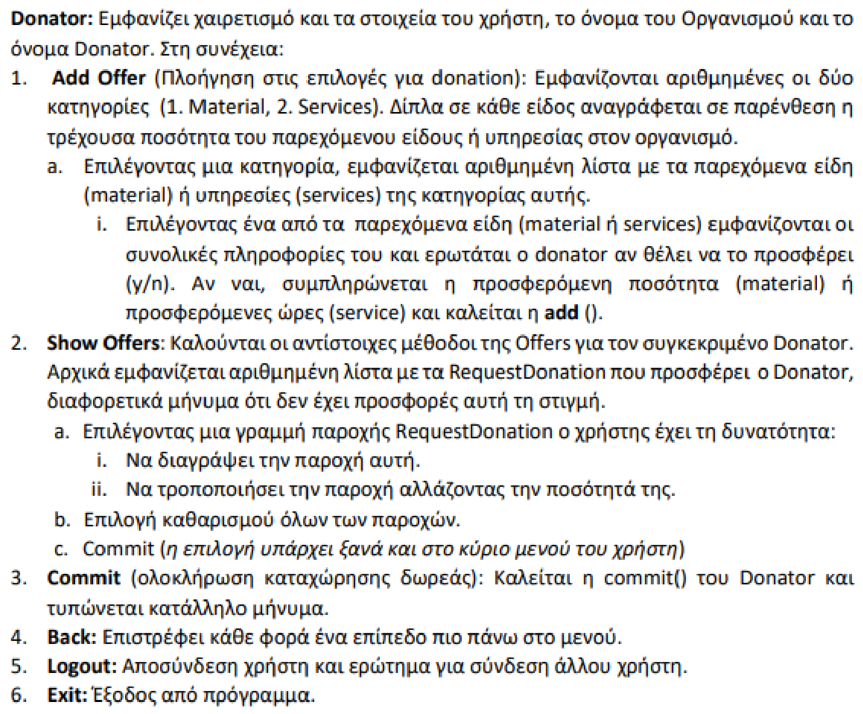
Περιήγηση στο μενού

Η περιήγηση στο μενού και τις υποκατηγορίες του γίνεται με τις μεθόδους της κλάσης Menu. Για να μην μένουν ανοιχτές πολλές μέθοδοι (π.χ. ένας δωρητής μπορεί να δει θελήσει να δει τα είδη που έχει δωρήσει, μετά να δωρήσει ένα επιπλέον είδος, να κάνει Logout, μετά κάποιος άλλος να συνδεθεί στο δικό του λογαριασμό κλπ...) κάθε μέθοδος που καλείται για την εμφάνιση πληροφοριών επιστρέφει μια boolean μεταβλητή true/false και είναι μέσα σε ένα βρόχο while της μορφής:

```
while(true) {
   if(!MenuFunction() ) break;
}
```

Με λίγα λόγια, άμα ο χρήστης αποφασίσει να βγει απο μια υποκατηγορία του μενού, είναι λογικό πως το πρόγραμμα πρέπει να του δείξει το προηγούμενο «επίπεδο» του μενού, και να μην του εμφανίσει πάλι τον αρχικό κόμβο επιλογών.

Για να επιτευχθεί αυτό, περικλείουμε την συνάρτηση με έναν βρόχο while. Όσο ο χρήστης συνεχίζει να χρησιμοποιεί τις επιλογές ενός υπο-menu (π.χ. Add Offer), η συνάρτηση επιστρέφει true, έτσι ώστε να ξανατρέξει αυτό το υπο-menu. Μόλις ο χρήστης επιλέξει την επιλογή back που υπάρχει σε κάθε υπο-menu, η συνάρτηση επιστρέφει false, ενεργοποιείται** ο βρόχος ελέγχου if και εκτελείται η εντολή break. Αυτό είναι πιο εύκολο να το καταλάβει κάποιος με ένα παράδειγμα:



Donator: Εμφανίζει χαιρετισμό και τα στοιχεία του χρήστη, το όνομα του Οργανισμού και το όνομα Donator. Στη συνέχεια:

1. **Add Offer** (Πλοήγηση στις επιλογές για donation): Εμφανίζονται αριθμημένες οι δύο κατηγορίες (1. Material, 2. Services). Δίπλα σε κάθε είδος αναγράφεται σε παρένθεση η τρέχουσα ποσότητα του παρεχόμενου είδους ή υπηρεσίας στον οργανισμό.
    a. Επιλέγοντας μια κατηγορία, εμφανίζεται αριθμημένη λίστα με τα παρεχόμενα είδη (material) ή υπηρεσίες (services) της κατηγορίας αυτής.
        i. Επιλέγοντας ένα από τα παρεχόμενα είδη (material ή services) εμφανίζονται οι συνολικές πληροφορίες του και ερωτάται ο donator αν θέλει να το προσφέρει (y/n). Αν ναι, συμπληρώνεται η προσφερόμενη ποσότητα (material) ή προσφερόμενες ώρες (service) και καλείται η **add** ().
2. **Show Offers**: Καλούνται οι αντίστοιχες μέθοδοι της Offers για τον συγκεκριμένο Donator. Αρχικά εμφανίζεται αριθμημένη λίστα με τα RequestDonation που προσφέρει ο Donator, διαφορετικά μήνυμα ότι δεν έχει προσφορές αυτή τη στιγμή.
    a. Επιλέγοντας μια γραμμή παροχής RequestDonation ο χρήστης έχει τη δυνατότητα:
        i. Να διαγράψει την παροχή αυτή.
        ii. Να τροποποιήσει την παροχή αλλάζοντας την ποσότητά της.
    b. Επιλογή καθαρισμού όλων των παροχών.
    c. Commit (*η επιλογή υπάρχει ξανά και στο κύριο μενού του χρήστη*)
3. **Commit** (ολοκλήρωση καταχώρησης δωρεάς): Καλείται η commit() του Donator και τυπώνεται κατάλληλο μήνυμα.
4. **Back:** Επιστρέφει κάθε φορά ένα επίπεδο πιο πάνω στο μενού.
5. **Logout:** Αποσύνδεση χρήστη και ερώτημα για σύνδεση άλλου χρήστη.
6. **Exit:** Έξοδος από πρόγραμμα.

Με βάση αυτή την εικόνα για την δομή του μενού δωρητή, μπορούμε να χωρίσουμε το μενού σε υποκατηγορίες ή επίπεδα. Το μενού που εμφανίζει τις επιλογές 1 έως 6 (DonatorMenu) είναι το «βασικό».

Αν πατήσουμε 1, θα κληθεί το μενού πρόσθεσης δωρεάς (DonatorAddOfferMenu). Μέχρι να επιλέξουμε back ώστε η DonatorAddOfferMenu να επιστρέψει false και να βγούμε απο το while loop της, αυτή θα καλείται μετά απο κάθε ενέργειά μας μέσα σε αυτήν.

Με αυτόν τον τρόπο παραμένουμε στο μενού πρόσθεσης δωρεάς όσο θέλουμε, μέχρι να επιλέξουμε την επιλογή back που θα μας επιστρέψει πίσω στο βασικό μενού.

Αυτή η λογική ακολουθείται απο κάθε υπο-μενού και το προηγούμενό του: **true** σημαίνει ότι ο χρήστης παραμένει στο τρέχων μενού και έχει την επιλογή να «ανέβει» επίπεδο (να καλέσει μια άλλη μέθοδο, δηλαδή να μπει ακόμη πιο μέσα στις επιλογές. Στη περίπτωση του DonatorAddOfferMenu να επιλέξει ένα είδος και να το δωρίσει), ενώ **false** σημαίνει πως θέλει να πάει πίσω στο προηγούμενο μενού.

Για υποκατηγορίες όπου λόγω της μικρής έκτασής τους δεν κρίθηκε απαραίτητη η χρήση μεθόδων χρησιμοποιήσαμε απλώς κώδικα μέσα σε while με if checks που εκτελούν break όταν ο χρήστης θέλει να πάει πίσω ή τον αναγκάζει το πρόγραμμα.

Είσοδος χρήστη

Με σκοπό την εξοικονόμηση πόρων και το γράψιμο ευανάγνωστου κώδικα, το input όλου του προγραμμάτος γίνεται με μια στατική μεταβλητή sc τύπου scanner. Λόγω αυτού, το κλείσιμο της sc (sc.close();) δεν είναι απαραίτητο, αφού το hardware που απαιτείται είναι μηδαμηνό.

Ο χρήστης εισάγει κυρίως ακέραιους αριθμούς που αντιστοιχούν σε μια επιλογή (π.χ. πατήστε 5 για να κάνετε Logout). Για το διάβασμα των επιλογών αυτών δημιουργήσαμε μια μέθοδο με όνομα readInputIntRange.

Η readInputIntRange δέχεται ως ορίσμα το εύρος επιλογών που έχει ο χρήστης (lower και upper bound) και επιστρέφει την επιλογή του χρήστη, η οποία αποθηκεύεται συνήθως σε μια ακέραια μεταβλητή inChoice.

Μέσα στο μπλοκ κώδικα της readInputIntRange γίνεται έλεγχος ώστε να διασφαλιστεί ότι η είσοδος του χρήστη είναι ακέραιος αριθμός και τίποτε άλλο (δεκαδικός, γραμμάτων, συνδυασμός αριθμών και γραμμάτων) με τον χειρισμό της εξαίρεσης InputMismatchException. Τέλος, άμα ο ακέραιος που εισάγει δεν είναι στο εύρος επιλογών που αντιστοιχεί στο τρέχων μενού, η readInputIntRange «πετάει» μια εξαίρεση MenuInputException που πρέπει να χειριστεί απο την μέθοδο που την κάλεσε.

Στη συνέχεια, μεταβλητές όπως οι inChoice χρησιμοποιούνται σε switch cases, όπου κάθε ακέραιος αντιστοιχεί και σε ένα case ενέργειας, όπως το άνοιγμα ή το κλείσιμο ενός μενού.

Για την είσοδο (θετικών) double αριθμών με σκοπό τον ορισμό ποσότητας ενός RequestDonation χρησιμοποιείται η readInputQuantity. Η μέθοδος διαβάζει τον αριθμό χρησιμοποιώντας την parseDouble και διαχειρίζεται την εξαίρεση NumberFormatException με μια try-catch. Στην περίπτωση που ο χρήστης εισάγει αρνητικό αριθμό, η readInputQuantity πετάει εξαίρεση NegativeQuantityException που διαχειρίζεται εξωτερικά απο την μέθοδο που την καλεί.

Πέρα απο την είσοδο ακέραιων μεταβλητών, μερικές φορές ο χρήστης εισάγει συμβολοσειρές για να συνεχίσει σε ένα υπο-μενού. Λόγω της ύπαρξης της μεθόδου equals(“”) για τις συμβολοσειρές, η είσοδος για π.χ. μια επιλογή yes or no (y/n) επιτυγχάνεται με την sc.nextLine().

Κλάση Organization

Σκοπός του προγράμματος είναι η ροή ειδών απο δωρητές στον οργανισμό και απο τον οργανισμό στους επωφελούμενους. Προφανώς, το πρόγραμμα πρόκειται για μόνο έναν οργανισμό, οπότε και τα πεδία της κλάσης Organization είναι πάντα ίδια, αφού αφορούν τον ίδιο οργανισμό. Για αυτό, τα κάναμε static (μέθοδοι και λίστες).

Αφού είναι static, ο χρόνος ζωής του κάθε πεδίου είναι όσο το runtime του προγράμματος και στην μνήμη κατανέμεται χώρος μόνο για ένα τέτοιο αντικείμενο ή μεθοδο.

Αυτό διευκολύνει τη συγγραφή του κώδικα επειδή για την κλήση των μεθόδων τις Organization δεν χρειάζεται κάθε φορα να περνάμε το συγκεκριμένο αντικείμενο που δημιουργήσαμε ως παράμετρο, αλλά να τις καλούμε άμεσα μέσω της κλάσης Organization (π.χ. Organization.findAndPrintDonator()).

Μπορεί επίσης το πρόγραμμα να διαχειρίζεται πολλαπλά συστήματα δεδομένων οργανισμών με την αφαίρεση των static keyword απο τις μεθόδους της κλάσης Organization και την πρόσθεση/τροποποίηση μερικών μεθόδων στη Menu κλάση.

Κλάση Menu

Κάτι παρόμοιο κάναμε με την κλάση Menu. Αφού χρειαζόμαστε μόνο ένα μενού στο πρόγραμμά μας, τα μέλη της κλάσης Menu τα δηλώσαμε ως static. Αυτή η απόφαση έγινε για τους ίδιους λόγους που κάναμε και τα μέλη της Organization στατικά.

Επίσης όπως αναφέρθηκε νωρίτερα, σε συνδυασμό με τη χρήση while βρόχων επανάληψης για το flow control επανάληψης, γίνεται πλούσια χρήση εμφωλευμένων μεθόδων.

Εξαιρέσεις

Σε περιπτώσεις μη επιτρεπτών ενεργειών, καλούνται εξαιρέσεις που υλοποιήσαμε με σκοπό την ορθή λειτουργία του προγράμματος.

MenuInputException: Όταν ο χρήστης εισάγει έναν αριθμό που δεν αντιστοιχεί σε έγκυρη επιλογή μενού. Διαχειρίζεται με μια while η οποία συνεχίζει μέχρι ο χρήστης να δώσει έγκυρη επιλογή.

NegativeQuantityException: Όταν ο χρήστης εισάγει αρνητικό αριθμό ενός είδους που θα ήθελε να δωρίσει ή να ζητήσει. Διαχειρίζεται με μια while η οποία συνεχίζει μέχρι ο χρήστης να δώσει έγκυρη ποσότητα.

EmptyListException: Πολλές φορές όταν γίνεται προσπάθεια για commit ή για καθαρισμό μιας λίστας, η λίστα μπορεί να είναι ήδη κενή. Κρίναμε απαραίτητη την προσθήκη ενός ενημερωτικού μηνύματος σε τέτοιες περιπτώσεις, ώστε ο χρήστης να ξέρει ακριβώς τι γίνεται «μέσα» στο πρόγραμμα και κάθε αλλαγή ή μη-αλλαγή να διερμηνεύεται ρητά.

RequestedQtyExceedsAllowedException: Όταν ένας επωφελούμενος ζητήσει ποσότητα είδους που δεν αναλογεί στα άτομα που αντιπροσωπεύει ο λογαριασμός του (noPersons).

RequestedQtyExceedsAvailabeException: Όταν ένας επωφελούμενος ζητήσει ποσότητα είδους που υπερβαίνει τη ποσότητα που υπάρχει εκείνη τη στιγμή στον Organization διαθέσιμη.

EntityIDAlreadyExistsException: Αυτή η εξαίρεση δεν είναι δυνατό να προκληθεί απο εναν απλό χρήστη του προγράμματος παρά μόνο έναν προγραμματιστή στο κώδικα της main — προκαλείται μόνο με την κλήση της μεθόδου addEntity που προσθέτει ένα αντικείμενο Entity στην λίστα των αντικειμένων που διακινεί ο οργανισμός. Αν το αντικείμενο που προστίθεται υπάρχει ήδη, η μέθοδος entityAlreadyExists πετάει εξαίρεση αυτού του τύπου.

Για τις τρεις τελευταίες εξαιρέσεις, σαν όρισμα δίνεται το αντικείμενο που προκάλεσε την εξαίρεση ώστε ο χρήστης να είναι καλύτερα ενημερωμένος για το πρόβλημα που παρουσιάστηκε.

Debugging

Στη main υπάρχουν οι εξής μέθοδοι για debugging (μέσα σε σχόλια ώστε να μην γεμίζει το terminal υπό κανονικές συνθήκες), κατά σειρά:

- Προσθήκη δωρεών στον οργανισμό
- Μέθοδοι για επίδειξη της εξαίρεσης EntityIDAlreadyExistsException.

Επίσης, μέσα στο try catch υπάρχουν διάφορες μέθοδοι για:

- Δοκιμή ενεργειών των beneficiary (π.χ. αλλαγή αριθμών ατόμων)
- Δοκιμή ενεργειών των donator (π.χ. προσθήκη δωρεάς).

Τέλος, υπάρχουν μέθοδοι αποκλειστικά για debugging όπως:

- Εμφάνιση εκκρεμών Request
- Εμφάνιση εκκρεμών Offer.

Κώδικας Ανά Αρχείο

Admin.java

```
public class Admin extends User{

    private boolean isAdmin = true;

    Admin(String name, String phone) {
        super(name, phone);
    }
}
```

Beneficiary.java

```
public class Beneficiary extends User{

    Beneficiary(String name, String phone) {
        super(name, phone);
    }

    Beneficiary(String name, String phone, int noPersons) {
        super(name, phone);
        this.noPersons = noPersons;
    }

    private int noPersons = 1;

    private RequestDonationList receivedList = new RequestDonationList(); //have
    private Requests requestsList = new Requests();//want

    public void setNoPerson(int noPersons){
        this.noPersons = noPersons;
    }

    public int getNoPersons() {
        return noPersons;
    }

    public RequestDonationList getReceivedList() {
        return receivedList;
    }

    public Requests getRequestsList() {
        return requestsList;
    }

    public void addToReceivedList(RequestDonation requestDonation) {
        receivedList.add(requestDonation);
    }
```

```
public void removeFromReceivedList(RequestDonation requestDonation) {
    receivedList.remove(requestDonation);
}

public void clearReceivedList(){
    try {
        receivedList.reset();

    } catch (EmptyListException ELe) {
        System.out.println(ELe.getMessage());
        return;
    }

    System.out.println("The received list has been cleared successfully");
}
/**
 * @return true if list was not already clear else false
 */
public boolean clearReceivedListNoMessage(){
    try {
        receivedList.reset();
        return true;
    } catch (EmptyListException ELe) {
        return false;
    }
}

public void clearRequestsList(){
    try {
        requestsList.reset();

    } catch (EmptyListException ELe) {
        System.out.println(ELe.getMessage());
        return;
    }

    System.out.println("The pending requests have been cleared successfully");
}

public void clearRequestsListNoMessage(){
    try {
        requestsList.reset();
    } catch (EmptyListException ELe) {
        return;
    }
}
```

```
    public void addToRequestList(RequestDonation request) throws NegativeQuantityExcepti
on {
        if (request.getQuantity()<0) {
            throw new NegativeQuantityException();
        }
        requestsList.add(this, request);
    }

    public void removeFromRequestList(int index) {
        requestsList.remove(requestsList.getWithIndex(index));
    }

    public void modifyRequestQuantity(int index, double quantity) {
        requestsList.modify(this, requestsList.getWithIndex(index), quantity);
    }

    public void commit() {
        requestsList.commit(this.phone);
    }

    public void monitorRequests(){
        requestsList.monitor();
    }

    public int getRequestsListSize(){
        return requestsList.getRdEntities().size();
    }

    @Override
    public String toString() {
        return this.name + ", " + this.phone + " (Beneficiary at " + Organization.getName() + ")";
    }

    public String getNameQuantityAtIndex(int index) {
        var rdEntity = requestsList.getWithIndex(index);
        return " ‖ " + rdEntity.getEntity().getName() + " (" + rdEntity.getQuantity() + ")";
    }

}
```

Donator.java:

```
public class Donator extends User{

    Donator(String name, String phone) {
        super(name, phone);
    }
```

```java
private Offers offersList = new Offers();

public void addToOffersList(RequestDonation rd) throws NegativeQuantityException {

   if (rd.getQuantity()<0) {
      throw new NegativeQuantityException();
   }
   offersList.add(rd);
}

public void removeFromOffersList(int index) {
   offersList.remove(offersList.getWithIndex(index));
}

public void modifyQuantity(int index, double quantity) {
   offersList.modify(offersList.getWithIndex(index), quantity);
}

public void commit() {
   offersList.commit();
}

public void monitorOffers(){
   offersList.monitor();
}

public void clearOffersList(){
   try {
      offersList.reset();

   } catch (EmptyListException ELe) {
      System.out.println(ELe.getMessage());
      return;
   }

   System.out.println("The pending offers have been cleared successfully");

}

public void clearOffersListNoMessage(){
   try {
      offersList.reset();
   } catch (EmptyListException ELe) {
      return;
   }
}

public int getOffersListSize(){
        return offersList.getRdEntities().size();
    }

    @Override
    public String toString() {
        return this.name + ", " + this.phone + " (Donator at " + Organization.getName() + ")";
    }

    public String getNameQuantityAtIndex(int index) {
        var rdEntity = offersList.getWithIndex(index);
        return rdEntity.getEntity().getName() + " (" + rdEntity.getQuantity() + ")";
    }

}
```

EmptyListException.java

```
public class EmptyListException extends Exception {

    //If a list is getting deleted but is empty, throw this exception
    EmptyListException() {
        super("The list you are trying to clear is already empty!");
    }
}
```

Entity.java

```
abstract public class Entity

{
    //Variables protected so they can be initialized from Material or Service
    protected String name;
    protected String description;
    protected int id;

    Entity(String name, String description, int id) {
        this.name = name;
        this.description = description;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getEntityInfo() {
        return name + ": " + description + ". (" + id + ")";
    }

    abstract String getDetails();

    @Override
    public String toString() {
        return
        "╔" + "═".repeat(9) + "╦" + "═".repeat(39)
        + "\n║ Info     ║ " + this.getEntityInfo()
        + "\n╠" + "═".repeat(9) + "╬" + "═".repeat(39)
        + "\n║ Details ║" + this.getDetails()
        + "\n╚" + "═".repeat(9) + "╩" + "═".repeat(39);
    }

}
```

EntityIDAlreadyExistsException.java

```
public class EntityIDAlreadyExistsException extends Exception{
    /**If there is an entity added to the system and another one already exists with
    the same ID, throw this exception*/
    public EntityIDAlreadyExistsException(Entity entity) {
        super("The ID " + entity.getId()+ " that the new entity \"" + entity.getName() + "\""
        + " uses already exists in the system. Therefore the new entity was not added.");
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        Organization org1 = new Organization("Amazon");

        Material wood = new Material("Wood", "piece of wood", 3546, 234, 764, 3495);
        Material bandaids = new Material("Bandaids", "5 bandaids", 1236, 123, 545, 1234);
        Material tissues = new Material("Tissues", "box of tissues", 6546, 231, 900, 3000);

        Organization.addEntity(wood);
        Organization.addEntity(bandaids);
        Organization.addEntity(tissues);

        Service MedicalSupport = new Service("MedicalSupport", "Assistance", 3542);
        Service NurserySupport = new Service("NurserySupport", "Assistance", 1231);
        Service BabySitting = new Service("BabySitting", "Assistance", 6542);

        Organization.addEntity(MedicalSupport);
        Organization.addEntity(NurserySupport);
        Organization.addEntity(BabySitting);

        Beneficiary john = new Beneficiary("Giannis", "6954712676");
        Beneficiary mike = new Beneficiary("Mixalis", "6954347506");
        Donator will = new Donator("Will", "6954391840");

        Organization.insertBeneficiary(john);
        Organization.insertBeneficiary(mike);
        Organization.insertDonator(will);

        Admin nick = new Admin("Nikos", "6954512346");
        org1.setAdmin(nick);

        //DEBUGGING Tools
          //Pre-existing donations to the org.
        Organization.addToDonations(new RequestDonation(wood, 3500));
        Organization.addToDonations(new RequestDonation(bandaids, 2));
        Organization.addToDonations(new RequestDonation(MedicalSupport, 500));
        Organization.addToDonations(new RequestDonation(NurserySupport, 1.5));
          //Epideiksh EntityIDAlreadyExcistsException
        // Organization.addEntity(new Service("IDTest1", "IDTest1 Details", 3542));
        // Organization.addEntity(new Service("IDTest2", "IDTest2 Details", 3546));
        // Organization.addEntity(new Material("IDTest3", "IDTest3 Details", 3546, 11,111,1111))
;
        // Organization.addEntity(tissues);


          /*
           * Beneficiry/Donator RD testing
           *
           * NOTE: When trying to add to received list, use addToRequestList and then commit, el
se
           * the needed checks won't take place!
           * You can also skip commiting to test the functionality of the commit buttons in
           * the command line instead.
           * Lastly, activating the try/catch clause wis mandatory for most of the methods
           * being tested because of the NegativeQuantityException
           */
        //  try {
              // Permitted request + commit
          //john.addToRequestList(new RequestDonation(wood, 200));
          //john.commit();
```

```
        // Unpermitted because of negative value
      //john.addToRequestList(new RequestDonation(bandaids, -3));

        // Unpermitted because it exceeds availabe quantity at organization
      // john.addToRequestList(new RequestDonation(bandaids, 3));
      // john.setNoPerson(10);

        // Adding more to requests list and commiting with updated beneficiary LEVEL
      // john.addToRequestList(new RequestDonation(wood, 1500));
      // john.commit();
      // john.addToRequestList(new RequestDonation(wood, 2000));
      // john.commit();

      // mike.setNoPerson(2);
      // mike.addToRequestList(new RequestDonation(MedicalSupport, 2000));
      // mike.commit();

        /*
         * First one doesn't go through because of beneficiary LEVEL restrictions
         * Next 2 go through, but later on cannot be commited.
         */

      // mike.addToRequestList(new RequestDonation(wood, 765.5));
      // mike.addToRequestList(new RequestDonation(wood, 763.5));
      // mike.addToRequestList(new RequestDonation(wood, 1.5));
      // mike.commit();

      // will.addToOffersList(new RequestDonation(tissues, -400));
      // will.addToOffersList(new RequestDonation(tissues, 400));
      // will.addToOffersList(new RequestDonation(NurserySupport, 250.187));
      // will.commit();
      // } catch (NegativeQuantityException NQe){
      //    System.out.println(NQe.getMessage());
      // }

      /*
       * Prints out all donations that are currently in the system ready to be picked
       * by the beneficiaries
       */
    // for (var rD : Organization.getCurrentDonationsCopy().getRdEntities()){
    //     System.out.println(rD.getEntity().getName() + " " + rD.getQuantity());
    // }
      //Self-explanatory
```

```
        // john.monitorRequests();
        // mike.monitorRequests();
        // will.monitorOffers();

          //These are for seeing everything the beneficiaries have already received.
        // for (var rD : john.getReceivedList().getRdEntities()){
        //     System.out.println(rD.getEntity().getName() + " " + rD.getQuantity());
        // }
        // for (var rD : mike.getReceivedList().getRdEntities()){
        //     System.out.println(rD.getEntity().getName() + " " + rD.getQuantity());
        // }

        Menu.mainLoop();
        return;
    }
}
```

Material.java

```
class Material extends Entity
{
    //LEVEL_1 : ena atomo
    //LEVEL_2 : 2-4 atoma
    //LEVEL_3 : >=5

    final double LEVEL_1 , LEVEL_2, LEVEL_3;

    Material(String name, String description, int id, double LEVEL_1, double LEVEL_2, double L
EVEL_3)
    {
        super(name, description, id);
        this.LEVEL_1=LEVEL_1;
        this.LEVEL_2=LEVEL_2;
        this.LEVEL_3=LEVEL_3;
    }

    /**
     * @return To level 1/2/3, analoga me to @param level pou eisagei o xrhsths
     */
    public double getLevelQuantity(int level){
        switch (level) {
            case 1:
                return LEVEL_1;
```

```
      case 2:
        return LEVEL_2;
      case 3:
        return LEVEL_3;
      default:
        return -1;
    }
  }

  public String getDetails()
  {
    return
      "\n║ Material║"
      + "\n╠" + "═".repeat(9) + "╬" + "═".repeat(39)
      + "\n║ Level 1 ║ " + LEVEL_1
      + "\n║ Level 2 ║ " + LEVEL_2
      + "\n║ Level 3 ║ " + LEVEL_3;
  }

}
```

Menu.java

```
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.Iterator;
import java.util.List;

public class Menu {
  //Koinh scanner gia kathe methodo wste na mhn exoume memory leak
  private static Scanner sc;

  /**
   * Epilogh, h filosofia tou return true/false einai pws mexri na ginei return
   * false, theloume na epistre4oume sto idio menu me ta idia stoixeia.
   * Opote otan ginetai mia epilogh apo ton xrhsth, 1-2 kai teleiwsei h
   * leitourgeia ths, o kwdikas sto menuBasedOnInput 8a 3anatre3ei
   * to idio UserMenu (opou User = Donator,Beneficiary,Admin)
   */
  public static void mainLoop(){
    while(true){
      if(!Menu.menuBasedOnInput(Menu.inputPhone())){
        break;
      };
    }
    sc.close();
    }

    /**Diabazei to input tou xrhsth, xeirizetai eswterika thn eksairesh mh egkurou input.
     * @return Epistrefei egkuro arithmo thlefwnou.
     */
    private static String inputPhone() {
        boolean isValidInput = true;
        String phone_number;
        sc = new Scanner(System.in);

        System.out.println("╔" + "═".repeat(47 + Organization.getName().length()) + "╗");
        System.out.println("║ Welcome to the donator/beneficiary system of " + Organization.ge
tName() + " ║");
        System.out.println("╚" + "═".repeat(47 + Organization.getName().length()) + "╝");

        do {
            System.out.print("Please input your phone number to authenticate: ");
            phone_number = sc.nextLine();

            try {
                Long.parseLong(phone_number);
                isValidInput = true;
            } catch (NumberFormatException e) {
                System.out.println("Please input a valid phone number!");
                isValidInput = false;
            }
        } while (!isValidInput);

        return phone_number;
    }

    /**@return Returns true if method needs to be rerun, returns false if escape
     * back to menuBasedOnInput, or lastly exit program
     */
    private static boolean DonatorMenu(String phone) {
        System.out.println("╔" + "═".repeat(38) + "╗");
        System.out.println("║" + "█".repeat(14) + " Welcome! " + "█".repeat(14) + "║");
        System.out.println("╚" + "═".repeat(38) + "╝");
        Organization.findAndPrintDonator(phone);

        //Ektupwsh epilogwn
        System.out.println("╔" + "═".repeat(24) + "╗");
        System.out.println("║" + " ".repeat(8) + "Options" + " ".repeat(9) + "║");
        System.out.println("╠" + "═".repeat(24) + "╣");
        System.out.println("║ 1.Add Offers           ║");
System.out.println("║ 2.Show Offers        ║");
System.out.println("║ 3.Commit             ║");
System.out.println("║ 4.Back               ║");
System.out.println("║ 5.Logout             ║");
System.out.println("║ 6.Exit               ║");

//Epilogh 1 ews 6
int inChoice = -1;
do {
  try{
    inChoice = readInputIntRange(1, 6);
  } catch (MenuInputException MIe) {
    System.out.println(MIe.getMessage());
  }
} while (inChoice == -1);

/**
 * Epilogh, h filosofia tou return true/false einai pws mexri na ginei return
 * false, theloume na epistrepsoume sto idio menu me ta idia stoixeia.
 * Opote otan ginetai mia epilogh apo ton xrhsth, 1-3 kai teleiwsei h
 * leitourgeia ths, o kwdikas sto menuBasedOnInput 8a ksanatreksei
 * to idio UserMenu (opou User = Donator,Beneficiary,Admin)
 */
switch(inChoice) {
  case 1:
    while(true){
      if(!DonatorAddOfferMenu(phone)) break;
    }
    return true;

  case 2:
    while(true){
      if(!DonatorShowOffersMenu(phone)) break;
    }
    return true;

  case 3:
    Organization.findDonator(phone).commit();
    return true;

  case 4:
    Organization.findDonator(phone).clearOffersListNoMessage();
    return false;

  case 5:
```

```
            Organization.findDonator(phone).clearOffersListNoMessage();
            return false;

        case 6:
            System.out.println("╔" + "═".repeat(36) + "╗");
            System.out.println("║" + " The program will now be terminated " + "║");
            System.out.println("╚" + "═".repeat(36) + "╝");
            System.exit(0);
            return false;
    }
    return false;
}

/**@return Returns true if method needs to be rerun, returns false if escape
*/
private static boolean DonatorAddOfferMenu(String phone) {
    //Proswrinh lista gia thn emfanish material/services ston xrhsth.
    RequestDonationList availableDonations = new RequestDonationList();

    System.out.println("╔" + "═".repeat(14) + "╗");
    System.out.println("║ Categories:   ║");
    System.out.println("╠" + "═".repeat(14) + "╣");
    System.out.println("║ 1.Materials   ║");
    System.out.println("║ 2.Services    ║");
    System.out.println("║ 3.Back        ║");

    int inChoice = -1;
    do {
        try{
            inChoice = readInputIntRange(1, 3);
        } catch (MenuInputException MIe) {
            System.out.println(MIe.getMessage());
        }
    } while (inChoice == -1);

    if (inChoice == 3) return false;    //Return false aka. return to donmenu

    double quantity = 0;              //H posothta tou eidous pou tha prosferei o xrhsths
    Entity selectedEntity = null;      //To eidos pou epilegei o xrhsths

    switch(inChoice) {
        //Case for Materials selected
        case 1:
            //Display materials until back is pressed, then go back (rerun AddOfferMenu)
            while(true){
```

```java
            //Reload arraylist in case user viewed another category before
            try {
                availableDonations.reset();
            } catch (EmptyListException ELe){}
            availableDonations = getAvailabeDonationsOfType(1);

            //Print all available products for donator and then the back button (number of bac
k button is last donation +1)
            availableDonations.monitor();
            System.out.println("║ " + (availableDonations.getRdEntities().size()+1) + ".Back");
            System.out.println("╚" + "═".repeat(49));

            inChoice = -1;
            while(inChoice == -1){
                try {
                    inChoice = readInputIntRange(1, availableDonations.getRdEntities().size() + 1);
                } catch (MenuInputException MIe) {
                    System.out.println(MIe.getMessage());
                }
            }

            //Return true (go back a step) if back is pressed
            if (inChoice == availableDonations.getRdEntities().size() + 1) return true;

            selectedEntity = availableDonations.getWithIndex(inChoice-1).getEntity();
            System.out.println(selectedEntity);

            String yesNoIn = "";
            System.out.print("Would you like to donate quantity? (y/n): ");
            yesNoIn = sc.nextLine();

            while(!(yesNoIn.equals("y") || yesNoIn.equals("n"))){
                System.out.print("Please input a valid answer! (y/n): ");
                yesNoIn = sc.nextLine();
            }

            if(yesNoIn.equals("n")) continue; //Go back to showing all materials

            quantity = -1;
            while (quantity < 0){
                try {
                    quantity = readInputQuantity();
                } catch (NegativeQuantityException NQe) {
                    System.out.println(NQe.getMessage());
                }
```

```java
          }
          try {
            Organization.findDonator(phone).addToOffersList(new RequestDonation(selecte
dEntity, quantity));
          } catch (NegativeQuantityException NQe) {}
        }

      //Case for services selected
      case 2:
        //Display services until back is pressed, then go back (rerun AddOfferMenu)
        while(true){
          //Reload arraylist in case user viewed another category before
          try {
            availableDonations.reset();
          } catch (EmptyListException ELe){}
          availableDonations = getAvailabeDonationsOfType(2);

          //Print all available products for donator and then the back button (number of bac
k button is last donation +1)
          availableDonations.monitor();
          System.out.println("║ " + (availableDonations.getRdEntities().size()+1) + ".Back");
          System.out.println("╚" + "═".repeat(49));

          inChoice = -1;
          while(inChoice == -1){
            try {
              inChoice = readInputIntRange(1, availableDonations.getRdEntities().size() + 1);
            } catch (MenuInputException MIe) {
              System.out.println(MIe.getMessage());
            }
          }

          //return true (go back a step) if back is pressed
          if (inChoice == availableDonations.getRdEntities().size() + 1) return true;

          selectedEntity = availableDonations.getWithIndex(inChoice-1).getEntity();
          System.out.println(selectedEntity);

          String yesNoIn = "";
          System.out.print("Would you like to donate quantity? (y/n): ");
          yesNoIn = sc.nextLine();

          while(!(yesNoIn.equals("y") || yesNoIn.equals("n"))){
            System.out.print("Please input a valid answer! (y/n): ");
            yesNoIn = sc.nextLine();
```

```
                    }

                    if(yesNoIn.equals("n")) continue; //Go back to showing all materials

                    quantity = -1;
                    while (quantity < 0){
                        try {
                            quantity = readInputQuantity();
                        } catch (NegativeQuantityException NQe) {
                            System.out.println(NQe.getMessage());
                        }
                    }
                    try {
                        Organization.findDonator(phone).addToOffersList(new RequestDonation(selecte
dEntity, quantity));
                    } catch (NegativeQuantityException NQe) {}
                }
            }

        return false; //if all else fails, go back to previous menu
    }

    /**
     * @return Returns true if method needs to be rerun, returns false if escape
     */
    private static boolean DonatorShowOffersMenu(String phone) {
        int offersListSize = Organization.findDonator(phone).getOffersListSize();

        Organization.findDonator(phone).monitorOffers();
        System.out.println("║ " + (offersListSize + 1) + ".Clear all requests\n"
                + "║ " + (offersListSize + 2) + ".Commit\n"
                + "║ " + (offersListSize + 3) + ".Back");
        System.out.println("╚" + "═".repeat(49));

        int inChoice = -1;
        while(inChoice == -1){
            try{
                inChoice = readInputIntRange(1, offersListSize+3);
            } catch (MenuInputException MIe) {
                System.out.println(MIe.getMessage());
            }
        }

        //Back
        if(inChoice==offersListSize+3) return false;
    if(inChoice==offersListSize+2){
      Organization.findDonator(phone).commit();
      return true;
    }

    if(inChoice==offersListSize+1){
      Organization.findDonator(phone).clearOffersList();
      return true;
    }

    //Else selection HAS to be an offer.
    while(true){
      System.out.println("╔" + "═".repeat(49));
      System.out.println(Organization.findDonator(phone).getNameQuantityAtIndex(inChoic
e-1));
      System.out.println("║ 1.Delete this offer\n"
                + "║ 2.Change quantity\n"
                + "║ 3.Back");
      System.out.println("╚" + "═".repeat(49));

      int inChoiceTwo = -1;
      while(inChoiceTwo == -1){
        try{
          inChoiceTwo = readInputIntRange(1, 3);
        } catch (MenuInputException MIe) {
          System.out.println(MIe.getMessage());
        }
      }

      if(inChoiceTwo == 3) return true; //back is pressed.

      if(inChoiceTwo == 1){
        Organization.findDonator(phone).removeFromOffersList(inChoice-1);
        System.out.println("The donation has been deleted successfully.");
        return true; //return true, go back to showing everything

      } else if (inChoiceTwo == 2) {
        double quantity = -1;
        while (quantity < 0){
          try {
            quantity = readInputQuantity();
          } catch (NegativeQuantityException NQe) {
            System.out.println(NQe.getMessage());
          }
```

```java
            }

            Organization.findDonator(phone).modifyQuantity(inChoice-1, quantity);

            //after changing quantity don't break so you can stay at the same entity
        }
    }

}

/**@return Returns true if method needs to be rerun, returns false if escape
 * back to menuBasedOnInput, or lastly exit program
 */
private static boolean BeneficiaryMenu(String phone) {
    System.out.println("╔" + "═".repeat(38) + "╗");
    System.out.println("║" + "█".repeat(14) + " Welcome! " + "█".repeat(14) + "║");
    System.out.println("╚" + "═".repeat(38) + "╝");
    Organization.findAndPrintBeneficiary(phone);

    //ektupwsh epilogwn
    System.out.println("╔" + "═".repeat(24) + "╗");
    System.out.println("║" + " ".repeat(8) + "Options" + " ".repeat(9) + "║");
    System.out.println("╠" + "═".repeat(24) + "╣");
    System.out.println("║ 1.Add Requests         ║");
    System.out.println("║ 2.Show Requests        ║");
    System.out.println("║ 3.Commit               ║");
    System.out.println("║ 4.Back                 ║");
    System.out.println("║ 5.Logout               ║");
    System.out.println("║ 6.Exit                 ║");

    int inChoice = -1;
    //input
    do {
        try{
            inChoice = readInputIntRange(1, 6);
        } catch (MenuInputException MIe) {
            System.out.println(MIe.getMessage());
        }

    } while (inChoice == -1);

    /**
     * Epilogh, h filosofia tou return true/false einai pws mexri na ginei return
     * false, theloume na epistrepsoume sto idio menu me ta idia stoixeia.
     * Opote otan ginetai mia epilogh apo ton xrhsth, 1-3 kai teleiwsei h
     * leitourgeia ths, o kwdikas sto menuBasedOnInput 8a ksanatreksei
     * to idio UserMenu (opou User = Donator,Beneficiary,Admin)
     */
    switch(inChoice) {
        case 1:
            while(true){
                if(!BeneficiaryAddRequestMenu(phone)) break;
            }
            return true;

        case 2:
            while(true) {
                if(!BeneficiaryShowRequestsMenu(phone)) break;
            }
            return true;

        case 3:
            Organization.findBeneficiary(phone).commit();
            return true;

        case 4:
            Organization.findBeneficiary(phone).clearRequestsListNoMessage();
            return false;

        case 5:
            Organization.findBeneficiary(phone).clearRequestsListNoMessage();
            return false;

        case 6:
            System.out.println("╔" + "═".repeat(36) + "╗");
            System.out.println("║" + " The program will now be terminated " + "║");
            System.out.println("╚" + "═".repeat(36) + "╝");
            System.exit(0);
            return false;
    }
    return false;
}

/**@return Returns true if method needs to be rerun, returns false if escape
*/
private static boolean BeneficiaryAddRequestMenu(String phone) {
    //Proswrinh lista gia thn emfanish material/services ston xrhsth.
    RequestDonationList availableDonations = new RequestDonationList();

    System.out.println("╔" + "═".repeat(14) + "╗");
System.out.println("║ Categories:  ║");
System.out.println("╠" + "═".repeat(14) + "╣");
System.out.println("║ 1.Materials  ║");
System.out.println("║ 2.Services   ║");
System.out.println("║ 3.Back       ║");

int inChoice = -1;
do {
    try{
        inChoice = readInputIntRange(1, 3);
    } catch (MenuInputException MIe) {
        System.out.println(MIe.getMessage());
    }
} while (inChoice == -1);

if (inChoice == 3) return false;    //Return false aka. return to donmenu

double quantity = 0;
Entity selectedEntity = null;

switch(inChoice) {
    //Case for Materials selected
    case 1:
        //Display materials until back is pressed, then go back (rerun AddRequestMenu)
        while(true){
            //Reload arraylist in case user viewed another category before
            try {
                availableDonations.reset();
            } catch (EmptyListException ELe){}

            availableDonations = getAvailabeDonationsOfType(1);

            inChoice = -1;
            availableDonations.monitor();
            System.out.println("║  " + (availableDonations.getRdEntities().size()+1) + ".Back");
            System.out.println("╚" + "═".repeat(49));

            while(inChoice == -1){
                try {
                    inChoice= readInputIntRange(1, availableDonations.getRdEntities().size() + 1);
                } catch (MenuInputException MIe) {
                    System.out.println(MIe.getMessage());
                }
            }
```

```
          //return true (go back a step) if back is pressed
          if (inChoice == availableDonations.getRdEntities().size() + 1) return true;

          selectedEntity = availableDonations.getWithIndex(inChoice-1).getEntity();
          System.out.println(selectedEntity.toString());

          String yesNoIn = "";
          System.out.print("Would you like to receive quantity? (y/n): ");
          yesNoIn = sc.nextLine();

          while(!(yesNoIn.equals("y")| yesNoIn.equals("n"))){
            System.out.print("Please input a valid answer! (y/n): ");
            yesNoIn = sc.nextLine();
          }

          if(yesNoIn.equals("n")) continue; //Go back to showing all materials

          quantity = -1;
          while (quantity < 0){
            try {
              quantity = readInputQuantity();
            } catch (NegativeQuantityException NQe) {
              System.out.println(NQe.getMessage());
            }
          }

          try {
            Organization.findBeneficiary(phone).addToRequestList(new RequestDonation(se
lectedEntity, quantity));
          } catch (NegativeQuantityException NQe) {}

        }

      //Case for services selected
      case 2:
        //Display services until back is pressed, then go back (rerun AddOfferMenu)
        while(true){
          //Reload arraylist in case user viewed another category before
          try {
            availableDonations.reset();
          } catch (EmptyListException ELe){}
          availableDonations = getAvailabeDonationsOfType(2);

          inChoice = -1;
          availableDonations.monitor();
```

```java
                    System.out.println("║ " + (availableDonations.getRdEntities().size()+1) + ".Back");
                    System.out.println("╚" + "═".repeat(49));

                    while(inChoice == -1){
                        try {
                            inChoice = readInputIntRange(1, availableDonations.getRdEntities().size() + 1);
                        } catch (MenuInputException MIe) {
                            System.out.println(MIe.getMessage());
                        }
                    }

                    //return true (go back a step) if back is pressed
                    if (inChoice == availableDonations.getRdEntities().size() + 1) return true;

                    selectedEntity = availableDonations.getWithIndex(inChoice-1).getEntity();
                    System.out.println(selectedEntity.toString());

                    String yesNoIn = "";
                    System.out.print("Would you like to receive quantity? (y/n): ");
                    yesNoIn = sc.nextLine();

                    while(!(yesNoIn.equals("y")| yesNoIn.equals("n"))){
                        System.out.print("Please input a valid answer! (y/n): ");
                        yesNoIn = sc.nextLine();
                    }
                    if(yesNoIn.equals("n")) continue; //Go back to showing all materials

                    System.out.println("\nInput the quantity you would like to receive:");

                    quantity = -1;
                    while (quantity < 0){
                        try {
                            quantity = readInputQuantity();
                        } catch (NegativeQuantityException NQe) {
                            System.out.println(NQe.getMessage());
                        }
                    }
                    try {
                        Organization.findBeneficiary(phone).addToRequestList(new RequestDonation(se
lectedEntity, quantity));
                    } catch (NegativeQuantityException NQe) {}



                }
```

```
    }

    return false; //if all else fails, go back to previous menu
  }

  private static boolean BeneficiaryShowRequestsMenu(String phone) {
    int requestsListSize = Organization.findBeneficiary(phone).getRequestsListSize();

    Organization.findBeneficiary(phone).monitorRequests();
    System.out.println("║ " + (requestsListSize + 1) + ".Clear all requests\n"
                + "║ " + (requestsListSize + 2) + ".Commit\n"
                + "║ " + (requestsListSize + 3) + ".Back");
    System.out.println("╚" + "═".repeat(49));

    int inChoice = -1;
    while(inChoice == -1){
      try{
        inChoice = readInputIntRange(1, requestsListSize+3);
      } catch (MenuInputException MIe) {
        System.out.println(MIe.getMessage());
      }
    }

    if(inChoice==requestsListSize+3) return false; //Back to previous menu

    if(inChoice==requestsListSize+2){
      Organization.findBeneficiary(phone).commit();
      return true;
    }

    if(inChoice==requestsListSize+1){
      Organization.findBeneficiary(phone).clearRequestsList();
      return true;
    }

    //Else selection HAS to be a request.
    while(true){
      System.out.println("╔" + "═".repeat(49));
      System.out.println(Organization.findBeneficiary(phone).getNameQuantityAtIndex(inCh
oice-1));
      System.out.println("╠" + "═".repeat(49));
      System.out.println("║ 1.Delete this request\n"
                + "║ 2.Change quantity\n"
                + "║ 3.Back");
      System.out.println("╚" + "═".repeat(49));
```

```
        int inChoiceTwo = -1;
        while(inChoiceTwo == -1){
            try{
                inChoiceTwo = readInputIntRange(1, 3);
            } catch (MenuInputException MIe) {
                System.out.println(MIe.getMessage());
            }
        }

        if(inChoiceTwo == 3) return true; //back is pressed

        if(inChoiceTwo == 1){
            Organization.findBeneficiary(phone).removeFromRequestList(inChoice-1);
            System.out.println("The request has been deleted successfully.");
            return true; //return true so go back to showing everything

        } else if (inChoiceTwo == 2){
            double quantity = -1;
            while (quantity < 0){
                try {
                    quantity = readInputQuantity();
                } catch (NegativeQuantityException NQe) {
                    System.out.println(NQe.getMessage());
                }
            }

            Organization.findBeneficiary(phone).modifyRequestQuantity(inChoice-1, quantity);

        }
    }
}

/**@return Returns true if method needs to be rerun, returns false if escape
 * back to menuBasedOnInput, or lastly exit program
 */
private static boolean AdminMenu(String phone) {
    System.out.println("╔" + "═".repeat(38) + "╗");
    System.out.println("║" + "█".repeat(14) + " Welcome! " + "█".repeat(14) + "║");
    System.out.println("╚" + "═".repeat(38) + "╝");
    Organization.printAdmin();

    //ektupwsh epilogwn
    System.out.println("╔" + "═".repeat(24) + "╗");
    System.out.println("║" + " ".repeat(8) + "Options" + " ".repeat(9) + "║");
```

```java
System.out.println("╠" + "═".repeat(24) + "╣");
System.out.println("║ 1.View                 ║");
System.out.println("║ 2.Monitor Organization ║");
System.out.println("║ 3.Back                 ║");
System.out.println("║ 4.Logout               ║");
System.out.println("║ 5.Exit                 ║");

//input
int inChoice = -1;
do {
   try{
      inChoice = readInputIntRange(1, 5);
   } catch (MenuInputException MIe) {
      System.out.println(MIe.getMessage());
   }

} while (inChoice == -1);

switch(inChoice) {
   case 1:
      while(true){
         if(!AdminViewMenu()){
            break;
         };
      }

      return true;

   case 2:
      while(true){
         if(!AdminMonitorOrgMenu()){
            break;
         };
      }
      return true;

   case 3:
      //back
      return false;

   case 4:
      //logout
      return false;

   case 5:
```

```
            System.out.println("╔" + "═".repeat(36) + "╗");
            System.out.println("║" + " The program will now be terminated " + "║");
            System.out.println("╚" + "═".repeat(36) + "╝");
            System.exit(0);
            return false;
    }
    return false;
}
/**
 * @return Returns true if needs to be rerun, else return false to go back a menu
 */
private static boolean AdminViewMenu(){
    int inChoice = -1;

    do {
        System.out.println("╔" + "═".repeat(14) + "╗");
        System.out.println("║  Categories  ║");
        System.out.println("╠" + "═".repeat(14) + "╣");
        System.out.println("║ 1.Materials  ║");
        System.out.println("║ 2.Services   ║");
        System.out.println("║ 3.Back       ║");

        try{
            inChoice = readInputIntRange(1, 3);
        } catch (MenuInputException MIe) {
            System.out.println(MIe.getMessage());
        }

    } while (inChoice == -1);

    switch (inChoice) {
        case 1:
            //While back option is not selected, allow user to select a material to see the info
            while (true){
                //Get all availabematerials
                var availabeMaterials = getAvailabeDonationsOfType(1);

                inChoice = -1;

                availabeMaterials.monitor();
                System.out.println("║  " + (availabeMaterials.getRdEntities().size()+1) + ".Back");
                System.out.println("╚" + "═".repeat(49));

                try{
                    inChoice = readInputIntRange(1, availabeMaterials.getRdEntities().size()+1);
                } catch (MenuInputException MIe) {
                    System.out.println(MIe.getMessage());
                }

                if(inChoice == availabeMaterials.getRdEntities().size()+1) break;

                System.out.println(availabeMaterials.getWithIndex(inChoice-1).getEntity());
            }

            return true;

        case 2:
            //While back option is not selected, allow user to select a service to see the info
            while (true){
                //Get all availabe services
                var availabeServices = getAvailabeDonationsOfType(2);

                inChoice = -1;

                availabeServices.monitor();
                System.out.println("║ " + (availabeServices.getRdEntities().size()+1) + ".Back");
                System.out.println("╚" + "═".repeat(49));


                try{
                    inChoice = readInputIntRange(1, availabeServices.getRdEntities().size()+1);
                } catch (MenuInputException MIe) {
                    System.out.println(MIe.getMessage());
                }

                if(inChoice == availabeServices.getRdEntities().size()+1) break;

                System.out.println(availabeServices.getWithIndex(inChoice-
1).getEntity().toString());
            }

            return true;
        case 3:
            return false;
    }
    return false;
}

private static boolean AdminMonitorOrgMenu(){
    int inChoice = -1;
```

```
do {
  System.out.println("╔" + "═".repeat(28) + "╗");
  System.out.println("║        Categories        ║");
  System.out.println("╠" + "═".repeat(28) + "╣");
  System.out.println("║ 1.List Beneficiaries      ║");
  System.out.println("║ 2.List Donators           ║");
  System.out.println("║ 3.Reset Beneficiaries List ║");
  System.out.println("║ 4.Back                    ║");

  try{
    inChoice = readInputIntRange(1, 4);
  } catch (MenuInputException MIe) {
    System.out.println(MIe.getMessage());
  }

} while (inChoice == -1);

switch (inChoice) {
  case 1:
    inChoice = -1;

    // While back is not pressed, always ask user to select a beneficiary and then go
    // into the options menu for that beneficiary
    while(true){
      System.out.println("╔" + "═".repeat(49));
      Organization.listBeneficiaries();
      System.out.println("║  " + (Organization.totalBeneficiaries() + 1) + ".Back");
      System.out.println("╚" + "═".repeat(49));
      do {
        try{
          inChoice = readInputIntRange(1, Organization.totalBeneficiaries() + 1 );
        } catch (MenuInputException MIe) {
          System.out.println(MIe.getMessage());
        }
      } while (inChoice == -1);

      if (inChoice == Organization.totalBeneficiaries() + 1) break;

      //Show options for beneficiary until user presses back, then go back into selecting a
beneficiary.
      while (true){
        if(!AdminMonitorOrgSelectedUserMenu(1, inChoice)) break;
      }
    }
```

```java
            return true;
        case 2:
            inChoice = -1;

            // While back is not pressed, always ask user to select a donator and then go
            // into the options menu for that donator
            while(true){
                System.out.println(" ╔" + "═".repeat(49));
                Organization.listDonators();
                System.out.println(" ║ " + (Organization.totalDonators() + 1) + ".Back");
                System.out.println(" ╚" + "═".repeat(49));

                do {
                    try{
                        inChoice = readInputIntRange(1, Organization.totalDonators() + 1 );
                    } catch (MenuInputException MIe) {
                        System.out.println(MIe.getMessage());
                    }
                } while (inChoice == -1);

                if (inChoice == Organization.totalDonators() + 1) break;

                //Show options for donator until user presses back, then go back into selecting a do
nator.
                while (true){
                    if(!AdminMonitorOrgSelectedUserMenu(2, inChoice)) break;
                }
            }

            return true;
        case 3:
            //If all lists were empty, show a different message than if a single one was not.
            boolean allListsEmpty = true;
            for(int i=0; i < Organization.totalBeneficiaries(); i++){
                if(Organization.clearBeneficiaryReceivedListNoMessage(i)){
                    allListsEmpty = false;
                }
            }
            if(allListsEmpty){
                System.out.println("All lists were already empty");
            } else System.out.println("The lists have been cleared sucessfuly!");
            return true;
        case 4:
            return false;
        }
        return false;
    }

    /**
     * @param userType 1 for Beneficiaries, 2 for Donators.
     * @return true, until user presses back button, then false
     */
    private static boolean AdminMonitorOrgSelectedUserMenu(int userType, int indexOfUser)
{
        int inChoice = -1;

        // if 1 == beneficiary, else 2 == 2 donator, show needed actions
        if (userType == 1){
            do {
                System.out.println("╔" + "═".repeat(37) + "╗");
                System.out.println("║ 1.Clear beneficiary's received list ║");
                System.out.println("║ 2.Delete beneficiary                ║");
                System.out.println("║ 3.Back                              ║");
                System.out.println("╚" + "═".repeat(37) + "╝");

                try{
                    inChoice = readInputIntRange(1, 3);
                } catch (MenuInputException MIe) {
                    System.out.println(MIe.getMessage());
                }
            } while (inChoice == -1);
            if(inChoice == 3) return false;

            if(inChoice == 1){
                try {
                Organization.clearBeneficiaryReceivedList(indexOfUser-1);
                } catch (EmptyListException ELe) {
                    System.out.println(ELe.getMessage());
                }

            } else if (inChoice == 2){
                Organization.removeBeneficiary(indexOfUser-1);
                return false;
                //return false because the beneficiary doesn't exist anymore.
            }

            // Return true so this method repeats itself, aka show options again.
            return true;
    } else if (userType == 2){
        do {
            System.out.println("╔" + "═".repeat(18) + "╗");
            System.out.println("║ 1.Delete donator ║");
            System.out.println("║ 2.Back           ║");
            System.out.println("╚" + "═".repeat(18) + "╝");

            try{
                inChoice = readInputIntRange(1, 2);
            } catch (MenuInputException MIe) {
                System.out.println(MIe.getMessage());
            }
        } while (inChoice == -1);

        if(inChoice == 2) return false;

        if(inChoice == 1){
            Organization.removeDonator(indexOfUser-1);
            return false;
            //return false because the donator doesn't exist anymore.
        }

        //return to previous menu, error-protection, for the most part
        //irrelevant because you either pressed back -> false or deleted -> false.
        return false;

    }
    //If all else fails, return to previous menu.
    return false;
}

/**@return Returns true if method needs to be rerun, returns false if escape
 * to program exit or finally there is the possibility of immediately
 * exiting the program.
 */
private static boolean menuBasedOnInput(String phone) {
    sc = new Scanner(System.in);

    switch (Organization.searchPhone(phone)) {
        case 0:
            System.out.print("You are not a registered user, would you like to register? Press (y)
to begin registration " +
            "or any other key to go back. ");

            if (sc.nextLine().equals("y")) {
```

```
        System.out.print("Would you like to register as a beneficiary (1) or donator (2)? ");

        int inChoice = -1;
        do {
            try{
                inChoice = readInputIntRange(1, 2);
            } catch (MenuInputException MIe) {
                System.out.println(MIe.getMessage());
            }

        } while (inChoice == -1);

        String inName = new String();
        switch (inChoice) {
            case 1:
                System.out.print("Please enter your name: ");
                //inName = sc.next();
                inName = sc.nextLine();
                Organization.insertBeneficiary(new Beneficiary(inName, phone));
                System.out.println("Your registration as a beneficiary has been completed!\n"
);
                while(true){
                    if(!BeneficiaryMenu(phone)){
                        break;
                    };
                }
                return true;

            case 2:
                System.out.print("Please enter your name: ");
                // inName = sc.next();
                inName = sc.nextLine();
                Organization.insertDonator(new Donator(inName, phone));
                System.out.println("Your registration as a donator has been completed!\n");
                while(true){
                    if(!DonatorMenu(phone)){
                        break;
                    };
                }
                return true;

            default:
                System.out.println("Cancelling registration...");
                return true;
                }
            } else {
                System.out.println("Cancelling registration...\n");
                return true;
            }

        case 1:
            while(true){
                if(!BeneficiaryMenu(phone)){
                    break;
                };
            }
            return true;

        case 2:
            while(true){
                if(!DonatorMenu(phone)){
                    break;
                };
            }
            return true;

        case 3:
            while(true){
                if(!AdminMenu(phone)){
                    break;
                }
            }
            return true;
    }
    //sc.close();
    return false;
  }

  /**
   * @return Returns the input if it was valid, else returns -1 (handled externally with a do-
while check)
   * @throws MenuInputException
   */
  private static int readInputIntRange(int lowerBound, int upperBound) throws MenuInputE
xception{
    int inChoice = -1;

    try {
    inChoice = sc.nextInt();
    sc.nextLine();
    // nextLine is needed because of problems when reading a String line (eg. the name)
      if (inChoice > upperBound || inChoice < lowerBound){
        inChoice = -1;
        throw new MenuInputException();
      }

    } catch (InputMismatchException IMe) {
      inChoice = -1;
      System.out.println("\nYou entered characters! Please input a valid integer!");
      sc.next();
    }
    return inChoice;
  }

  private static double readInputQuantity() throws NegativeQuantityException{
    double quantity = -1;

    try {
      System.out.print("\nInput quantity: ");
      quantity = Double.parseDouble(sc.next());
    } catch (NumberFormatException NFe) {
      System.out.println("Please insert a valid number!");
      return -1;
    }

    if(quantity < 0) throw new NegativeQuantityException();

    return quantity;
  }

  /**
   * @param choice 1 for Material, 2 for Service
   * @return Returns a requestdonationlist that contains every material/service depending on
choice,
   * if it doesn't exist currently in the organization currentDonations, it adds it with 0 quantity.
   */
  private static RequestDonationList getAvailabeDonationsOfType(int choice){
    var entityListOneTypeOnly = Organization.getEntityListCopy();
    var availabeRDOfType = Organization.getCurrentDonationsCopy();

    Iterator<RequestDonation> rDIter = availabeRDOfType.getRdEntities().iterator();
    Iterator<Entity> entIter = entityListOneTypeOnly.iterator();

    switch(choice){
        case 1:
          //Get a copy of all donations, remove everything that's not a material
          while(rDIter.hasNext()){
            Entity ent = rDIter.next().getEntity();
            if (ent.getDetails().equals("Service")) rDIter.remove();
          }

          //next, do the same but for the entities
          while(entIter.hasNext()){
            Entity ent = entIter.next();
            if (ent.getDetails().equals("Service")){
              entIter.remove();
              continue;
            }

            //If entity doesn't exist as RD in availabe materials, add it with 0 quant.
            if(!availabeRDOfType.isInRequestDonationList(ent.getId())){
              availabeRDOfType.add(new RequestDonation(ent, 0));
            }
          }
          return availabeRDOfType;

        case 2:
          //Get a copy of all donations, remove everything that's not a service
          while(rDIter.hasNext()){
            Entity ent = rDIter.next().getEntity();
            if ( !(ent.getDetails().equals("Service"))) rDIter.remove();
          }

          //next, do the same but for the entities
          while(entIter.hasNext()){
            Entity ent = entIter.next();
            if ( !(ent.getDetails().equals("Service"))) {
              entIter.remove();
              continue;
            }


            //If entity doesn't exist as RD in availabe services, add it with 0 quant.
            if(!availabeRDOfType.isInRequestDonationList(ent.getId())){
              availabeRDOfType.add(new RequestDonation(ent, 0));
            }
          }
        return availabeRDOfType;
      }
```

```
        //In case of total failure, should never happen.
        return null;
    }
}
```

MenuInputException.java

```
public class MenuInputException extends Exception {

    //If user selects an invalid option, throw this exception
    public MenuInputException(){
        super("Please input a valid choice");
    }
}
```

NegativeQuantityException.java

```
public class NegativeQuantityException extends Exception {

    //If user attempts to input negative quantity for an entity, throw this exception
    public NegativeQuantityException() {
        super("You've entered a negative value");
    }
}
```

Offers.java

```
import java.util.Iterator;
public class Offers extends RequestDonationList {

    public void commit() {
        Iterator<RequestDonation> rdEntitiesIterator = rdEntities.iterator();
        if (rdEntities.size() == 0){
            System.out.println("No offers to be commited");
            return;
        }
        while(rdEntitiesIterator.hasNext()){
            var rd = rdEntitiesIterator.next();

            //This check happens because of the way it is implemented in main
            if (rd.getQuantity() != 0) {
                Organization.addToDonations(rd);
                rdEntitiesIterator.remove();
            }
        }
        System.out.println("The pending requests have been committed successfully");
    }
}
```

Organization.java

```
import java.util.*;

public class Organization
{
   //Ola einai static epeidh uparxei panta mono enas organismos sto programma, opote xreiaz
ontai mono mia fora ta pedia tou.
   private static String name;
   private static Admin admin;
   private static List<Entity> entityList = new ArrayList<Entity>();
   private static List<Donator> donatorList = new ArrayList<Donator>();
   private static List<Beneficiary> beneficiaryList = new ArrayList<Beneficiary>();
   private static RequestDonationList currentDonations = new RequestDonationList();

   public Organization(String name) {
      Organization.name = name;
   }

   public void setAdmin(Admin admin) {
      Organization.admin = admin;
   }

   public Admin getAdmin() {
      return admin;
   }

   public static String getName() {
      return Organization.name;
   }

   public static void printAdmin() {
      System.out.println(admin.name + ", " + admin.phone + " (" + Organization.name + ")");
      System.out.println("    You're the admin!");
   }

   //Epistrefei anafora
   public static List<Entity> getEntityList() {
      return Organization.entityList;
   }

   //Epistrefei antigrafo
   public static List<Entity> getEntityListCopy() {
      return new ArrayList<Entity>(Organization.entityList);
```

```
    }

    //Epistrefei antigrafo
    public static RequestDonationList getCurrentDonationsCopy() {
        RequestDonationList copy = new RequestDonationList(currentDonations);
        return copy;
    }

    //Wrappers
    public static void addToDonations(RequestDonation rd) {
        currentDonations.add(rd);
    }

    public static void subtractFromDonations(RequestDonation rd) {
        RequestDonation targetRD = currentDonations.get(rd.getEntity().getId());

        currentDonations.modify(rd, targetRD.getQuantity() - rd.getQuantity());
    }

    public static double getRDQuantity(int id) {
        return Organization.currentDonations.getRDQuantity(id);
    }

    public static void entityAlreadyExists(Entity entityToSearchFor) throws EntityIDAlreadyExis
tsException{
        for (Entity entity : entityList) {
            if (entity.getId() == entityToSearchFor.getId()){
                throw new EntityIDAlreadyExistsException(entityToSearchFor);
            }
        }
    }

        //MODIFY METHODS

    public static void addEntity(Entity entity) {
        try {
            //Check if entity already exists, if it does it will throw an exception and it won't be adde
d.
            entityAlreadyExists(entity);
            entityList.add(entity);
        } catch (EntityIDAlreadyExistsException EIDAEe) {
            System.out.println(EIDAEe.getMessage());
        }

    }
    public static void removeEntity(Entity entity) {
        entityList.remove(entity);
    }

    public static void insertDonator(Donator donator) {
        donatorList.add(donator);
    }

    public static void removeDonator(int index) {
        donatorList.remove(index);
    }

    public static void findAndPrintDonator(String phone) {
        for(Donator donator : donatorList) {
            if(donator.phone.equals(phone)) {
                System.out.println(donator.name + ", " + donator.phone + " (" + Organization.name
+ ")");
                System.out.println("    You're a donator!");
            }
        }
    }

    public static Donator findDonator(String phone) {
        for(Donator donator : donatorList) {
            if(donator.phone.equals(phone)) {
                return donator;
            }
        }
        return null;
    }

    public static Beneficiary findBeneficiary(String phone) {
        for(Beneficiary beneficiary : beneficiaryList) {
            if(beneficiary.phone.equals(phone)) {
                return beneficiary;
            }
        }
        return null;
    }

    public static void insertBeneficiary(Beneficiary beneficiary) {
        beneficiaryList.add(beneficiary);
    }
    public static void removeBeneficiary(int index) {
        beneficiaryList.remove(index);
    }

    public static void findAndPrintBeneficiary(String phone) {
        for(Beneficiary beneficiary : beneficiaryList) {
            if(beneficiary.phone.equals(phone)) {
                System.out.println(beneficiary.name + ", " + beneficiary.phone + " (" + Organization.
name + ")");
                System.out.println("  You're a beneficiary!");
            }
        }
    }

    public static void clearBeneficiaryReceivedList(int indexOfBeneficiary) throws EmptyListExc
eption {
        if (beneficiaryList.get(indexOfBeneficiary).getReceivedList().getRdEntities().isEmpty() == t
rue) {
            throw new EmptyListException();
        }

        beneficiaryList.get(indexOfBeneficiary).clearReceivedList();
    }

    /**
     * @return true if list was not already empty else false
     */
    public static boolean clearBeneficiaryReceivedListNoMessage(int indexOfBeneficiary){
        return beneficiaryList.get(indexOfBeneficiary).clearReceivedListNoMessage();
    }

    public static void listEntities() {
        for (var entity: entityList)
            System.out.println(entity);
    }

    public static void listBeneficiaries() {
        int count = 1;
        for (var beneficiary: beneficiaryList) {
            System.out.println("║  " + count + "." + beneficiary);
            count++;
        }
    }

    public static void listDonators() {
```

```java
        int count = 1;
        for (var donator: donatorList) {
            System.out.println("║ " + count + "." + donator);
            count++;
        }
    }

    public static int totalBeneficiaries(){
        return beneficiaryList.size();
    }

    public static int totalDonators(){
        return donatorList.size();
    }

    /**
     * Searches phone number
     * @return
     * 1 if beneficiary,
     * 2 if donator,
     * 3 if admin,
     * 0 if new
     */
    public static int searchPhone (String Phone) {
        for (var user: beneficiaryList) {
            if (Phone.equals(user.phone)) {
                return 1;
            }
        }

        for (var user: donatorList) {
            if (Phone.equals(user.phone)) {
                return 2;
            }
        }

        if (Phone.equals(admin.phone)) {
            return 3;
        }

        return 0;
    }
}
```
RequestDonation.java

```
import java.util.Comparator;

public class RequestDonation implements Comparator{

   private Entity entity;
   private double quantity;

   RequestDonation(Entity entity, double quantity) {
      this.entity = entity;
      this.quantity = quantity;
   }


   //Getters/Setters
   public Entity getEntity() {
      return entity;
   }

   public void setEntity(Entity entity) {
      this.entity = entity;
   }

   public double getQuantity() {
      return quantity;
   }

   public void setQuantity(double quantity) {
      this.quantity = quantity;
   }

   //Overwrites
   @Override
   public int compare(Object o1, Object o2) {
      return 0;
   }

   @Override
   public boolean equals(Object obj) {
      //Auto-generated method stub
      //return super.equals(obj);

      if(this.entity.getId()==((RequestDonation) obj).entity.getId()){
         return true;
      } else {
         return false;
```

```
        }
    }
}
```

RequestDonationList.java

```
import java.util.*;

public class RequestDonationList {
    protected List<RequestDonation> rdEntities;


    RequestDonationList() {
        rdEntities = new ArrayList<RequestDonation>();
    }

    /**
     * This constructor is used to make copies
     */

    RequestDonationList(RequestDonationList original) {
        this.rdEntities = new ArrayList<RequestDonation>(original.rdEntities);
    }

    //Getters/Setters
    public List<RequestDonation> getRdEntities() {
        return rdEntities;
    }

    public RequestDonation get(int id){
        /**
         * Searches through the list and returns the requestdonation
         * object with matching id
         */
        for (var rdEntity : rdEntities){
            if(id == rdEntity.getEntity().getId()){
                return rdEntity;
            }
        }
        return null;
    }

    public RequestDonation getWithIndex(int index){
        return rdEntities.get(index);
    }
```

```
  /**
   * Me mia prospelash ths rdEntities, elegxei an to eidos me to id einai stis diathesimes prosf
ores tou organismou
   * @return True an to eidos einai sthn rdEntities
   */
  public boolean isInRequestDonationList(int id) {
    for (RequestDonation rdEntity : rdEntities) {
      if (rdEntity.getEntity().getId() == id) {
        return true;
      }
    }

    return false;
    }

  //Adds new request
  public void add(RequestDonation rdToAdd) {
    /*
     * Loop through all RequestDonations in rdEntities, if a match is found
     * add the quantity of rdToAdd to the pre-existing one, else add the
     * new RequestDonation
     */

    for (RequestDonation rdEntity : rdEntities){
      if(rdToAdd.equals(rdEntity)){
        rdEntity.setQuantity(rdEntity.getQuantity() + rdToAdd.getQuantity());
        return;
      }
    }
    rdEntities.add(rdToAdd);
  }

  /**
  *  @apiNote  needed in Requests.
  */
  public void add(Beneficiary beneficiary, RequestDonation rdToAdd){}

  public void remove(RequestDonation rdToRemove){
    //Loop through all RequestDonations in rdEntities, if a match is found, delete it.
    for (RequestDonation rdEntity : rdEntities){
      if(rdToRemove.equals(rdEntity)){
        rdEntities.remove(rdEntity);
        return;
      }
    }
    }

    /**
     *
     * @param rdToModify RequestDonation object to search for in list
     * @param quantity new quantity
     * quantity based on entity
     */
    public void modify(RequestDonation rdToModify, double quantity){
        //Searches for RD of argument and if a match is found, modifies quantity of it in arraylist.
        for (RequestDonation rdEntity : rdEntities){
            if(rdToModify.equals(rdEntity)){
                rdEntity.setQuantity(quantity);
                return;
            }
        }
    }

    /**
    *  @apiNote  needed in Requests.
    */
    public void modify(Beneficiary beneficiary ,RequestDonation rdToModify, double quantity
){}

    /**
     * Emfanizei to sunolo twn eidwn ths listas rdEntities me arithmish.
     */
    public void monitor(){
        int count = 1;
        System.out.println("╔" + "═".repeat(49));
        for (RequestDonation rdEntity : rdEntities){
            System.out.print ("║ " + count + ".");
            System.out.println(rdEntity.getEntity().getName() + " (" + rdEntity.getQuantity() + ")");
            count++;
        }
    }

    public double getRDQuantity(int id) {
        if(get(id) == null) return 0;
        return get(id).getQuantity();
    }

    //Removes everything from the list
    public void reset() throws EmptyListException {
        if (rdEntities.size()==0) throw new EmptyListException();
```

```java
        rdEntities.clear();
    }

}
```

RequestedQtyExceedsAllowedException.java

```java
public class RequestedQtyExceedsAllowedException extends Exception{
    //If user attempts to request more qty than allowed by organization, throw this exception
    public RequestedQtyExceedsAllowedException(RequestDonation rD) {
        super("The requested quantity of " + rD.getEntity().getName() + " exceeds the "
        + "allowed limits set from the organization for this beneficiary and was not applied.");
    }
}
```

RequestedQtyExceedsAvailabeException.java

```java
public class RequestedQtyExceedsAvailabeException extends Exception {
    //If user attempts to request more quantity than the organization has, throw this exception
    public RequestedQtyExceedsAvailabeException(RequestDonation rD) {
        super("The requested quantity of " + rD.getEntity().getName() + " is not availabe from "
        + "the organisation and it was not applied.");
    }
}
```

Requests.java

```java
import java.util.Iterator;

public class Requests extends RequestDonationList {

    @Override
    public void add(Beneficiary beneficiary, RequestDonation rdToAdd){
        try {
            //We don't care what this returns, just need to catch the exceptions thrown.
            totalCheck(beneficiary, rdToAdd);
            super.add(rdToAdd);
        } catch (RequestedQtyExceedsAvailabeException RQEAVe) {
            System.out.println(RQEAVe.getMessage());
        } catch (RequestedQtyExceedsAllowedException RQEALe) {
            System.out.println(RQEALe.getMessage());
        }



    }
    @Override
    public void modify(Beneficiary beneficiary, RequestDonation rdToModify, double quantity
){
        //double oldQty = rdToModify.getQuantity();
        try {
            //Create a copy of the given rd it's not affected
            var rdCopy = new RequestDonation(rdToModify.getEntity(), quantity);
            //We don't care what this returns, just need to catch the exceptions thrown.
            totalCheck(beneficiary, rdCopy);

        } catch (RequestedQtyExceedsAvailabeException RQEAVe) {
            System.out.println(RQEAVe.getMessage());
            quantity = rdToModify.getQuantity();
        } catch (RequestedQtyExceedsAllowedException RQEALe) {
            System.out.println(RQEALe.getMessage());
            quantity = rdToModify.getQuantity();
        } finally {
            super.modify(rdToModify, quantity);
        }


    }

    //For notes on variables, look at cmpRequestQuantitiesWithMaterialLevels method below.
    private boolean validRequestDonation(Beneficiary beneficiary, RequestDonation rq) thro
ws RequestedQtyExceedsAllowedException {
        if(rq.getEntity().getDetails().equals("Service")){
            return true;
        } else {
            if(beneficiary.getNoPersons() == 1){
                boolean  cmpResults = cmpRequestQuantitiesWithMaterialLevels(beneficiary, rq, 1);

                if(!cmpResults){
                    throw new RequestedQtyExceedsAllowedException(rq);
                }
            } else if (beneficiary.getNoPersons() >= 2 && beneficiary.getNoPersons() <= 5){
                boolean  cmpResults = cmpRequestQuantitiesWithMaterialLevels(beneficiary, rq, 2);
                if(!cmpResults){
                    throw new RequestedQtyExceedsAllowedException(rq);
                }
            } else if (beneficiary.getNoPersons() >= 5){
                boolean  cmpResults = cmpRequestQuantitiesWithMaterialLevels(beneficiary, rq, 3);
                if(!cmpResults){
                    throw new RequestedQtyExceedsAllowedException(rq);
                }
            }
            //If no exception was thrown, then request is valid, return true.
            return true;
        }
    }

    /**
     * Id: Id of entity
     * qty: qty that beneficiary already has received
     * rqQty: requested additional quantity
     * @return True if LEVEL quantity > already received + current request quantity.
     * False if else.
     */
    private boolean cmpRequestQuantitiesWithMaterialLevels(Beneficiary beneficiary, Reques
tDonation rq, int level) {
        //Get entity id and quantity for request, then compare them to the material level.
        int id = rq.getEntity().getId();
        double rqQty = rq.getQuantity();
        double qty;
        double levelQty = ((Material) rq.getEntity()).getLevelQuantity(level);

        if(beneficiary.getReceivedList().get(id) == null){
            qty = 0;
        } else {
            qty = beneficiary.getReceivedList().get(id).getQuantity();
        }

        if(levelQty >= qty + rqQty){
            return true;
        }
        return false;
    }

    private void sufficientQtyCheck(RequestDonation rD) throws RequestedQtyExceedsAvaila
beException {
        if (rD.getQuantity() > Organization.getRDQuantity(rD.getEntity().getId())){
            throw new RequestedQtyExceedsAvailabeException(rD);
        }
    }

    /**
     * @param beneficiary beneficiary in question
     * @param rD RequestDonation object to be checked.
     * Combines both quantity checks into one for ease of use.
     */
```

```
private void totalCheck(Beneficiary beneficiary, RequestDonation rD) throws
RequestedQtyExceedsAvailabeException, RequestedQtyExceedsAllowedException {

  try {
    validRequestDonation(beneficiary, rD);
    sufficientQtyCheck(rD);
  } catch (RequestedQtyExceedsAvailabeException RQEAVe) {
    throw RQEAVe;
  } catch (RequestedQtyExceedsAllowedException RQEALe) {
    throw RQEALe;
  }

}

// Recheck for exceptions because you could add two different times such that
// qt1+qt2>allowed/availabe, while not being over the limit themselves.

public void commit(String phone) {
  Iterator<RequestDonation> requestRDEntitiesIterator = rdEntities.iterator();
  Beneficiary beneficiary = Organization.findBeneficiary(phone);

  if (rdEntities.size()==0) {
    System.out.println("No requests to be commited");
    return;
  }

  while(requestRDEntitiesIterator.hasNext()){
    try{
      var requestRD = requestRDEntitiesIterator.next();

      //This throws exceptions, so you only need to call it.
      totalCheck(beneficiary, requestRD);

      //This check happens because of the way it is implemented in main
      if (requestRD.getQuantity() != 0) {
        Organization.subtractFromDonations(requestRD);
        beneficiary.addToReceivedList(requestRD);
        requestRDEntitiesIterator.remove();
      }

    } catch (RequestedQtyExceedsAvailabeException RQEAVe) {
      System.out.println(RQEAVe.getMessage());
    } catch (RequestedQtyExceedsAllowedException RQEALe) {
      System.out.println(RQEALe.getMessage());
    }
```

```
        }
        System.out.println("The pending requests have been committed successfully");
    }
}
```

Service.java

```
class Service extends Entity
{
    Service(String name, String description, int id) {
        super(name, description, id);
    }

    public String getDetails()
    {
        return "Service";
    }



}
```

User.java

```
abstract public class User {

    protected String name, phone;

    protected User(String name, String phone) {
        this.name = name;
        this.phone = phone;
    }
}
```