ΕΡΓΑΣΙΑ ΟΝΤΟΚΕΝΤΡΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2020 - 2021

ΜΗΛΤΙΑΔΗΣ ΜΑΝΤΕΣ
ΑΜ 1084661
E - MAIL up1084661@upnet.gr

ΦΙΛΙΠΠΟΣ – ΠΑΡΑΣΚΕΥΑΣ ΖΥΓΟΥΡΗΣ
ΑΜ 1084660
E - MAIL up1084660@upnet.gr

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΤΜΗΜΑ ΜΗΧ Η / Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

08. 06. 2021

ΕΡΓΑΣΙΑ ΟΝΤΟΚΕΝΤΡΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2020 - 2021

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΚΟΠΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

Η δημιουργία ενός ηλεκτρονικού συστήματος οργάνωσης εθελοντών-δωρητών, το οποίο θα βρίσκει εφαρμογή στην εποχή της πανδημίας της COVID – 19 με στόχο τη συγκέντρωση ειδών από δωρητές (**Donators**) και αιτημάτων από επωφελούμενους (**Beneficiaries**). Το σύστημα θα φέρει το όνομα **donation** και θα πρέπει να υποστηρίζει διαχείριση χρηστών, διαχείριση ειδών, καταχώρηση και εξυπηρέτηση αιτημάτων για τα είδη.

ΑΝΑΛΥΣΗ ΟΡΓΑΝΩΣΗΣ ΑΝΑΦΟΡΑΣ

ΕΝΟΤΗΤΑ 1 :

Στην πρώτη ενότητα αναπτύσσονται με **σύντομο τρόπο** δύο (2) αντικείμενα :

1. Οι σχέσεις κληρονομικότητας ανάμεσα στις κλάσεις καθώς και τα μέλη αυτών (δηλαδή πεδία και μέθοδοι) όπως ακριβώς εμφανίζονται και στον τελικό κώδικα.
2. Τα επιπρόσθετα στοιχεία που ξεφεύγουν από το εύρος της εκφώνησης, αλλά κατά την άποψή μας καθιστούν ολόκληρο το πρόγραμμα πιο λειτουργικό και εύκολα χρησιμοποιήσιμο από τον χρήστη.

Στο τέλος, παρατίθεται το UML διάγραμμα για ολόκληρο το πρόγραμμα, ωστόσο και από πριν έχουμε παραθέσει τη σύνδεση όσων κλάσεων κληρονομούν από άλλες.

ΕΝΟΤΗΤΑ 2 :

Στην δεύτερη ενότητα αναπτύσσονται με **αναλυτικό τρόπο** τα εξής δύο (2) ζητούμενα για κάθε κλάση :

1. Επιλογή των αναγνωριστικών για πεδία, κατασκευαστές και μεθόδους.
2. Αιτιολόγηση της σύνταξης του κώδικα για κάθε μέθοδο, κατασκευαστή.

ΕΝΟΤΗΤΑ 3 :

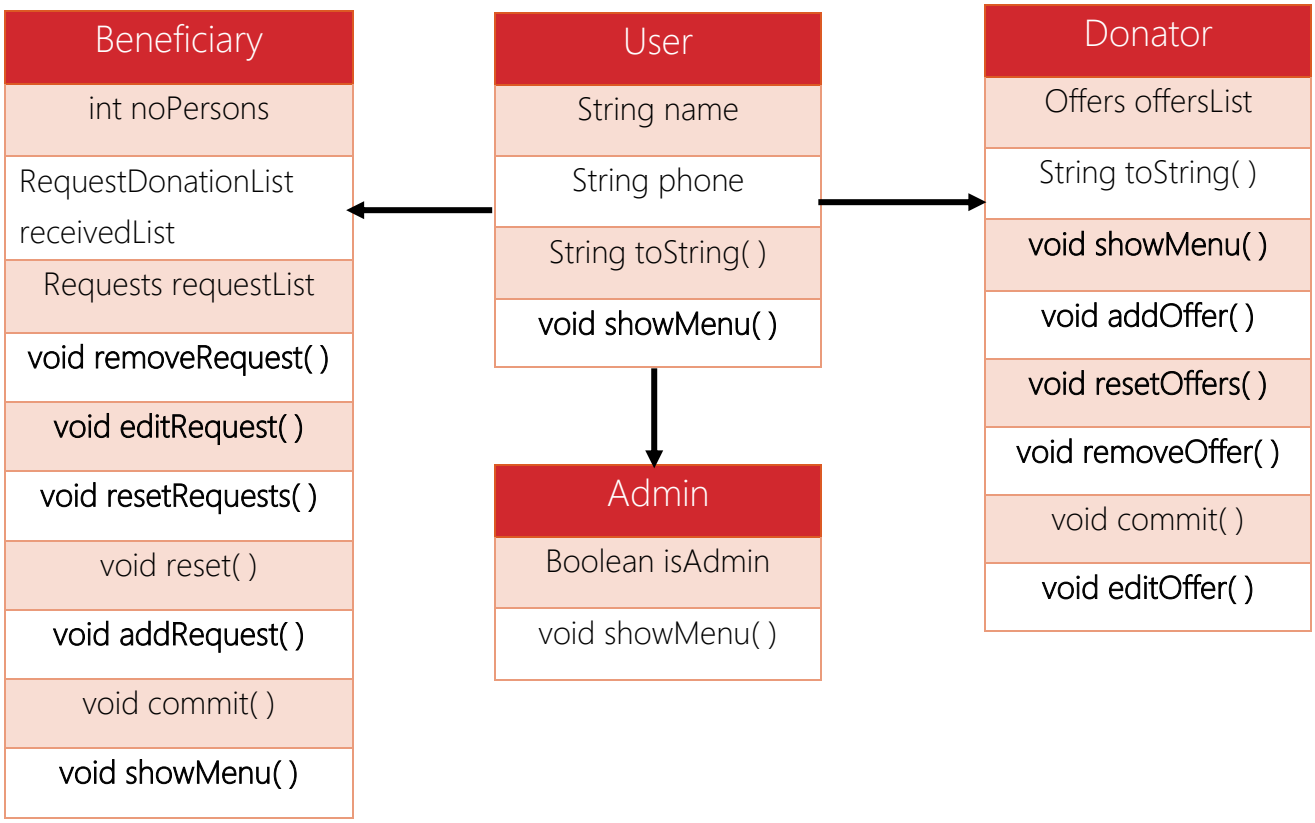
Στην τρίτη ενότητα – παράρτημα παρουσιάζεται ο συνολικός κώδικας για το συγκεκριμένο project, ενώ ακόμα παρατίθεται και κατάλληλος σύνδεσμος που παραπέμπει στο επιλεγμένο private code repository του **GitHub**. Εκεί θα υπάρχει ένας φάκελος ***Project – code*** που θα περιλαμβάνει το αρχείο του κώδικα για κάθε κλάση.

- **Για την συγγραφή και την εκτέλεση του κώδικα έχει επιλεγεί το εργαλείο BlueJ.**

1 ΔΙΑΓΡΑΜΜΑ UML

1.1 ΑΝΑΛΥΣΗ ΔΙΑΣΥΝΔΕΣΗΣ ΚΛΑΣΕΩΝ

A. CLASS USER *

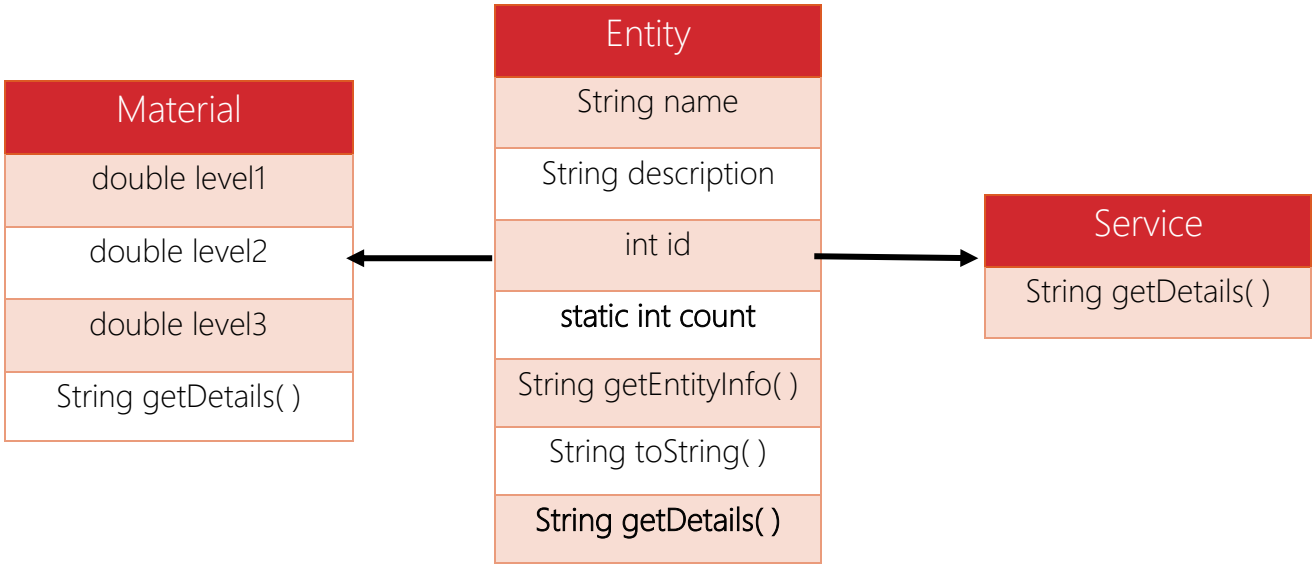


ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *User* αρχικά ορίζεται ως abstract, δηλαδή ο γενικός της σκοπός είναι να περιγράφει τη λειτουργία και το ρόλο κάθε χρήστη μέσα στο σύστημα donation. Υπάρχουν τρία (3) είδη χρήστη :
   i. Διαχειριστής – *Admin* : ορίζει τα είδη που μπορούν να παρέχονται από τον οργανισμό.
   ii. Δωρητής – *Donator* : μπορεί να προσφέρει είδη από τη συγκεκριμένη λίστα ειδών του οργανισμού.
   iii. Επωφελούμενος – *Beneficiary* : μπορεί να ζητήσει να του δοθούν είδη από τα παρεχόμενα από τον οργανισμό.
2. Τα τρία (3) είδη χρήστη στην ουσία αποτελούν τρεις ανεξάρτητες abstract κλάσεις που κληρονομούν από την *User*. Έτσι, μπορούν να δημιουργηθούν μόνο στιγμιότυπα των υποκλάσεων λόγω της abstract φύσης τους, δηλαδή είτε χρήστης – διαχειριστής, είτε χρήστης – δωρητής είτε χρήστης - επωφελούμενος.

3. Κάθε είδος χρήστη περιλαμβάνει δύο (2) βασικά χαρακτηριστικά που κληρονομεί από την πατρική κλάση :
   i. Ένα αλφαριθμητικό – String *name* που αναπαριστά το όνομα του.
   ii. Ένα αλφαριθμητικό – String *phone* που αναπαριστά το κινητό τηλέφωνο του χρήστη ως έναν πίνακα χαρακτήρων από αριθμούς ('0' – '9').

B. CLASS ENTITY



ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *Entity* αρχικά ορίζεται επίσης ως abstract, δηλαδή ο γενικός της σκοπός είναι να περιγράφει τα είδη και τις ενέργειες που μπορεί να πραγματοποιηθούν πάνω στις παροχές από κάθε χρήστη. Υπάρχουν δύο (2) είδη παροχών :
   i. Υλικές παροχές – *Material* : είναι τα υλικά είδη που προσφέρονται στον Επωφελούμενο.
   ii. Υπηρεσίες – *Service* : είναι οι υπηρεσίες που προσφέρονται στον Επωφελούμενο.
2. Τα δύο (2) είδη παροχών στην ουσία αποτελούν δύο ανεξάρτητες abstract κλάσεις που κληρονομούν από την *Entity*. Έτσι, μπορούν να δημιουργηθούν μόνο στιγμιότυπα των υποκλάσεων λόγω της abstract φύσης τους, δηλαδή είτε υλικές προσφορές είτε υπηρεσίες.
3. Κάθε είδος παροχής περιλαμβάνει τρία (3) βασικά χαρακτηριστικά που κληρονομεί από την πατρική κλάση :
   i. Ένα αλφαριθμητικό – String *name* που αναπαριστά το όνομα του.

ii. Ένα αλφαριθμητικό – String *description* που αναπαριστά μια σύντομη περιγραφή κάθε παροχής.
iii. Έναν ακέραιο – int *id* που αναπαριστά το αναγνωριστικό της κάθε παροχής.

4. Κάθε είδος υλικής παροχής – *Material* περιλαμβάνει τρία (3) επιπλέον χαρακτηριστικά που αφορούν αποκλειστικά αυτό :
   i. Έναν ακέραιο – int *level1* που αναπαριστά τη ποσότητα της υλικής παροχής που αντιστοιχεί σε ένα άτομο.
   ii. Έναν ακέραιο – int *level2* που αναπαριστά τη ποσότητα της υλικής παροχής που αντιστοιχεί σε δύο έως τέσσερα άτομα.
   iii. Έναν ακέραιο – int *level3* που αναπαριστά τη ποσότητα της υλικής παροχής που αντιστοιχεί σε πέντε άτομα και πάνω.

C. CLASS REQUESTDONATION *

RequestDonation
Entity entity
double quantity
int id
String toString( )
int compare( )

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *RequestDonation* έχει ως γενικό της σκοπό να περιγράφει το σύνολο των αιτημάτων δωρεών ή παροχών από τον χρήστη μέσα στο σύστημα donation για μια συγκεκριμένη παροχή.
2. Η κλάση αυτή δεν έχει περιορισμό και μπορεί να δημιουργεί στιγμιότυπα, δηλαδή αιτήματα για κάθε δωρεά – *Entity* που διατίθεται στον οργανισμό.
3. Κάθε αίτημα δωρεάς ή παροχή περιλαμβάνει δύο (2) βασικά χαρακτηριστικά :
   i. Ένα αντικείμενο αναφοράς τύπου *Entity* που προσδιορίζει για ποιο στιγμιότυπο της παραπάνω κλάσης εγείρεται αίτημα δωρεάς ή παροχής.
   ii. Έναν ακέραιο – int *quantity* που αναπαριστά το αίτημα για συγκεκριμένη ποσότητα ενός *Entity*.

D. CLASS ORGANIZATION *

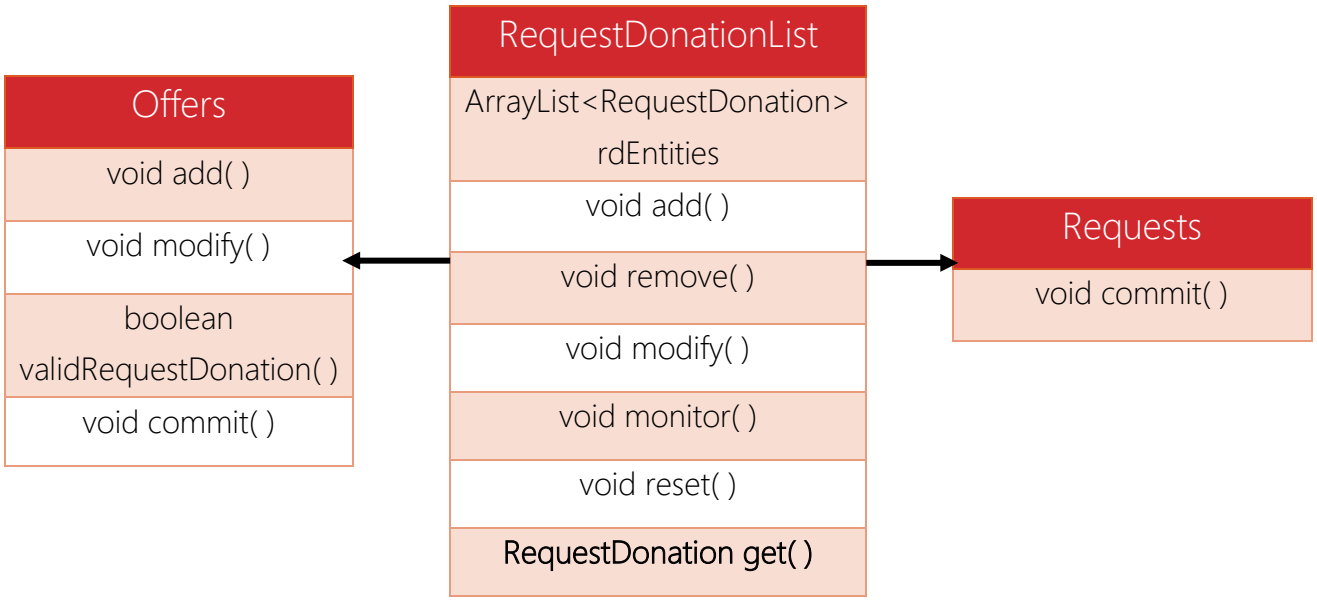
Organization
String name
Admin admin
int id
ArrayList<Entity> entityList
ArrayList<Donator> donatorList
ArrayList<Beneficiary> beneficiaryList
RequestDonationList currentDonations
boolean hasEnough( )
addEntity( )
removeEntity( )
insertDonator( )
removeDonator( )
insertBeneficiary( )
removeBeneficiary ( )
listEntities( )
listBeneficiaries( )
listDonators( )
void printListedServices( )
void printListedMaterials( )
User findUser( )

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *Organization* έχει ως γενικό της σκοπό να περιγράφει τον οργανισμό που διαχειρίζεται ο χρήστης – διαχειριστής μέσα στο σύστημα donation.
2. Η κλάση αυτή δεν έχει περιορισμό και μπορεί να δημιουργεί στιγμιότυπα, δηλαδή οργανισμούς που διατίθενται στο σύστημα donation για δωρεά παροχών.

3. Κάθε οργανισμός περιλαμβάνει έξι (6) βασικά χαρακτηριστικά :
    i. Ένα αλφαριθμητικό – String *name* που αναπαριστά το όνομα του οργανισμού.
    ii. Ένα αντικείμενο admin τύπου *Admin* που αναπαριστά τον χρήστη – διαχειριστή που ελέγχει τον συγκεκριμένο οργανισμό.
    iii. Μια λίστα – ArrayList *entityList* που αναπαριστά το σύνολο των ειδών που μπορούν να διανεμηθούν σε χρήστες – επωφελούμενους.
    iv. Μια λίστα – ArrayList *donatorList* που αναπαριστά το σύνολο των χρηστών – δωρητών.
    v. Μια λίστα – ArrayList *beneficiaryList* που αναπαριστά το σύνολο των χρηστών – επωφελούμενων.
    vi. Ένα αντικείμενο currentDonations τύπου *RequestDonationList* που αναπαριστά τις διαθέσιμες εκείνη τη στιγμή προσφορές ειδών και τις ποσότητες αυτών.
4. Το Organization είναι αυτό που έχει τις συναρτήσεις εκτύπωσης της λίστας των χρηστών ανάλογα τον ρόλο τους στο σύστημα donation.

E. CLASS REQUESTDONATIONLIST *



ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *RequestDonationList* έχει ως γενικό της σκοπό να περιγράφει ένα σύνολο από αιτήματα δωρεών ή παροχών διάφορων ειδών από χρήστες μέσα στο σύστημα donation. Υπάρχουν δύο (2) είδη λιστών :

i. **Λίστα αιτημάτων χρήστη – επωφελούμενου – Requests** : ορίζει τα είδη που ζητά να λάβει ένας χρήστης – επωφελούμενος από τα ήδη διαθέσιμα.
ii. **Λίστα παροχών χρήστη – δωρητή – Offers** : ορίζει τα είδη που ζητά να δώσει ένας χρήστης – δωρητής από τα ήδη διαθέσιμα.

2. Η κλάση αυτή δεν έχει περιορισμό και μπορεί να δημιουργεί στιγμιότυπα, δηλαδή λίστες από αιτήματα για κάθε χρήστη πάνω σε συγκεκριμένα είδη που επιθυμεί ή λίστες από αιτήματα δωρεάς ειδών για κάθε χρήστη που επιθυμεί να δωρίσει.
3. Κάθε περιλαμβάνει ένα (1) βασικό χαρακτηριστικό :
   i. Μια λίστα – ArrayList *rdEntities* που αναπαριστά το σύνολο των αιτημάτων που μπορούν να εγερθούν από τους χρήστες για μια συγκεκριμένη παροχή.

F. CLASS MENU *

Menu
Organization org
String scanForName( )
String scanForPhone( )
User completeRegister( )
void greet( )
int scanNum( )
int getMenuOption( )
void adminViewMenu( )
void adminMonitorMenu( )
void adminMenu( )
void beneficiaryRequest( )
void beneficiaryAddRequest( )
void beneficiaryShowRequests( )
void beneficiaryCommit( )

void beneficiaryMenu( )
void donatorAddOffer( )
void donatorShowOffers( )
void donatorCommit( )
void donatorMenu( )
void initMenu( )

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση *Menu* έχει όλη την επικοινωνία με τον χρήστη κάθε οργανισμού και λαμβάνει το input του, ενώ τυπώνεται διαφορετικό menu αναλόγως το ποιος χρήστης έχει συνδεθεί.
2. Υπάρχουν τρία (3) είδη menu, ένα για κάθε χρήστη, τα οποία είναι :
   i. **adminMenu** – αλληλεπίδραση μόνο με χρήστη – διαχειριστή.
   ii. **donatorMenu** – αλληλεπίδραση μόνο με χρήστη – δωρητή.
   iii. **beneficiaryMenu** – αλληλεπίδραση μόνο με χρήστη – επωφελούμενο.

G. CLASS EXCEPTIONERROR

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η κλάση **ExceptionError** απλά καλείται για τα errors που μπορεί να προκύψουν κατά το τρέξιμο του κώδικα και απλά στον constructor της τυπώνει το μήνυμα λάθους που λαμβάνει ως όρισμα.

H. CLASS MAIN *

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1. Η βασική κλάση *Main* έχει ως γενικό της σκοπό το τρέξιμο όλου του προγράμματος, δηλαδή περιέχει τη συνάρτηση **main( )**, που τρέχει πρώτη στο πρόγραμμα, στην οποία δεν γίνεται κάτι άλλο πέραν του να δημιουργηθούν κάποια αντικείμενα και μετά να κληθεί η κλάση *Menu*.

---

a. *Σε όσες κλάσεις δίπλα στο όνομά τους βρίσκεται το σύμβολο* ***** *σημαίνει ότι επικοινωνούν και με άλλες κλάσεις με τις οποίες δεν σχετίζονται με κάποια σχέση κληρονομικότητας, αλλά χρησιμοποιούν στοιχεία τους για την υλοποίηση μεθόδων στο σώμα τους. Στο τελικό UML διάγραμμα η σχέση αυτή εκφράζεται με διακεκομμένα βέλη τα οποία «δείχνουν» στην κλάση / κλάσεις που χρησιμοποιούνται.*

b. *Τα στοιχεία κάθε κλάσης που φαίνονται παραπάνω, τα οποία είναι με έντονη γραφή (**bold**), είναι πρόσθετα στοιχεία που δεν ζητούνται στην εκφώνηση και θα αναλυθούν στην υποενότητα 1.2.*

1.2 ΕΠΕΞΗΓΗΣΗ ΠΡΟΣΘΕΤΩΝ ΣΤΟΙΧΕΙΩΝ ΑΝΑ ΚΛΑΣΗ

A. CLASS USER

1. **void showMenu( )** : μια abstract μέθοδος που υλοποιείται διαφορετικά σε κάθε υποκλάση της ***User*** προκειμένου να εκτυπώνεται το menu που αντιστοιχεί στον σωστό χρήστη κάθε φορά.
2. **void addRequest( ),void resetRequests( ),void editRequest( ),void removeRequest( )** : αυτές οι τέσσερις μέθοδοι στην ουσία περικλείουν (*wrap*) τις πιο σημαντικές μεθόδους της κλάσης ***Requests*** και έτσι ο χρήστης – επωφελούμενος μπορεί να τις καλεί και να επεξεργάζεται τα αιτήματα για τα είδη που θέλει να λάβει.
3. **void addOffer( ), void resetOffers( ), void editOffer( ), void removeOffer( )** : αυτές οι τέσσερις μέθοδοι στην ουσία περικλείουν (*wrap*) τις πιο σημαντικές μεθόδους της κλάσης ***Offers*** και έτσι ο χρήστης – δωρητής μπορεί να τις καλεί και να επεξεργάζεται τα είδη που θέλει να προσφέρει.

B. CLASS ENTITY

1. **static int count** : είναι ένα πεδίο που η τιμή του αφού οριστεί δεν μεταβάλλεται και στην ουσία κάθε φορά που δημιουργείται ένα αντικείμενο τύπου ***Entity*** η τιμή του αυξάνεται κατά 1 και εκχωρείται στο πεδίο **id**. Έτσι, τα **id** των ειδών είναι πιο οργανωμένα και βρίσκονται σε αύξουσα σειρά.
2. **String getDetails( ) :** μια abstract μέθοδος που υλοποιείται διαφορετικά σε κάθε υποκλάση της ***Entity*** προκειμένου να εκτυπώνονται τα στοιχεία που αντιστοιχούν στο σωστό είδος κάθε φορά.

C. CLASS REQUESTDONATION

1. **int compare( )** : η μέθοδος αυτή συγκρίνει το **id** του αντικειμένου που την καλεί με το **id** ενός άλλου αντικειμένου προκειμένου να διακρίνει αν αυτά είναι ίσα, δηλαδή αν αυτά τα δύο είδη της κλάσης ***RequestDonation*** θεωρούνται ταυτόσημα.

D. CLASS ORGANIZATION

1. **boolean hasEnough( )** : η μέθοδος αυτή ελέγχει αν το είδος της ***RequestDonation*** που ζητείται είναι διαθέσιμο και η ποσότητά του είναι επαρκής για να παραδοθεί στον χρήστη.

2. void printListedServices( ), void printListedMaterials( ) : αυτές οι μέθοδοι εκτυπώνουν τα διαθέσιμα είδη material και service της currentDonations μαζί με τις ποσότητές τους εκείνη τη στιγμή.
3. User findUser( ) : η μέθοδος αυτή ψάχνει στις donatorList και beneficiaryList για να αναγνωρίσει και να επιστρέψει το είδος του χρήστη. Αν ο χρήστης δεν ανήκει σε καμία λίστα, τότε ελέγχει αν είναι διαχειριστής.

E. CLASS REQUESTDONATIONLIST

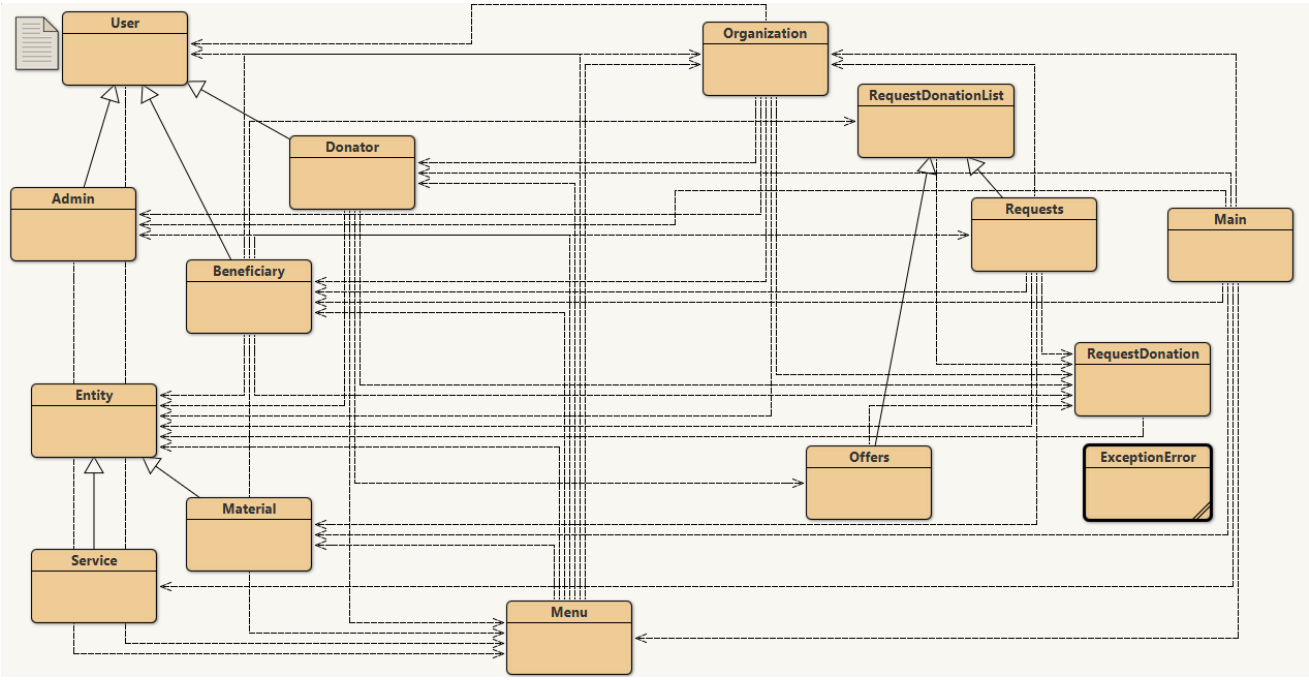
1. RequestDonation get( ) : η μέθοδος αυτή ψάχνει στη λίστα rdEntities για ένα συγκεκριμένο αντικείμενο τύπου *RequestDonation* με id που εισάγει ο χρήστης και άμα το βρει το επιστρέφει.

F. CLASS MENU

1. Organization org : είναι ένα πεδίο της κλάσης *Menu* που αντιστοιχίζει το menu με τον συγκεκριμένο οργανισμό.
2. String scanForName( ), String scanForPhone( ), User completeRegister( ) : αυτές οι μέθοδοι ζητάνε τα στοιχεία του χρήστη και μετά τη ταυτοποίηση του ζητούν να εισάγει τι ρόλο αναλαμβάνει μέσα στο σύστημα donation.
3. void greet( ) : αυτή η μέθοδος εμφανίζει μήνυμα χαιρετισμού στον χρήστη μόλις συνδέεται.
4. int getMenuOption( ) : αυτή η μέθοδος ορίζει τον αριθμό των επιλογών στο menu του αντίστοιχου χρήστη από τις οποίες αυτός καλείται να επιλέξει.
5. void adminMonitorMenu( ) : αυτή η μέθοδος αφορά ΜΟΝΟ τον χρήστη – διαχειριστή και εκτελεί τις διάφορες επιλογές που αφορούν τον χειρισμό του οργανισμού, όπως αναφέρονται και στην εκφώνηση.
6. void donatorAddOffer( ), void donatorShowOffers( ), void donatorCommit( ) : αυτές οι μέθοδοι αφορούν ΜΟΝΟ τον χρήστη – δωρητή και εκτελούν τις διάφορες επιλογές που αφορούν τον χειρισμό των ειδών που θέλει να προσφέρει, όπως αναφέρονται και στην εκφώνηση.
7. void beneficiaryAddRequest( ), void beneficiaryShowRequests( ), void beneficiaryCommit( ), void beneficiaryRequest( ) : αυτές οι μέθοδοι αφορούν ΜΟΝΟ τον χρήστη – επωφελούμενο και εκτελούν τις διάφορες επιλογές που αφορούν τον χειρισμό των αιτημάτων για είδη που θέλει να λάβει, όπως αναφέρονται και στην εκφώνηση.

8. void adminMenu( ), void donatorMenu( ), void beneficiaryMenu( ) : αυτές οι μέθοδοι εμφανίζουν ολόκληρο το menu με τις επιλογές κάθε χρήστη και ανάλογα με το τι θα διαλέξει καλούν και την αντίστοιχη μέθοδο που επιτελεί αυτή τη λειτουργία.
9. void initMenu( ) : αυτή η μέθοδος αρχικοποιεί το menu αφού πρώτα ελέγξει τα στοιχεία του χρήστη με τις μεθόδους που περιγράψαμε στο (2).

- Το ολοκληρωμένο UML διάγραμμα που αναπαριστά τη διασύνδεση όλων των κλάσεων φαίνεται πιο κάτω :



2 ΥΛΟΠΟΙΗΣΗ ΚΛΑΣΕΩΝ

2.1 User

- Με βάση την αρχή απόκρυψης των δεδομένων, όλα τα πεδία ορίζονται ως *private*.

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***User*** ορίζουμε *public* constructor με ορίσματα αλφαριθμητικά **name** και **phone**. Με τη λέξη – κλειδί this εκχωρούμε στα πεδία **name** και **phone** του στιγμιότυπου τις τιμές των ορισμάτων.

2. Methods

Αρχικά, ορίζουμε **getters** που επιστρέφουν τα περιεχόμενα των πεδίων της ***User***, ώστε να μπορούμε να τα επεξεργαστούμε και εκτός της κλάσης λόγω της *private* φύσης τους.

Έπειτα, ορίζουμε μια *abstract* μέθοδο **showMenu( )** με τύπο επιστροφής void, δίνοντας μόνο την υπογραφή της, δηλαδή χωρίς κάποιο σώμα, καθώς θα υλοποιείται διαφορετικά σε κάθε υποκλάση της ***User***. Η μέθοδος παίρνει σαν όρισμα ένα αντικείμενο **menu** τύπου ***Menu***, το οποίο θα είναι είτε το menu του χρήστη – διαχειριστή είτε το menu του χρήστη – δωρητή είτε το menu του χρήστη – επωφελούμενου .

Τέλος, ορίζουμε τη μέθοδο **toString( )** με τύπο επιστροφής String, η οποία κάνει @Override και επιστρέφει με μορφή αλφαριθμητικού το όνομα και τον αριθμό τηλεφώνου του χρήστη.

2.1.1 Admin

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***Admin*** ορίζουμε *public* constructor με ορίσματα πάλι αλφαριθμητικά **name** και **phone**. Με τη λέξη – κλειδί super καλούμε τον constructor της ***User*** με ορίσματα αυτά τα δύο και εκτελείται η αντίστοιχη διαδικασία που περιγράφτηκε παραπάνω. Ακόμα, θέτουμε την τιμή της **isAdmin** σε

true, έτσι ώστε κάθε φορά που δημιουργείται στιγμιότυπο της *Admin* να ξέρει το πρόγραμμα ότι αυτός ο χρήστης είναι διαχειριστής.

2. Methods

Υλοποιούμε την *abstract* μέθοδο showMenu( ) καλώντας στο όρισμά της menu τη μέθοδο adminMenu( ) της κλάσης *Menu* με όρισμα το συγκεκριμένο αντικείμενο της *Admin* χρησιμοποιώντας τη λέξη – κλειδί this. Έτσι, θα εκτυπώνεται στην οθόνη καλώντας τη showMenu( ) σε ένα αντικείμενο της *Admin* το menu που αντιστοιχεί στον χρήστη – διαχειριστή με τι κατάλληλες επιλογές.

2.1.2 Donator

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Donator* ορίζουμε *public* constructor με ορίσματα πάλι αλφαριθμητικά name και phone. Με τη λέξη – κλειδί super καλούμε τον constructor της *User* με ορίσματα αυτά τα δύο και εκτελείται η αντίστοιχη διαδικασία που περιγράφτηκε παραπάνω. Ακόμα, δημιουργούμε με την εντολή new ένα αντικείμενο offersList τύπου *Offers*, έτσι ώστε για κάθε χρήστη – δωρητή που δημιουργείται να του αντιστοιχίζεται αυτόματα μια λίστα από είδη που επιθυμεί να προσφέρει.

2. Methods

Αρχικά, κάνουμε wrap τις μεθόδους add( ), remove( ), reset( ) και modify( ) της κλάσης *Offers* δημιουργώντας τις αντίστοιχες addOffer( ), removeOffer( ), resetOffers( ) και editOffer( ) :

- addOffer( ) : έχει ορίσματα ένα αντικείμενο entity τύπου *Entity* και ένα double πεδίο quantity με τύπο επιστροφής void. Στο σώμα της δημιουργούμε ένα αντικείμενο rd της *RequestDonation* με τη λέξη – κλειδί new και το προσθέτουμε στη λίστα offersList με τη μέθοδο add( ) της *RequestDonationList*.
- removeOffer( ) : έχει ορίσμα ένα int πεδίο id με τύπο επιστροφής void. Στο σώμα της καλούμε στη λίστα offersList τη μέθοδο remove( ) της *RequestDonationList*, η οποία παίρνει σαν όρισμα το αντικείμενο που επιστρέφει η μέθοδος get( ) της *RequestDonationList* με όρισμα id αν τη καλέσουμε στην offersList, δηλαδή το αντικείμενο rd με το συγκεκριμένο id.

Έτσι, το αντικείμενο αυτό αφαιρείται από τη λίστα δωρεών του χρήστη – δωρητή.

- resetOffers( ) : δεν παίρνει κάποιο όρισμα και έχει τύπο επιστροφής void. Στο σώμα της καλούμε στη λίστα offersList τη μέθοδο reset( ) της *RequestDonationList*, η οποία καθαρίζει τη λίστα δωρεών του χρήστη – δωρητή.
- editOffer( ) : έχει ορίσμα ένα int πεδίο id και ένα double πεδίο quan με τύπο επιστροφής void. Στο σώμα της καλούμε στη λίστα offersList τη μέθοδο modify( ) της *RequestDonationList*, η οποία παίρνει σαν όρισμα το αντικείμενο που επιστρέφει η μέθοδος get( ) της *RequestDonationList* με όρισμα id αν τη καλέσουμε στην offersList, δηλαδή το αντικείμενο rd με το συγκεκριμένο id καθώς και το quan. Έτσι, ενημερώνει τη ποσότητα αυτού του αντικειμένου.

Υλοποιούμε έπειτα την μέθοδο commit( ) η οποία επίσης δεν παίρνει όρισμα και έχει τύπο επιστροφής void. Στο σώμα της καλούμε στη λίστα offersList τη μέθοδο commit( ) της *Offers*, η οποία ενημερώνει τα currentDonations του οργανισμού με τις προσφορές που περιέχονται στη λίστα rdEntities και όταν αυτό ολοκληρωθεί επιτυχώς, διαγράφει τα περιεχόμενα της λίστας rdEntities. Η υλοποίηση της commit() περιγράφεται πιο κάτω.

Ακόμα, υλοποιούμε πάλι την abstract μέθοδο showMenu( ) με τον ίδιο τρόπο καλώντας τώρα τη μέθοδο donatorMenu( ) της *Menu*, για να εκτυπώνει το menu που αντιστοιχεί στο χρήστη – δωρητή.

Στη συνέχεια, ορίζουμε και μια getter μέθοδο getOffers( ) η οποία επιστρέφει τη λίστα offersList για να μπορούμε να την επεξεργαστούμε και εκτός κλάσης λόγω της *private* φύσης της.

Τέλος, ορίζουμε τη μέθοδο toString( ) με τύπο επιστροφής String, η οποία κάνει @Override και επιστρέφει το όνομα και τον αριθμό τηλεφώνου του χρήστη χρησιμοποιώντας τη λέξη – κλειδί super ώστε να κληθεί η αντίστοιχη μέθοδος της πατρικής κλάσης.

2.1.3 Beneficiary

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***Beneficiary*** ορίζουμε δύο *public* constructors με ορίσματα πάλι αλφαριθμητικά **name** και **phone** ή αλφαριθμητικά **name**, **phone** και int πεδίο **noPersons**. Με τη λέξη – κλειδί super καλούμε τον constructor της ***User*** με ορίσματα αυτά τα δύο και εκτελείται η αντίστοιχη διαδικασία που περιγράφτηκε παραπάνω. Ακόμα, αν καλέσουμε τον πρώτο constructor χωρίς όρισμα για το **noPersons**, τότε η τιμή του αρχικιποιείται στο 1, ενώ αν καλέσουμε τον δεύτερο τότε πάλι με τη λέξη – κλειδί this εκχωρούμε στο πεδίο **noPersons** του στιγμιότυπου την τιμή του ορίσματος.

2. Methods

Όμοια με πριν, κάνουμε *wrap* τις μεθόδους **add( )**, **remove( ), reset( )** και **modify( )** των κλάσεων ***Requests*** και ***RequestDonationList*** δημιουργώντας τις αντίστοιχες **addRequest( )**, **removeRquset( )**, **resetRequests( )** και **editRequest( )** :

- οι **removeRquset( )**, **resetRequests( )** και **editRequest( )** καλούν στο σώμα τους τις αντίστοιχες μεθόδους της ***Requests*** που κάνουν *wrap* πάνω στο πεδίο **requestList**, προκειμένου ο χρήστης – επωφελούμενος να μπορεί να επεξεργαστεί τη λίστα με τα τρέχοντα είδη και ποσότητες που θέλει να λάβει.
- οι **addRequest( ), reset( )** και **commit( )** καλούν στο σώμα τους τις αντίστοιχες μεθόδους των ***Requests*** και ***RequestDonationList*** που κάνουν *wrap* πάνω στο πεδίο **receivedList**, προκειμένου ο χρήστης – επωφελούμενος να μπορεί να επεξεργαστεί τη λίστα με τα είδη και ποσότητες που έχει ήδη λάβει.

Όλες οι μέθοδοι αναλύεται πιο κάτω πώς υλοποιούνται.

Ακόμα, υλοποιούμε πάλι την abstract μέθοδο **showMenu( )** με τον ίδιο τρόπο καλώντας τώρα τη μέθοδο **beneficiaryMenu( )** της ***Menu***, για να εκτυπώνει το menu που αντιστοιχεί στο χρήστη – επωφελούμενο.

Τέλος, ορίζουμε και **getters getRequests( )** και **getNoPersons( ),** οι οποίες επιστρέφουν αντίστοιχα τη λίστα **requestList** και τη τιμή του πεδίου **noPersons** για να μπορούμε να τα επεξεργαστούμε και εκτός κλάσης λόγω της *private* φύσης τους.

2.2 Entity

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Entity* ορίζουμε *public* constructor με ορίσματα αλφαριθμητικά **name** και **description**. Με τη λέξη – κλειδί this εκχωρούμε στα πεδία **name** και **description** του στιγμιότυπου τις τιμές των ορισμάτων. Ακόμα, με την εντολή id = count ++; εκχωρούμε στο πεδίο **id** κάθε αντικειμένου που δημιουργείται κάθε φορά ένα **id** αυξημένο κατά 1 σε σχέση με το **id** του προηγούμενου αντικειμένου που δημιουργήσαμε για καλύτερη οργάνωση.

2. Methods

Ορίζουμε την **getEntityInfo( )** με τύπο επιστροφής String, η οποία επιστρέφει με τη μορφή αλφαριθμητικού τα στοιχεία ενός αντικειμένου της *Entity*, δηλαδή το **id** και την περιγραφή του.

Ακόμα, ορίζουμε τη μέθοδο **toString( )** με τύπο επιστροφής String, η οποία κάνει @Override και επιστρέφει με τη μορφή αλφαριθμητικού το αποτέλεσμα της **getEntityInfo( )**, δηλαδή τα στοιχεία κάθε αντικειμένου της *Entity*.

Τέλος, ορίζουμε και μια **getter** μέθοδο **getID( )** η οποία επιστρέφει το πεδίο **id** για να μπορούμε να το επεξεργαστούμε και εκτός κλάσης λόγω της *private* φύσης του.

2.2.1 Material

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Material* ορίζουμε *public* constructor με ορίσματα πάλι αλφαριθμητικά **name** και **phone**. Με τη λέξη – κλειδί super καλούμε τον constructor της *User* με ορίσματα αυτά τα δύο και εκτελείται η αντίστοιχη διαδικασία που περιγράφτηκε παραπάνω. Ακόμα, θέτουμε και άλλα τρία double ορίσματα **level1**, **level2**, **level3** όπου πάλι με τη λέξη – κλειδί this εκχωρούμε στα πεδία **level1**, **level2**, **level3** του στιγμιότυπου τις τιμές των ορισμάτων.

2. Methods

Ορίζουμε την **getDetails( )** με τύπο επιστροφής String, η οποία επιστρέφει με τη μορφή αλφαριθμητικού τις τιμές των **level1**, **level2**, **level3**, δηλαδή τις ποσότητες κάθε αντικειμένου της *Entity* που αντιστοιχούν στο χρήστη – επωφελούμενο ανάλογα με το πόσα άτομα (**noPersons**) έχει δηλώσει.

Τέλος, ορίζουμε και **getters getLevel1( ), getLevel2( )** και **getLevel3( ),** οι οποίες επιστρέφουν αντίστοιχα τις τιμές των πεδίων **level1, level2, level3** για να μπορούμε να τις επεξεργαστούμε και εκτός κλάσης λόγω της *private* φύσης τους.

2.2.2 Service

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Service* ορίζουμε *public* constructor με ορίσματα πάλι αλφαριθμητικά **name** και **phone**. Με τη λέξη – κλειδί super καλούμε τον constructor της *User* με ορίσματα αυτά τα δύο και εκτελείται η αντίστοιχη διαδικασία που περιγράφτηκε παραπάνω.

2. Methods

Ορίζουμε την **getDetails( )** με τύπο επιστροφής String, η οποία επιστρέφει με τη μορφή αλφαριθμητικού ότι το αντικείμενο τύπου *Entity* που την καλεί είναι *Service*.

Εδώ έχουμε υπερκάλυψη (overriding) της μεθόδου getDetails( ) καθώς υλοποιείται με διαφορετικό τρόπο στις δύο υποκλάσεις της *Entity*.

2.3 RequestDonation

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***RequestDonation*** ορίζουμε *public* constructor με ορίσματα ένα αντικείμενο **entity** τύπου ***Entity*** και ένα double **quantity**. Με τη λέξη – κλειδί this εκχωρούμε στα πεδία **entity** και **quantity** του στιγμιότυπου τις τιμές των ορισμάτων. Έπειτα, στο int πεδίο **ID** εκχωρούμε την τιμή του **id** του συγκεκριμένου **entity** για το οποίο μιλάμε, καλώντας σε αυτό την **getter** μέθοδο **getID( )**.

2. Methods

Αρχικά, ορίζουμε τη μέθοδο **compare( )** με όρισμα ένα αντικείμενο **rd** της ***RequestDonation*** και τύπο επιστροφής int. Με έναν βρόχο if ελέγχουμε εάν το **id** του συγκεκριμένου **entity** είναι το ίδιο με το **id** του **entity** του αντικειμένου **rd** που έχουμε περάσει σαν όρισμα. Αυτό συμβαίνει εάν η τιμή του **id** που επιστρέφει η **getID( )** είναι ίση με την τιμή του **id** που επιστρέφει η **getID( )** αν τη καλέσουμε σε αυτό το αντικείμενο **rd**. Εάν ο έλεγχος είναι επιτυχής, σημαίνει ότι τα δύο entities είναι ταυτόσημα και επιστρέφεται η τιμή 1, αλλιώς επιστρέφεται 0.

Ακόμα, ορίζουμε τη μέθοδο **toString( )** με τύπο επιστροφής String, η οποία κάνει @Override και επιστρέφει με μορφή αλφαριθμητικού το id, τις πληροφορίες του συγκεκριμένου **entity** καθώς και την διαθέσιμη ποσότητά του. Για να εκτυπωθούν όλα αυτά καλούμε στο **entity** τη μέθοδο **toString( )** της ***Entity***, ενώ κάνουμε και return τις τιμές των πεδίων **ID** και **quantity**.

Τέλος, ορίζουμε **setters setQuantity( )** και **getters getQuantity( )**, **getEntity( )** που μας επιτρέπουν να επεξεργαζόμαστε τα πεδία της ***RequestDonation*** και εκτός του εύρους της, λόγω της *private* φύσης τους.

2.4 Organization

- Για τη χρήση δυναμικών πινάκων – arraylists κάνουμε import το πακέτο java.util.ArrayList.

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***Organization*** ορίζουμε *public* constructor χωρίς ορίσματα. Με τη λέξη – κλειδί new δημιουργούμε τέσσερα arraylists **entityList**, **donatorList**, **beneficiaryList** και **currentDonations** που αντιπροσωπεύουν :

- τη λίστα των ειδών
- τη λίστα των χρηστών – δωρητών
- τη λίστα των χρηστών – επωφελούμενων
- τη λίστα με τις τρέχουσες δωρεές ειδών

που ανήκουν σε κάθε οργανισμό που δημιουργείται.

2. Methods

Αρχικά, ορίζουμε τη μέθοδο **hasEnough( )** με όρισμα ένα αντικείμενο **rd** της ***RequestDonation*** και τύπο επιστροφής boolean που ελέγχει αν υπάρχει στον οργανισμό επαρκής ποσότητα ενός συγκεκριμένου **entity**. Σε ένα double πεδίο **quan** αποθηκεύουμε τη ποσότητα του συγκεκριμένου **entity** καλώντας σε αυτό τη **getter** μέθοδο **getQuantity( )** που επιστρέφει την τιμή του **quantity** για αυτό το είδος. Με έναν βρόχο if – else ελέγχουμε αν το συγκεκριμένο είδος ανήκει στη λίστα **currentDonations** και αν η διαθέσιμη ποσότητά του είναι μεγαλύτερη από την **quan** που ζητάμε. Αυτό γίνεται άμα καλέσουμε στην **currentDonations** τη μέθοδο **get( )** με όρισμα το **id** του επιθυμητού αντικειμένου **rd** που επιστρέφει η μέθοδος **getID( )** καλούμενη σε αυτό. Έτσι, εξασφαλίζουμε ότι το συγκεκριμένο **entity** υπάρχει και μετά πάνω σε αυτό καλούμε και τη **getter** μέθοδο **getQuantity( )** που επιστρέφει τη ποσότητα του entity από το **rd.** Αν η ποσότητα είναι μεγαλύτερη ή ίση από το πεδίο **quan** τότε επιστρέφεται τιμή true, αλλιώς false.

Έπειτα, ορίζουμε τη μέθοδο **findUser( )** με String παράμετρο **phone** και τύπο επιστροφής ***User***. Καλώντας στο αντικείμενο **admin** της ***Admin*** την **getter** μέθοδο

getPhone( ) επιστρέφεται το αλφαριθμητικό με το τηλέφωνο του admin και έπειτα καλώντας πάνω σε αυτό τη μέθοδο matches( ) με όρισμα τη παράμετρο phone, ελέγχουμε αν το τηλέφωνο που δώσαμε αντιστοιχεί σε κάποιον χρήστη – διαχειριστή. Αυτή τη συνθήκη τη βάζουμε σε έναν βρόχο for και αν επαληθεύεται επιστρέφουμε τον διαχειριστή admin. Στη συνέχεια, για να δούμε τι είδους χρήστης είναι με ένα επαυξημένο βρόχο for διατρέχουμε όλα τα αντικείμενα των donatorList και beneficiaryList και πάλι με τον ίδιο τρόπο ελέγχουμε αν το τηλέφωνο του χρήστη αντιστοιχεί σε δωρητή ή επωφελούμενο. Αν δεν υπάρχει πουθενά χρήστης με τέτοιο τηλέφωνο επιστρέφουμε null.

Ακόμα, για τα entities ορίζουμε τις μεθόδους :

- addEntity( ) : παίρνει ως ορίσματα ένα αντικείμενο e τύπου *Entity* και ένα double πεδίο q με τύπο επιστροφής void. Ελέγχουμε εάν το συγκεκριμένο entity υπάρχει ήδη και εκτυπώνουμε το κατάλληλο μήνυμα. Πάλι με έναν επαυξημένο βρόχο for διατρέχουμε τη λίστα entityList και με έναν βρόχο if ελέγχουμε την ύπαρξη. Αυτό γίνεται αν καλέσουμε τη μέθοδο matches( ) με όρισμα το αποτέλεσμα της getEntityInfo( ) στο αντικείμενο e πάνω στο αποτέλεσμα της getEntityInfo( ) στο αντικείμενο en της entityList. Έτσι, αν επαληθεύεται η συνθήκη εγείρουμε ένα νέο αντικείμενο της κλάσης ExceptionError (δηλαδή εξαίρεση) με όρισμα το κατάλληλο μήνυμα για να εκτυπώνει ότι αυτό το entity υπάρχει. Αλλιώς το entity e το προσθέτουμε στη λίστα entityList με τη μέθοδο add( ), δημιουργούμε και ένα αντικείμενο rd τύπου *RequestDonation* με ορίσματα e, q και το προσθέτουμε και αυτό στη λίστα currentDonations για να ξέρουμε ότι το έλαβε ο επωφελούμενος.
- removeEntity( ): παίρνει ως όρισμα ένα int πεδίο id με τύπο επιστροφής void. Πάλι με έναν επαυξημένο βρόχο for διατρέχουμε τη λίστα entityList και αν το id του αντικειμένου e της λίστας entityList έχει το ίδιο id με αυτό που δώσαμε σαν παράμετρο τότε το αφαιρούμε από αυτή και αφαιρούμε και το id από την currentDonations. Αυτό γίνεται αν καλέσουμε τη remove( ) με όρισμα το αντικείμενο e στην entityList και με όρισμα το αποτέλεσμα που επιστρέφει η get( ) με όρισμα id αν τη καλέσουμε στην currentDonations.

Για τους donators ορίζουμε τις μεθόδους :

- insertDonator( ): παίρνει ως όρισμα ένα αντικείμενο d τύπου *Donator* με τύπο επιστροφής void. Προσθέτουμε τον δωρητή d καλώντας στη λίστα donator List τη μέθοδο add( ) με όρισμα τον d.
- removeDonator( ) : παίρνει ως όρισμα ένα πεδίο String phoneNumber με τύπο επιστροφής void. Πάλι με έναν βρόχο for διατρέχουμε τη λίστα donatorList και ελέγχουμε όπως και πριν αν υπάρχει δωρητής με το τηλέφωνο που έχουμε δώσει σαν όρισμα. Αν ναι τότε καλούμε στη donatorList τη μέθοδο remove( ) με όρισμα τον d και σταματάμε τον έλεγχο με την εντολή break.

Για τους beneficisries ορίζουμε τις μεθόδους removeBeneficiary( ), insertBeneficiary( ) οι οποίες λειτουργούν όπως και για τους donators αλλά ψάχνοντας τώρα στη λίστα beneficiaryList.

Ορίζουμε και τις μεθόδους printListedMaterials( ) και printListedServices( ) όπου πάλι με ένα βρόχο for διαβάζουμε κάθε αντικείμενο s της entityList. Αρχικά, ορίζουμε έναν μετρητή i=0 και ελέγχουμε αν το entity είναι material ή service με τον εξής τρόπο : αν καλέσουμε στο αποτέλεσμα της getDetails( ) πάνω στο s την μέθοδο matches( ) με όρισμα "Services" και επιστραφεί 1 τότε το s είναι *Service* αλλιώς είναι Material. Τότεαυξάνουμε την τιμή του i κατά 1 για να εκτυπώνεται σωστά η σειρά και καλούμε στο s τις κατάλληλες getters μεθόδους καθώς και την toString( ) για να εκτυπώνονται τα στοιχεία του. Μετά ορίζουμε και τη listEntities( ) που στην ουσία εκτελεί τις δύο παραπάνω μεθόδους μαζί.

Ορίζουμε και τις listDonators( ) και listBeneficiaries( ) που διατρέχουν τις donatorList και beneficiaryList και σε κάθε αντικείμενό τους καλούν την toString( ) για να επιστρέφονται τα χαρακτηριστικά κάθε είδους της *Entity* ως αλφαριθμητικά και να εκτυπώνονται.

Επιπλέον, ορίζουμε τη resetBeneficiariesLists( ) που διατρέχει τη λίστα beneficiaryList και σε κάθε αντικείμενό της b καλεί τη μέθοδο reset( ) έτσι ώστε να κληθεί η clear( ) σε κάθε b και να διαγραφτεί η λίστα receivedList που του αντιστοιχεί.

Τέλος, ορίζουμε κατάλληλες setters setAdmin( ) και getters getAdmin, getEntity( ) που μας επιτρέπουν να διαχειριζόμαστε αντικείμενα της *Admin* και της *Entity* εκτός από το εύρος των κλάσεων.

2.5 RequestDonationList

- Για τη χρήση δυναμικών πινάκων – arraylists κάνουμε import το πακέτο java.util.ArrayList.

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***RequestDonationList*** ορίζουμε *public* constructor χωρίς ορίσματα. Στο σώμα του δημιουργείται μια λίστα **rdEntities**, ώστε ο δωρητής να μπορέσει να δώσει τα είδη από τα ήδη διαθέσιμα τα οποία είναι αποθηκευμένα στην λίστα.

2. Methods

Αρχικά, ορίζουμε τη μέθοδο **get( )** η οποία δέχεται ως όρισμα ένα int πεδίο **id** και επιστρέφει ένα αντικείμενο τύπου ***RequestDonation***. Στο σώμα της μέσω του επαυξημένου βρόγχου for, ελέγχεται η λίστα **rdEntities** μέσα από το αντικείμενο **rd** του τύπου ***RequestDonation***, ώστε μέσα από τον βρόχο if, να συγκρίνουμε το **id** που έδωσε ο χρήστης αν ταιριάζει με κάποιο από τα αντικείμενα ***RequestDonation*** που βρίσκονται στην λίστα **rdEntities** (Γι' αυτό καλούμε την μέθοδο **getID( )** μέσω του αντικειμένου **rd**, ώστε να μας επιστρέφει το ID του κάθε στιγμιότυπου και να το συγκρίνει). Αν είναι ταυτόσημα επιστρέφεται το ίδιο το στιγμιότυπο. Αν όμως τα δεδομένα δεν είναι διαθέσιμα επιστρέφει null, έναν δείκτη ο οποίος δεν δείχνει πουθενά, δηλαδή επιστρέφει όλα τα id των αντικειμένων της λίστας **rdEntities**.

Έπειτα, ορίζουμε τη μέθοδο **add( )**, η οποία δέχεται ως όρισμα αντικείμενο **rd** τύπου ***RequestDonation*** και έχει τύπο επιστροφής void. Στο σώμα της μέσα από το βρόγχο επανάληψης for και με την χρήση του αντικειμένου **req**, διατρέχουμε όλη την λίστα **rdEntities** και μέσα από τον βρόγχο επανάληψης if ελέγχουμε αν το αντικείμενο **req** ταυτίζεται με το **rd**, ώστε να ελέγξουμε αν αυτό το αντικείμενο **rd** τύπου ***RequestDonation*** υπάρχει στην λίστα **rdEntities**. Αν όντως υπάρχει χρησιμοποιούμε την μέθοδο **modify( )**, ώστε να τροποποιήσουμε την ποσότητα του τρέχοντος στιγμιότυπου και διοχετεύουμε στην μέθοδο αυτή ως ορίσματα το **req** και το

άθροισμα των ποσοτήτων των δύο αντικειμένων τύπου ***RequestDonation***. Αν όμως δεν υπάρχει στην λίστα **rdEntities**, δηλαδή δεν υπάρχει σε κάποιο requestDonation, τότε προσθέτουμε στην λίστα το στιγμιότυπο **rd**.

Ακόμα, δημιουργούμε τις πιο κάτω μεθόδους για χειρισμό της λίστας :

- **remove( )** : δέχεται σαν όρισμα αντικείμενο **rd** τύπου ***RequestDonation*** και έχει τύπο επιστροφής void. Στο σώμα της καλεί την μέθοδο **remove( )** στη λίστα **rdEntities**, με όρισμα το στιγμιότυπο **rd**, έτσι ώστε να το διαγράψει από την λίστα.
- **modify( )** : δέχεται σαν όρισμα αντικείμενο **rd** τύπου ***RequestDonation*** και το double πεδίο **quantity** και έχει τύπο επιστροφής void. Στο σώμα της καλείται η μέθοδος **setQuantity( )** με όρισμα το πεδίο **quantity**, ώστε να μεταβάλουμε την ποσότητα του συγκεκριμένου στιγμιότυπου. Η μέθοδος καλείται μέσω της μεθόδου **get( )**, η οποία έχει σαν όρισμα το ID του στιγμιότυπου **rd** (το οποίο το λάβαμε μέσω της μεθόδου **getID( )**), όπου η μέθοδος **get( )** κλήθηκε μέσω της λίστας **rdEntinties**.
- **monitor( )** : δεν δέχεται κάποιο όρισμα και έχει τύπο επιστροφής void. Στο σώμα της μέσω του βρόγχου επανάληψης for και με την χρήση του αντικειμένου **rd** τύπου ***RequestDonation***, διατρέχουμε την λίστα **rdEntities** και σε κάθε αντικείμενο που βρίσκεται στην λίστα καλείται η μέθοδος του **toString( )**, ώστε να εκτυπωθούν όλα τα αντικείμενα που βρίσκονται στην λίστα με την κατάλληλη μορφοποίηση (Όνομα | Ποσότητα) .
- **reset( )** : δεν δέχεται κάποιο όρισμα και έχει τύπο επιστροφής void. Στο σώμα της καλείται μέσω της λίστας **rdEntities** η μέθοδος **clear( )**, ώστε να καθαριστεί η λίστα και να διαγραφούν όλα τα υπάρχοντα στιγμιότυπα σε αυτή.

Τέλος, ορίζουμε και κατάλληλη **getter** μέθοδο **getrdEntities( )**, η οποία επιστρέφει τη λίστα και μας επιτρέπει να την επεξεργαζόμαστε και έξω από το εύρος αυτής της κλάσης.

2.5.1 Requests

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***Requests*** ορίζουμε *public* constructor και με τη λέξη – κλειδί super καλούμε τον constructor της πατρικής της κλάσης *RequestDonationList*.

2. Methods

Αρχικά, υπερκαλύπτουμε τις μεθόδους add( ) και modify( ) της πατρικής κλάσης και τις υλοποιούμε με τον ακόλουθο τρόπο :

- add( ) : δέχεται 3 ορίσματα, ένα αντικείμενο o τύπου *Organization*, ένα αντικείμενο rd τύπου *RequestDonation* και ένα αντικίμενο b τύπου *Beneficiary*, ενώ έχει τύπο επιστροφής void. Στο σώμα της οποίας μέσω του βρόγχου if, ελέγχουμε εάν υπάρχει η κατάλληλη ποσότητα αντικειμένων rd μέσα στον οργανισμό καλώντας την μέθοδο hasEnough( ) της κλάσης *Organization* με όρισμα το rd. Εφόσον δεν υπάρχει η κατάλληλη ποσότητα εγείρεται εξαίρεση και αναγράφεται το αντίστοιχο μήνυμα δημιουργώντας ένα νέο στιγμιότυπο της κλάσης *ExceptionError*. Με έναν δεύτερο βρόγχο if χρησιμοποιούμε την μέθοδο validRequestDonation( ) που ορίζουμε παρακάτω, με ορίσματα το rd και το b, ώστε να ελέγξουμε αν το στιγμιότυπο b δικαιούται την αιτούμενη ποσότητα. Αν την δικαιούται μέσω της λέξης – κλειδί super καλούμε την μέθοδο add( ) της υπερκλάσης με όρισμα το rd, ώστε να ενημερωθεί η λίστα. Αν όμως δεν την δικαιούται εγείρεται εξαίρεση διαφορετικού τύπου από την προηγούμενη και αναγράφεται αντίστοιχο μήνυμα.
- modify( ) : δέχεται 4 ορίσματα, ένα στιγμιότυπο o τύπου *Organization*, ένα στιγμιότυπο rd τύπου *RequestDonation*, ένα double πεδίο quantity και ένα στιγμιότυπο b τύπου *Beneficiary*, ενώ έχει τύπο επιστροφής void. Στο σώμα της μέσω του βρόγχου if, ελέγχουμε εάν υπάρχει η κατάλληλη ποσότητα αντικειμένων rd μέσα στον οργανισμό καλώντας την μέθοδο hasEnough( ) της κλάσης *Organization* με όρισμα το rd. Εφόσον δεν υπάρχει η κατάλληλη ποσότητα εγείρεται εξαίρεση και αναγράφεται το αντίστοιχο μήνυμα. Με έναν δεύτερο βρόγχο if χρησιμοποιούμε την μέθοδο validRequestDonation( )

με ορίσματα το rd και το b, ώστε να ελέγξουμε αν το στιγμιότυπο b δικαιούται την ποσότητα. Αν την δικαιούται μέσω της λέξης – κλειδί super καλούμε την μέθοδο setQuantity( ) με όρισμα το πεδίο quantity, ώστε να μεταβάλουμε την ποσότητα του συγκεκριμένου στιγμιοτύπου. Η μέθοδος καλείται μέσω της μεθόδου get( ), η οποία έχει σαν όρισμα το ID του στιγμιότυπου rd (το οποίο το λάβαμε μέσω της μεθόδου getID( )), όπου η μέθοδος get( ) κλήθηκε μέσω του super, δηλαδή του constructor της υπερκλάσης. Αν όμως δεν την δικαιούται εγείρεται εξαίρεση διαφορετικού τύπου από την προηγούμενη και αναγράφεται αντίστοιχο μήνυμα.

Τώρα ορίζουμε και τη μέθοδο validRequestDonation( ) με τύπο επιστροφής boolean και δύο ορίσματα rd της *RequestDonation* και b της *Beneficiary*. Στο σώμα της δηλώνουμε αντικείμενο e τύπου *Entity* και το αρχικοποιούμε καλώντας την μέθοδο getEntity( ) στο στιγμιότυπο rd. Μέσω του βρόγχου if ελέγχουμε αν είναι service, για αυτό καλούμε μέσω του αντικειμένου e την μέθοδο getDetails( ), η οποία θα μας επιστρέψει αν είναι service ή material. Έπειτα, μέσω της μεθόδου matches( ) ελέγχουμε αν είναι όντως service, ώστε να επιστρέψει true και να μην γίνει περαιτέρω έλεγχος. Στη συνέχεια δηλώνουμε το int πεδίο noPersons, ώστε να συμβολίσουμε τον αριθμό των ατόμων, το οποίο το αρχικοποιούμε μέσω της μεθόδου getNoPersons( ), η οποία μας επιστρέφει τον αριθμό των ατόμων. Εφόσον ελέγξαμε ότι δεν είναι service, τότε θα είναι material και μέσω των διαδοχικών ελέγχων if-else, ελέγχουμε αν η ποσότητα των rd είναι εντός των επιτρεπόμενων ορίων που όρισε ο επωφελουμενος. Με το πρώτο if ελέγχουμε αν ο αριθμός των ατόμων είναι ίσος με 1. Αν είναι τότε ο εμφωλευμένος βρόγχος if ελέγχει αν η ποσότητα του στιγμιοτύπου rd είναι μικρότερη ή ίση από το level1 του είδους e τύπου *Material* και αν όντως είναι επιστρέφει true. Μετά ελέγχουμε αν ο αριθμός των ατόμων είναι μεγαλύτερος του 1 και μικρότερος του 5. Αν είναι τότε ο εμφωλευμένος βρόγχος if ελέγχει αν η ποσότητα του στιγμιοτύπου rd είναι μικρότερη ή ίση από το level2 του e τύπου Material και αν όντως είναι επιστρέφει true. Με το δεύτερο else - if δεν ισχύει ότι ο αριθμός των ατόμων είναι ίσος με 1 ούτε ότι είναι μεγαλύτερος του 1 και μικρότερος του 5 και έτσι ελέγχουμε αν ο αριθμός των ατόμων είναι μεγαλύτερος του 5. Αν είναι τότε ο εμφωλευμένος βρόγχος if ελέγχει αν η ποσότητα του στιγμιοτύπου rd είναι μικρότερη ή ίση από το level3 του e τύπου Material και αν όντως είναι επιστρέφει true. Αν δεν ισχύει καμία από τις συνθήκες ελέγχου τότε επιστρέφει false.

Τέλος, ορίζουμε τη μέθοδο commit( ) η οποία δέχεται όρισμα ένα στιγμιότυπο b τύπου *Beneficiary* και έχει τύπο επιστροφής void. Μέσα στο σώμα της με τον βρόγχο επανάληψης for τρέχουμε με την χρήση του αντικειμένου rd τύπου *RequestDonation* την μέθοδο getrdEntities( ) και μέσω του βρόγχου if ελέγχεται μέσω τις μεθόδου validRequestDonation( ), διοχετεύοντας τα ορίσματα b και rd , αν το b (ο επωφελούμενος δηλαδή) δικαιούται την ποσότητα με βάση των αριθμό των ατόμων και αν αυτή η ποσότητα είναι ακόμη προς χρήση στον οργανισμό donation. Αν ισχύουν οι προϋποθέσεις, τότε εκτυπώνεται μήνυμα ότι έγινε σωστός χειρισμός του στιγμιότυπου rd, αναγράφοντας το όνομά του και την ποσότητα του. Επιπλέον διαγράφουμε το rd και από την λίστα rdEntities.

2.5.2 Offers

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Offers* ορίζουμε *public* constructor ο οποίος όμοια με πριν με τη λέξη – κλειδί super καλεί τον constructor της πατρικής της κλάσης *RequestDonationList* και εκτελεί τις εντολές του.

2. Methods

Σαν επιπλέον μέθοδο ορίζουμε μόνο την commit( ) η οποία δεν δέχεται κάποιο όρισμα και έχει τύπο επιστροφής void. Με τον βρόγχο for διατρέχουμε τη λίστα και σε κάθε αντικείμενο rd τύπου *RequestDonation* καλούμε την μέθοδο getrdEntities( ), ώστε να ενημερώσουμε τις τρέχουσες δωρεές με τις προσφορές, δηλαδή διατρέχουμε την λίστα rdEntities καθώς η μέθοδος μας επιστρέφει την λίστα rdEntities και εκτυπώνουμε το μήνυμα ότι έγινε σωστή αντιμετώπιση του κάθε αντικειμένου rd, το οποίο βρίσκεται στην λίστα. Στο τέλος διαγράφουμε και όλα τα στιγμιότυπα rd μέσω της remove( ) από την λίστα.

- Πάλι υπερκαλύπτουμε και τη μέθοδο commit( ) καθώς την υλοποιούμε διαφορετικά στις δύο υποκλάσεις της *RequestDonationList*.

2.6 Menu

- Η επικοινωνία του προγράμματός μας με τον έξω κόσμο επιτυγχάνεται με την βοήθεια της κλάσης Scanner. Για να συμπεριλάβουμε την κλάση Scanner κάνουμε import το πακέτο java.util.Scanner. Ο χρήστης εισάγει δεδομένα από το πληκτρολόγιο με την χρήση του ρεύματος εισόδου System.in και το πρόγραμμά μας τα διαβάζει. Έτσι, οι τιμές κατανέμονται και αποθηκεύονται στην μνήμη.

1. Constructors

Για τη δημιουργία στιγμιότυπου της ***Menu*** ορίζουμε *public* constructor ο οποίος δέχεται σαν όρισμα ένα αντικείμενο org τύπου ***Organization***. Μέσα στο σώμα του με τη λέξη – κλειδί this . Έπειτα, καλούμε και τη μέθοδο initMenu( ) έτσι ώστε όταν δημιουργηθεί το στιγμιότυπο της ***Menu***, να εμφανιστεί το αρχικοποιημένο menu στην οθόνη.

2. Methods

Αρχικά, ορίζουμε τις μεθόδους που καλωσορίζουν τον χρήστη στο menu :

- scanForPhone( ) : δεν δέχεται καμία παράμετρο και έχει τύπο επιστροφής String. Στο σώμα της μέσω της System.out.println εκτυπώνεται το μήνυμα να δώσει ο χρήστης (είτε είναι ***Admin***, είτε ***Donator***, είτε ***Beneficiary***) το τηλέφωνό του. Έπειτα δημιουργούμε αντικείμενο in με το οποίο διαβάζουμε τα δεδομένα που εισάγει ο χρήστης και τα καταχωρούμε σε αυτό. Εφόσον το in απέκτησε περιεχόμενο, χρησιμοποιούμε την μέθοδο nextLine( ), που μας λέει : αυτό που διάβασες μέχρι να πετύχεις την αλλαγή γραμμής ανάθεσε το στο String πεδίο phone και επίστρεψέ το.
- scanForName( ) : Όμοια με την προηγούμενη μέθοδο, δεν δέχεται καμία παράμετρο και επιστρέφει String. Στο σώμα της μέσω της System.out.println εκτυπώνεται το μήνυμα να δώσει ο χρήστης (είτε είναι ***Admin***, είτε ***Donator***, είτε Beneficiary) το όνομά του. Έπειτα δημιουργούμε αντικείμενο in, διαβάζουμε τα δεδομένα που εισάγει ο χρήστης και τα καταχωρούμε στο in. Εφόσον το in απέκτησε περιεχόμενο, χρησιμοποιούμε πάλι την μέθοδο nextLine( ), που μας

λέει : αυτό που διάβασες μέχρι να πετύχεις την αλλαγή γραμμής ανάθεσε το στο String πεδίο **name** και επίστρεψέ το.

- **completeRegister( )** : έχει τύπο επιστροφής αντικείμενο τύπου *User* και 3 ορίσματα : ένα στιγμιότυπο **org** τύπου *Organization* και τις συμβολοσειρές **phone** και **name**, από τις προηγούμενες μεθόδους **scanForPhone( )** και **scanForName( )** αντίστοιχα. Στο σώμα της μέσω της System.out.println εκτυπώνεται το μήνυμα να δώσει ο χρήστης την τιμή 1 αν είναι *Donator* ή την τιμή 2 αν είναι *Beneficiary*. Έπειτα δημιουργούμε αντικείμενο **in**, διαβάζουμε τα δεδομένα που εισάγει ο χρήστης και τα καταχωρούμε στο **in**. Εφόσον το **in** απέκτησε περιεχόμενο ,χρησιμοποιούμε ξανά την μέθοδο **nextInt( )**,που μας λέει : αυτό που διάβασες μέχρι να πετύχεις τον επόμενο ακέραιο ανάθεσε το στο int πεδίο **selection**. Μέσω την συνθήκης ελέγχου switch, διακρίνουμε τις περιπτώσεις :
  i. Αν **selection**=1, τότε δημιουργούμε έναν *Donator* σύμφωνα με τον constructor της κλάσης *Donator* με ορίσματα το όνομα και το τηλέφωνο που δώσαμε στις προηγούμενες μεθόδους. Μέσω του στιγμιοτύπου **org** καλούμε την μέθοδο **insertDonator( )** με όρισμα το **u**, που είναι τύπου *Donator*, ώστε ο καινούριος *Donator* να προστεθεί στην λίστα **donatorList** και μετά να επιστρέφει το στιγμιότυπο **u**.
  ii. Αν **selection**=2 τότε δημιουργούμε έναν *Beneficiary* σύμφωνα με τον constructor της κλάσης *Beneficiary* με ορίσματα το όνομα και το τηλέφωνο που δώσαμε στις προηγούμενες μεθόδους. Μέσω του στιγμιοτύπου **org** καλούμε την μέθοδο **insertBeneficiary( )** με όρισμα το **b**, που είναι τύπου *Beneficiary*, ώστε ο καινούριος *Beneficiary* να προστεθεί στην λίστα **beneficiaryList** και να επιστρέφει το στιγμιότυπο **b**.

Σε περίπτωση που δοθεί άλλη τιμή εκτυπώνουμε το κατάλληλο μήνυμα λάθους και βάζουμε τον χρήστη να ξανά επιλέξει έναν από τους δύο ρόλους, επιστρέφοντας την μέθοδο **completeRegister( )** με τα κατάλληλα ορίσματα.

- **greet( )** : έχει τύπο επιστροφής void και δέχεται 2 παραμέτρους, ένα στιγμιότυπο **u** τύπου *User* και την συμβολοσειρά **role**. Στο σώμα της με την χρήση της System.out.println εκτυπώνεται το καλωσόρισμα εμφανίζοντας το όνομα που δώσαμε μέσω της μεθόδου **getName( )** της κλάσης *User* και τον ρόλο (Donator | Beneficiary) .

- scanNum( ) : Η μέθοδος αυτή έχει οριστεί ως *private*, οπότε δεν έχει πρόσβαση σε μεθόδους και μεταβλητές εκτός κλάσεις, έχει τύπο επιστροφής int και δεν έχει κανένα όρισμα. Στο σώμα της δηλώνουμε το int πεδίο selection. Έπειτα δημιουργούμε αντικείμενο in, διαβάζουμε τα δεδομένα που εισάγει ο χρήστης και τα καταχωρούμε στο in. Εφόσον το in απέκτησε περιεχόμενο, χρησιμοποιούμε την μέθοδο nextInt( ), που μας λέει : αυτό που διάβασες μέχρι να πετύχεις τον επόμενο ακέραιο ανάθεσε το στο int πεδίο selection και επίστρεψέ το.
- getMenuOption( ) : Η μέθοδος αυτή έχει οριστεί ως *private*, οπότε δεν έχει πρόσβαση σε μεθόδους και μεταβλητές εκτός κλάσεις, έχει τύπο επιστροφής int και έχει όρισμα ένα int πεδίο max. Στο σώμα της δηλώνουμε το int πεδίο selection. Έπειτα δημιουργούμε αντικείμενο in, διαβάζουμε τα δεδομένα που εισάγει ο χρήστης και τα καταχωρούμε στο in. Εφόσον το in απέκτησε περιεχόμενο, χρησιμοποιούμε την μέθοδο nextInt( ), που μας λέει : αυτό που διάβασες μέχρι να πετύχεις τον επόμενο ακέραιο ανάθεσε το στο int πεδίο selection. Μέσα από τον βρόγχο if ελέγχουμε αν το selection είναι μεγαλύτερο από το max ή αν το selection είναι μικρότερο του 1. Τότε εκτυπώνεται το κατάλληλο μήνυμα ότι πρέπει να δώσεις μια επιλογή από 1 έως το μέγιστο και δίνεται η δυνατότητα ο χρήστης να ξανά επιλέξει, επιστρέφοντας την μέθοδο getMenuOption( ). Διαφορετικά επιστρέφεται το selection.

Στην συνέχεια διαχωρίζουμε το menu ανάλογα με το αν ο χρήστης είναι ***Admin***, ***Beneficiary*** ή ***Donator***.

Admin:

- adminViewMenu( ) : έχει τύπο επιστροφής void και δέχεται ως όρισμα το στιγμιότυπο a τύπου ***Admin***. Στο σώμα της μέσω της System.out.println εκτυπώνονται οι επιλογές αν θέλει να δει τα ***Material*** ή τα ***Services*** αλλά έχει και την επιλογή να επιστρέψει πίσω. Δηλώνουμε το int πεδίο selection και το αρχικοποιούμε σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 3, καθώς έχουμε 3 επιλογές αλλά και για να μην ξαναγράφουμε όλο το κομμάτι του κώδικα το να επιλέξουμε μια επιλογή και αν δεν είναι η επιλογή κατάλληλη να έχουμε την δυνατότητα να ξαναεπιλέξουμε, το οποίο υλοποιήσαμε στην μέθοδο getMenuOption( ).

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για το selection:

→ 1 : μέσω του στιγμιοτύπου org καλείται η μέθοδος printListedMaterials( ), ώστε να εκτυπωθούν οι πληροφορίες για όλα τα materials που υπάρχουν στην λίστα entitylist. Μέσω της System.out.println εμφανίζουμε στην οθόνη τα μηνύματα αν θέλουμε να επιστρέψουμε πίσω να πατήσουμε -1, αλλιώς να πατήσουμε το ID του material που θέλουμε να δούμε τα χαρακτηριστικά του. Δηλώνουμε την ακέραια μεταβλητή mid και την αρχικοποιούμε με την μέθοδο scanNum( ), ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο mid. Μέσω της συνθήκης ελέγχου if αν η επιλογή που κάναμε είναι -1, επιστρέφουμε ξανά στην adminViewMenu( ) και εκτελείται από την αρχή η μέθοδος. Μέσω της System.out.println μέσω του αντικειμένου org καλούμε την μέθοδο getEntity( ) με όρισμα την επιλογή mid, ώστε αν το ID του material που εκχώρησε ο χρήστης υπάρχει στη λίστα, τότε επιστρέφεται το αντικείμενο material με το αντίστοιχο ID. Μετά σε αυτό καλεί την μέθοδο toString( ), ώστε να εκτυπωθούν τα χαρακτηριστικά του.

→ 2 : μέσω του στιγμιοτύπου org καλείται η μέθοδος printListedServices( ), ώστε να εκτυπωθούν οι πληροφορίες για όλα τα services που υπάρχουν στην λίστα entitylist. Μέσω της System.out.println εμφανίζουμε στην οθόνη τα μηνύματα αν θέλουμε να επιστρέψουμε πίσω να πατήσουμε -1, αλλιώς να πατήσουμε το ID του service που θέλουμε να δούμε τα χαρακτηριστικά του. Δηλώνουμε την ακέραια μεταβλητή sid και την αρχικοποιούμε με την μέθοδο scanNum( ), ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο sid. Μέσω της συνθήκης ελέγχου if αν η επιλογή που κάναμε είναι -1, επιστρέφουμε ξανά στην adminViewMenu( ) και εκτελείται από την αρχή η μέθοδος. Μέσω της System.out.println μέσω του αντικειμένου org καλούμε την μέθοδο getEntity( ) με όρισμα την επιλογή sid, ώστε αν το ID του service που εκχώρησε ο χρήστης υπάρχει στη λίστα τότε επιστρέφεται το αντικείμενο service με το αντίστοιχο ID. Μετά καλεί σε αυτό την μέθοδο toString( ), ώστε να εκτυπωθούν τα χαρακτηριστικά του.

→ 3 : επιστρέφουμε ένα στάδιο πίσω στην μέθοδο adminMenu( ).

Άμα δεν επιλέξουμε καμία από τις επιλογές 1-2-3, τότε επιστρέφουμε στην adminViewMenu( ) και έχουμε την δυνατότητα να κάνουμε ξανά τις κατάλληλες επιλογές αυτή την φορά.

- adminMonitorMenu( ) : έχει τα'υπο επιστροφής void και έχει ως παράμετρο το στιγμιότυπο a τύπου *Admin*. Στο σώμα της μέσω της System.out.println εκτυπώνονται οι επιλογές αν θέλει να δει την λίστα των Beneficiaries ή την λίστα των Donators ή να επαναφέρει την λίστα των Beneficiaries ή έχει και την επιλογή να επιστρέψει πίσω. Δηλώνουμε την ακέραια μεταβλητή selection και την αρχικοποιούμε, σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 4, καθώς έχουμε 4 επιλογές.

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την selection

→ 1 : μέσω του στιγμιοτύπου org καλείται η μέθοδος listBeneficiaries( ), ώστε να εκτυπωθούν όλοι οι Beneficiaries που υπάρχουν στην λίστα beneficiaryList. Μέσω της System.out.println εμφανίζουμε στην οθόνη τα μηνύματα αν θέλουμε να επιστρέψουμε πίσω να πατήσουμε -1, αλλιώς να πατήσουμε το τηλέφωνο του *Beneficiary* που θέλουμε να διαγράψουμε. Δηλώνουμε την ακέραια μεταβλητή bid και την αρχικοποιούμε με την μέθοδο scanNum( ),ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο bid. Μέσω της συνθήκης ελέγχου if αν η επιλογή που κάναμε είναι -1, επιστρέφουμε ξανά στην adminMonitorMenu( ) και εκτελείται από την αρχή η μέθοδος. Μέσω της System.out.println μέσω του αντικειμένου org καλούμε την μέθοδο removeBeneficiary( ) με όρισμα την επιλογή bid, ώστε αν το τηλέφωνο του *Beneficiary* που εκχώρησε ο χρήστης υπάρχει στη λίστα τότε επιστρέφεται το αντικείμενο *Beneficiary*. Μετά στο στιγμιότυπο Beneficiary καλεί την μέθοδο toString( ), ώστε να διαγραφεί από την λίστα.
→ 2 : μέσω του στιγμιοτύπου org καλείται η μέθοδος listDonators( ), ώστε να εκτυπωθούν όλοι οι Donators που υπάρχουν στην λίστα donatorList. Μέσω της System.out.println εμφανίζουμε στην οθόνη τα μηνύματα αν θέλουμε να επιστρέψουμε πίσω να πατήσουμε -1, αλλιώς να πατήσουμε το τηλέφωνο του *Donator* που θέλουμε να διαγράψουμε. Δηλώνουμε την ακέραια μεταβλητή bid και την αρχικοποιούμε με την μέθοδο scanNum( ), ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο sid. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1, επιστρέφουμε ξανά στην adminMonitorMenu( ) και εκτελείται από την αρχή η μέθοδος. Μέσω της System.out.println μέσω του αντικειμένου org καλούμε την μέθοδο removeDonator( ) με όρισμα την επιλογή sid, ώστε αν το τηλέφωνο του *Donator* που εκχώρησε ο χρήστης υπάρχει στη λίστα, τότε επιστρέφεται το αντικείμενο *Donator*. Μετά στο στιγμιότυπο *Donator* καλείται η

μέθοδος **toString( ),** ώστε να διαγραφεί από την λίστα.

→ 3 : μέσω του στιγμιοτύπου **org** καλούμε την μέθοδο **resetBeneficiariesLists( )**, η οποία κάνει επαναφορά την **beneficiaryList**.

→ 4 : επιστρέφουμε ένα στάδιο πίσω, δηλαδή καταφεύγουμε στην **adminMonitorMenu( )**.

- **adminMenu( )** : έχει τύπο επιστροφής void και έχει ως όρισμα το στιγμιότυπο **a** τύπου ***Admin***. Στο σώμα της καλούμε την μέθοδο **greet( )** με όρισμα το αντικείμενο ***Admin*** και την συμβολοσειρά ότι έχει την ειδικότητα του διαχειριστή του οργανισμού. Έπειτα μέσω της System.out.println εμφανίζονται οι επιλογές αν θέλουμε : View → 1 Monitor Organization →2 Back → 3 Logout → 4 Exit →5. Δηλώνουμε την ακέραια μεταβλητή **selection** και την αρχικοποιούμε σύμφωνα με την μέθοδο **getMenuOption( )** διοχετεύοντας ως **max** την τιμή 5, καθώς έχουμε 5 επιλογές.

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την **selection**

→ 1 : μεταβαίνουμε στην μέθοδο **adminViewMenu( )**, όπου έχουμε την δυνατότητα να δούμε τις λίστες των ***Materials*** και των ***Services*** και σύμφωνα με το ID να δούμε τις πληροφορίες για το κάθε ένα ***Entity***.

→ 2 : μεταβαίνουμε στην μέθοδο **adminMonitorMenu( )**, όπου έχουμε την δυνατότητα να δούμε τις λίστες των ***Beneficiaries*** και των ***Donators***, να διαγράψουμε στοιχεία από τις λίστες βάσει του τηλεφώνου και να κάνουμε reset στην λίστα των ***Beneficiaries***.

→ 3 : μεταβαίνουμε στην μέθοδο **adminMenu( ),** πάμε ένα επίπεδο πίσω, δηλαδή επιστρέφουμε στην κλάση που βρισκόμαστε και μας ρωτάει ξανά τα ίδια.

→ 4 : μεταβαίνουμε στην μέθοδο **initMenu( )**, προκειμένου να αποσυνδεθούμε.

→ 5 : μέσω του System.exit(0) τερματίζεται το πρόγραμμα και επιστρέφεται τιμή 0.

Αν δεν γίνει κάποια από τις επιλογές 1-2-3-4-5 , ξανακαλείται η μέθοδος **adminMenu( )**.

Beneficiary :

- **beneficiaryRequest( )** : έχει τύπο επιστροφής void και διαθέτει 2 παραμέτρους, το στιγμιότυπο **b** τύπου ***Beneficiary*** και την συμβολοσειρά **s**. Μέσω του System.out.println εμφανίζονται στην οθόνη τα μηνύματα, ώστε να κάνουμε τις επιλογές : αν θέλουμε να πάμε πίσω πρέπει να πατήσουμε το -1, διαφορετικά

να πληκτρολογήσουμε το ID του s που θέλουμε να δούμε. Δημιουργούμε το στιγμιότυπο toOffer τύπου *Material* και το αρχικοποιούμε με το *Entity* που επιστρέφει η getEntity( ) την οποία καλέσαμε μέσω του αντικειμένου org, το οποίο μετατρέψαμε σε *Material*. Δηλώνουμε την ακέραια μεταβλητή bid και την αρχικοποιούμε με την μέθοδο scanNum( ), ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο bid. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1 , επιστρέφουμε ξανά στην beneficiaryAddRequest( ) και εκτελείται από την αρχή η μέθοδος. Μέσω της System.out.println εκτυπώνεται μήνυμα αν ο χρήστης θέλει να κάνει αίτηση και του δίνεται η επιλογή 1 για να κάνει ή η επιλογή 2 για να μην κάνει και να επιστρέψει πίσω. Έπειτα δηλώνουμε την ακέραια μεταβλητή sel και την αρχικοποιούμε σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 2, καθώς έχουμε 2 επιλογές. Μέσω της συνθήκης ελέγχου if-else ελέγχουμε αν ο χρήστης έδωσε για την sel την τιμή 1. Τότε εκτυπώνεται το μήνυμα « Πόση ποσότητα θα χρειαστείτε ?» και δίνουμε την δυνατότητα στο χρήστη είτε να γράψει τον αριθμό της ποσότητας είτε να πατήσει το -1 για να επιστρέψει πίσω. Δηλώνουμε την ακέραια μεταβλητή quan και την αρχικοποιούμε με την μέθοδο scanNum( ), ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο quan. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1 πάμε ένα επίπεδο πίσω στην beneficiaryAddRequest( ). Μέσω του στιγμιοτύπου b της κλάσης *Beneficiary* καλούμε την addReques( ) με ορίσματα το στιγμιότυπο org της κλάσης *Organization*, το στιγμιότυπο toOffer της κλάσης *Entity* και το double πεδίο quan. Μέσω της μεθόδου addRequest( ), δημιουργείται αντικείμενο της κλάσης *RequestDonation*, το οποίο μαζί με το αντικείμενο της Organization προστίθεται στην λίστα requestList. Αν το sel δεν είναι 1, τότε καταφεύγουμε ένα επίπεδο πίσω στη beneficiaryAddRequest( ).

- beneficiaryAddRequest( ) : έχει τύπο επιστροφής void και δέχεται ως παράμετρο το στιγμιότυπο b της κλάσης *Beneficiary*. Στο σώμα της κλάσης χρησιμοποιούμε την System.out.println ώστε να εμφανίσουμε τα μηνύματα για να προσθέσουμε αίτηση 1 : για *Material*, 2 : για *Service* και 3 : για να επιστρέψει πίσω. Δηλώνουμε την ακέραια μεταβλητή selection και την αρχικοποιούμε σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 3, καθώς έχουμε 3 επιλογές.

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την selection :

→ 1 : μέσω του στιγμιοτύπου org καλείται η μέθοδος printListedMaterials( ), η οποία εκτυπώνει τα διαθέσιμα είδη *Material* της currentDonations μαζί με τις ποσότητές τους εκείνη τη στιγμή. Έπειτα καλούμε την μέθοδο beneficiaryRequest( ) με ορίσματα b και material, ώστε να κάνουμε την κατάλληλη αίτηση για τα material, να αναζητήσουμε πληροφορίες για κάποιο material βάσει του ID του και πόση ποσότητα μας είναι απαραίτητη.

→ 2 : μέσω του στιγμιοτύπου org καλείται η μέθοδος printListedServices( ), η οποία εκτυπώνει τα διαθέσιμα είδη *Service* της currentDonations μαζί με τις ποσότητές τους εκείνη τη στιγμή. Έπειτα καλούμε την μέθοδο beneficiaryRequest( ) με ορίσματα b και service, ώστε να κάνουμε την κατάλληλη αίτηση για τα service, να αναζητήσουμε πληροφορίες για κάποιο service βάσει του ID του και πόση ποσότητα μας είναι απαραίτητη.

→ 3 : μας πάει ένα επίπεδο πίσω στην beneficiaryMenu( ).

Αν δεν κάνουμε καμία επιλογή και δεν μπούμε στον βρόγχο ελέγχου switch , επιστρέφουμε ξανά στην beneficiaryAddRequest( ), ώστε να έχουμε την δυνατότητα να κάνουμε πάλι τις επιλογές.

- beneficiaryShowRequests( ) : έχει τύπο επιστροφής void και δέχεται ως όρισμα το στιγμιότυπο b τύπου *Beneficiary*. Στο σώμα της καλούμε την monitor( ), η οποία μας εμφανίζει τα ονόματα και τις ποσότητες των αντικειμένων *Entity* που περιέχονται στην λίστα rdEntities. Η μέθοδος κλήθηκε μέσω της getRequests( ) , η οποία μας επιστρέφει την requestList με αντικείμενα τύπου *Requests*.
- beneficiaryCommit( ) : έχει τύπο επιστροφής void και δέχεται ως όρισμα το στιγμιότυπο b τύπου *Beneficiary*. Στο σώμα της καλείται η μέθοδος commit( ), κατά την οποία γίνεται χειρισμός εξαιρέσεων και ο *Beneficiary* ενημερώνεται για τα αιτήματα που μπορεί να πραγματοποιήσει και ποια όχι.
- beneficiaryMenu( ) : έχει τύπο επιστροφής void και δέχεται ως όρισμα το στιγμιότυπο b τύπου *Beneficiary*. Στο σώμα της καλείται η μέθοδος greet( ) με όρισμα το αντικείμενο *Beneficiary* και την συμβολοσειρά ότι έχει την ειδικότητα του *Beneficiary* του οργανισμού. Έπειτα μέσω της System.out.println εμφανίζονται οι επιλογές αν θέλουμε : Add Request → 1 Show Requests →2 Commit → 3 Back → 4 Logout →5 Exit → 6. Δηλώνουμε την ακέραια μεταβλητή selection και την αρχικοποιούμε σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 6 καθώς έχουμε 6 επιλογές

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την selection

→ 1 : μεταβαίνουμε στην μέθοδο beneficiaryAddRequest( ), όπου έχουμε την δυνατότητα να προσθέσουμε μια αίτηση για κάποιο *Entity* (*Material* ή *Service*).

→ 2 : μεταβαίνουμε στην μέθοδο beneficiaryShowRequests( ), όπου έχουμε την δυνατότητα να δούμε τις δωρεές μέσω του id τους ή να τος διαγράψουμε.

→ 3 : μεταβαίνουμε στην μέθοδο beneficiaryCommit( ), στην οποία γίνεται χειρισμός της εξαίρεσης και πληροφορείται ο χρήστης ποια αιτήματα μπορεί να πραγματοποιήσει.

→ 4 : μεταβαίνουμε στην μέθοδο beneficiaryMenu( ), πάμε ένα επίπεδο πίσω, δηλαδή επιστρέφουμε στην κλάση που βρισκόμαστε και μας ρωτάει ξανά τα ίδια πράγματα.

→ 5 : μεταβαίνουμε στην μέθοδο initMenu( ), προκειμένου να αποσυνδεθούμε.

→ 6 : μέσω του System.exit(0) τερματίζεται το πρόγραμμα και επιστρέφεται τιμή 0.

Αν δεν γίνει κάποια από τις επιλογές 1-2-3-4-5-6 , ξανακαλείται η μέθοδος beneficiaryMenu( ).

Donator :

- donatorAddOffer( ) : Η κλάση αυτή έχει δηλωθεί ως public ,δεν επιστρέφει καμία τιμή , καθώς είναι τύπου void και δέχεται ως όρισμα στιγμιότυπο d τύπου *Donator*.Μέσα στο σώμα της χρησιμοποιούμε την System.out.println ώστε να εμφανίσουμε τα κατάλληλα μηνύματα, προκειμένου να προσθέσουμε την νέα προσφορά στα αντικείμενα *Entity*, καθώς έχουμε και τρίτη επιλογή να επιστρέψουμε πίσω. Δηλώνουμε την ακέραια μεταβλητή *selection* και την αρχικοποιούμε, σύμφωνα με την μέθοδο getMenuOption( ) διοχετεύοντας ως max την τιμή 3, καθώς έχουμε 3 επιλογές.

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την selection

→ 1 : μέσω του αντικειμένου org καλούμε την μέθοδο printListedMaterials( ), ώστε να διατρέχεται η λίστα entityList και αν εντοπιστούν στιγμιότυπα *Material* να εκτυπώνονται οι πληροφορίες τους και οι τρέχουσες ποσότητες τους. Μας

δίνονται οι επιλογές -1, ώστε να επιστρέψουμε ένα επίπεδο πίσω στον κώδικα και η επιλογή να πληκτρολογήσουμε το ID του material για να εμφανιστούν τα χαρακτηριστικά του. Δηλώνουμε την ακέραια μεταβλητή **bid** και την αρχικοποιούμε με την μέθοδο **scanNum( )**, ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο **bid**. Δημιουργούμε το στιγμιότυπο **toOffer** της κλάσης ***Entity*** και το αρχικοποιούμε με την μέθοδο **getEntity( )**, η οποία ελέγχει αν το **bid** βρίσκεται εντός της λίστας **entityList** . Αν όχι προσθέτει το ***Entity*** και την τρέχουσα ποσότητα του στην λίστα. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1 πάμε ένα επίπεδο πίσω στην **donatorAddOffer( )**. Ρωτάμε τον ***Donator*** αν θέλει να κάνει μια νέα προσφορά για το entity, εκτυπώνοντας και τα αντίστοιχα χαρακτηριστικά του **toOffer**. Δηλώνουμε την ακέραια μεταβλητή **sel** και την αρχικοποιούμε σύμφωνα με την μέθοδο **getMenuOption( )** διοχετεύοντας ως **max** την τιμή 2, καθώς έχουμε 2 επιλογές (yes or back). Μέσω του βρόγχου ελέγχου if, αν η επιλογή είναι 1 (ναι) ο δωρητής ρωτάται πόση ποσότητα θέλει να προσφέρει.

Δηλώνουμε την ακέραια μεταβλητή **quan** και την αρχικοποιούμε με την μέθοδο **scanNum( ),** ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο **quan**. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1 πάμε ένα επίπεδο πίσω στην **donatorAddOffer( )**. Μέσω της **addOffer( )** προσθέτουμε ένα νέο στιγμιότυπο ***RequestDonation*** στη λίστα **offersList**.

→ 2 : μέσω του αντικειμένου **org** καλούμε την μέθοδο **printListedServices( )**, ώστε να διατρέχεται η λίστα **entityList** και αν εντοπιστούν στιγμιότυπα της ***Service*** να εκτυπώνονται οι πληροφορίες τους και οι τρέχουσες ποσότητες τους.

→ 3 : πάμε ένα στάδιο πίσω στο πρόγραμμα μεταβαίνουμε στην **donatorMenu( )**.

Αν δεν γίνει κάποια από τις επιλογές 1-2-3 ξανακαλείται η μέθοδος **donatorAddOffer( )**.

- **donatorShowOffers( )** : έχει τύπο επιστροφής void και δέχεται ως όρισμα το στιγμιότυπο **d** τύπου ***Donator***. Στο σώμα της κλάσης καλούμε στο **d** τη μέθοδο **getOffers( )** η οποία επιστρέφει τη λίστα **offersList** και σε αυτή έπειτα καλούμε τη **monitor( )** της ***RequestDonationList*** για να εκτυπωθούν όλα τα αντικείμενα **rd** της λίστας **rdEntities**. Χρησιμοποιούμε την System.out.println ώστε να εμφανίσουμε στον χρήστη τα μηνύματα -1 : για επιστροφή πίσω, -2 : για διαγραφή όλων των δωρεών του και -3 : για να κάνει commit ή να εισάγει το ID του είδους που θέλει

να δει τα στοιχεία. Δηλώνουμε την ακέραια μεταβλητή **oid** και την αρχικοποιούμε με την μέθοδο **scanNum( )**, ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο **oid**. Μέσω της συνθήκη ελέγχου if αν η επιλογή που κάναμε είναι -1 , καλούμε τη μέθοδο **donatorMenu( )** με όρισμα το **d**. Ακόμα,

αν η επιλογή που κάναμε είναι -2 , καλούμε τη μέθοδο **resetOffers( )** στο **d** για να διαγράψουμε τις προσφορές και μετά τη μέθοδο **donatorMenu( )** με όρισμα το **d**. Μέσω της System.out.println εκτυπώνεται μήνυμα αν ο χρήστης θέλει 1 : να διαγράψει τη δωρεά, 2 : να αλλάξει τη ποσότητα της δωρεάς και 3 : να επιστρέψει πίσω. Έπειτα δηλώνουμε την ακέραια μεταβλητή **selc** και την αρχικοποιούμε σύμφωνα με την μέθοδο **scanNum( )**, ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο **selc.** Μέσω της συνθήκης ελέγχου if-else ελέγχουμε αν ο χρήστης έδωσε για την **secl** την τιμή 3. Τότε καλούμε τη μέθοδο **donatorMenu( )** με όρισμα το **d** για να επιστρέψουμε πίσω μια θέση. Αν ο χρήστης έδωσε τιμή 2 μέσω της System.out.println εκτυπώνεται μήνυμα για αλλαγή της ποσότητας ή -1 για επιστροφή πίσω. Δηλώνουμε το double πεδίο **quan** και το αρχικοποιούμε σύμφωνα με την μέθοδο **scanNum( )**, ώστε να επιλέξουμε τον κατάλληλο αριθμό και να τον θέσουμε στο **quan.** Μέσω της συνθήκη ελέγχου if αν η **quan** είναι -1 και αν ναι καλούμε τη μέθοδο **donatorMenu( )** με όρισμα το **d.** Αλλιώς, καλούμε στο **d** την **editOffer( )** με ορίσματα τα **oid** και **quan.** Τέλος, αν για το **selc** ο χρήστης έδωσε τιμή 1 καλούμε τη μέθοδο **removeOffer( )** στο **d** με όρισμα το **oid** και διαγράφουμε αυτή τη δωρεά.

- **donatorCommit( )** : έχει τύπο επιστροφής void και δέχεται ως όρισμα ένα αντικείμενο **d** της *Donator*. Στο σώμα της απλά καλούμε την **commit( )** στο **d**, δηλαδή την **commit( )** της *Donator* πάνω στην **offersList** για να κάνει ο χρήστης commit.
- **donatorMenu( )** : έχει τύπο επιστροφής void και έχει ως όρισμα το στιγμιότυπο **d** τύπου *Donatpr*. Στο σώμα της καλούμε την μέθοδο **greet( )** με όρισμα το αντικείμενο *Donator* και την συμβολοσειρά ότι έχει την ειδικότητα του δωρητή του οργανισμού. Έπειτα μέσω της System.out.println εμφανίζονται οι επιλογές αν θέλουμε : Add Offer → 1 Show Offers →2 Commit → 3 Back → 4 Log out →5 Exit → 6. Δηλώνουμε την ακέραια μεταβλητή **selection** και την αρχικοποιούμε σύμφωνα με την μέθοδο **getMenuOption( )** διοχετεύοντας ως **max** την τιμή 6, καθώς έχουμε 5 επιλογές.

Μέσω της συνθήκης ελέγχου switch ελέγχουμε αν ο χρήστης έδωσε για την selection

→ 1 : μεταβαίνουμε στην μέθοδο donatorAddOffer( ), όπου έχουμε την δυνατότητα να προσθέσουμε μια δωρεά στη λίστα offersList.

→ 2 : μεταβαίνουμε στην μέθοδο donatorShowOffers( ), όπου έχουμε την δυνατότητα να δούμε όλες τις δωρεές του χρήστη – δωρητή και να τις επεξεργαστούμε ανάλογα.

→ 3 : μεταβαίνουμε στην μέθοδο donatorCommit( ), όπου έχουμε την δυνατότητα να κάνουμε commit μια δωρεά.

→ 4 : μεταβαίνουμε στην μέθοδο donatorMenu( ), πάμε ένα επίπεδο πίσω, δηλαδή επιστρέφουμε στην κλάση που βρισκόμαστε και μας ρωτάει ξανά τα ίδια.

→ 5 : μεταβαίνουμε στην μέθοδο initMenu( ), προκειμένου να αποσυνδεθούμε.

→ 6 : μέσω του System.exit(0) τερματίζεται το πρόγραμμα και επιστρέφεται τιμή 0.

Αν δεν γίνει κάποια από τις επιλογές 1-2-3-4-5-6, ξανακαλείται η μέθοδος donatorMenu( ).

Τέλος, ορίζουμε τη μέθοδο initMenu( ) η οποία έχει τύπο επιστροφής void και δεν δέχεται ορίσματα. Εμφανίζει καλωσόρισμα στο menu και δημιουργείται η συμβολοσειρά phone, η οποία αρχικοποιείται με την μέθοδο scanForPhone( ). Έτσι ο χρήστης εισάγει το τηλέφωνο του, το οποίο αποθηκεύεται στην μεταβλητή phone. Δημιουργούμε ένα στιγμιότυπο user της κλάσης User, στο οποίο αρχικοποιούμε τον χρήστη (*Admin* ή *Beneficiary* ή *Donator*), ο οποίος διαθέτει αυτόν τον αριθμό τηλεφώνου. Σε περίπτωση που δεν βρεθεί χρήστης ζητείται αίτημα για εγγραφή, ο χρήστης βάζει το όνομά του δηλώνει τον ρόλο του, ολοκληρώνοντας την εγγραφή του. Στο τέλος μέσω του αντικειμένου user καλείται η showMenu( ) με όρισμα το τρέχον στιγμιότυπο user.

2.7 Exception Error

1. Constructors

Για τη δημιουργία στιγμιότυπου της *Offers* ορίζουμε *public* constructor ο οποίος έχει ως παράμετρο την συμβολοσειρά **hey**. Στο σώμα του εκτυπώνεται το κατάλληλο μήνυμα σφάλματος με βάση το αλφαριθμητικό αυτό που περάσαμε ως παράμετρο.

Πλέον έχουμε προστατέψει τον κώδικα μας από πιθανές εξαιρέσεις, εφόσον δημιουργήσαμε την κλάση *ExceptionError* η οποία τις διαχειρίζεται εκτυπώνοντας μηνύματα σε κρίσιμα σημεία του κώδικα. Έτσι, αντιμετωπίζουμε έμμεσα την εξαίρεση στο σημείο που μπορεί να προκληθεί. Δεν εμποδίζουμε τον χρήστη να εισάγει όποια τιμή θέλει, απλά αν παραδείγματος χάριν ο χρήστης – επωφελούμενος ζητήσει ποσότητα από ένα είδος που υπερβαίνει τη ποσότητα από τα **levels** και αυτό που δικαιούται, έχουμε φροντίσει να εκτυπωθεί το κατάλληλο ενημερωτικό μήνυμα.

2.8 Main

Δημιουργούμε ένα αντικείμενο org τύπου *Organization* σύμφωνα με τον constructor της κλάσης *Organization*. Με την χρήση του στιγμιότυπου org καλούμε την μέθοδο addEntity( ), ώστε να προσθέσουμε τα κατάλληλα είδη διοχετεύοντας και απαραίτητα ορίσματα. Η μέθοδος addEntity( ) έχει δύο ορίσματα ένα τύπου *Entity* (*Material* | *Service*) και ένα double που παριστάνει την ποσότητα του είδους. Στο όρισμα του entity κάθε φορά δημιουργούμε ένα νέο material, σύμφωνα με τον κατασκευαστή της κλάσης *Material* που δέχεται σαν ορίσματα το όνομα του υλικού, την περιγραφή και αρχικοποιεί την ποσότητα των levels που θα παρέχονται ανάλογα με τον αριθμό των ατόμων. Έπειτα δημιουργούμε έναν *Admin* καλώντας την μέθοδο setAdmin( ) στο στιγμιότυπο org διοχετεύοντας ως όρισμα ένα νέο αντικείμενο *Admin*, το οποίο δημιουργήσαμε σύμφωνα με τον constructor της κλάσης *Admin* , διοχετεύοντας ως ορίσματα το όνομα του και το τηλέφωνό του. Στη συνέχεια δημιουργούμε δύο *Beneficiaries*. Αρχικά, δημιουργούμε ένα στιγμιότυπο guy τύπου *Beneficiary*, σύμφωνα με τον constructor της κλάσης *Beneficiary*, διοχετεύοντας ως ορίσματα το όνομά του, το τηλέφωνό του και τον αριθμό των ατόμων που έχει η οικογένειά του. Έπειτα, καλούμε στο αντικείμενο org την μέθοδο insertBeneficiary( ) με όρισμα το guy, ώστε να προσθέσουμε στην λίστα beneficiaryList τον καινούργιο επωφελούμενο. Επιπλέον δημιουργούμε ένα στιγμιότυπο fella τύπου *Beneficiary* πάλι σύμφωνα με τον constructor της κλάσης *Beneficiary*, διοχετεύοντας ως ορίσματα το όνομά του, το τηλέφωνό του και τον αριθμό των ατόμων που έχει η οικογένειά του. Καλούμε πάλι στο αντικείμενο org την μέθοδο insertBeneficiary( ) με όρισμα το fella, ώστε να προσθέσουμε στην λίστα beneficiaryList και τον δεύτερο καινούργιο επωφελούμενο, ο οποίος θα έχει περισσότερο από ένα είδος αιτήματος. Μετέπειτα μέσω του στιγμιότυπου org της κλάσης *Organization* καλείται η μέθοδος insertDonator( ) μέσω από την οποία προσθέτουμε στην λίστα donatorList τον καινούριο δωρητή που κατασκευάσαμε μέσω του constructor της κλάσης *Donator*, με ορίσματα το όνομά και το τηλέφωνό του. Τέλος για να μεταβούμε στο menu του οργανισμού org, χρησιμοποιούμε τον constructor της κλάσης Menu, διοχετεύοντας σαν όρισμα το στιγμιότυπο org, μέσα από το οποίο μεταβαίνουμε στην μέθοδο initMenu( ),που καλωσορίζει τον χρήστη, ζητάει να δώσει τα στοιχεία του και εκτελείται ο υπόλοιπος κώδικας της κλάσης *Menu*.

3 ΠΑΡΑΡΤΗΜΑ

- Πατώντας στο παρακάτω σύνδεσμο μπορείτε να μεταφερθείτε στο private code repository του GitHub με όνομα *Project-code*. Εκεί υπάρχει ο συνολικός κώδικας του project οργανωμένος σε 15 αρχεία, ένα για κάθε κλάση :
  https://github.com/miltiadiss/Project-code/find/main

- Πιο κάτω παρατίθεται και ο συνολικός κώδικας σαν παράρτημα :

A. USER

```
public class User {
 //Initial Variables
 private String name, phone;

 //Constructors
 public User(String name, String phone)
 {
  this.name = name;
  this.phone = phone;
 }

 //Methods
 public String getName()
 {
  return this.name;
 }
 public String getPhone()
 {
  return this.phone;
 }
```

```
 public void showMenu(Menu menu)
 {
  //Depends on user sub-class
 }
 @Override
 public String toString()
 {
  return name + " " + phone;
 }
}
```

B. ADMIN

```
public class Admin extends User {
 //Initial Variables
 boolean isAdmin;

 //Constructors
 public Admin(String name,String phone)
 {
  super(name, phone);
  isAdmin = true;
 }

 //Methods
 public void showMenu(Menu menu)
 {
  menu.adminMenu(this);
 }
}
```

C. DONATOR

```
public class Donator extends User {


 //Initial Variables
 private Offers offersList;


 //Constructors
 public Donator(String name,String phone)
 {
  super(name, phone);
  offersList = new Offers();
 }


 //Methods
 public void addOffer(Entity entity, double quantity)
 {
  RequestDonation rd = new RequestDonation(entity, quantity);
  offersList.add(rd);
 }
 public void resetOffers()
 {
  offersList.reset();
 }
 public void removeOffer(int id)
 {
  offersList.remove(offersList.get(id));
 }
 public void commit()
 {
```

```
 offersList.commit();
}
public void editOffer(int id, double quan)
{
 offersList.modify(offersList.get(id), quan);
}
public void showMenu(Menu menu)
{
 menu.donatorMenu(this);
}

@Override
public String toString()
{
 return super.toString();
}
 //Getters
public Offers getOffers()
{
 return offersList;
}
}
```

D. BENEFICIARY

```
public class Beneficiary extends User {

 //Initial Variables
 private int noPersons;
 private RequestDonationList receivedList;
 private Requests requestList;
```

```
//Constructors
public Beneficiary(String name,String phone)
{
 super(name, phone);
 this.noPersons = 1;
}
public Beneficiary(String name,String phone,int noPersons)
{
 super(name, phone);
 this.noPersons = noPersons;
}

//Methods
public void removeRequest(int id)
{
 requestList.remove(requestList.get(id));
}
public void editRequest(Organization o, int id, double quan)
{
 requestList.modify(o, requestList.get(id), quan, this);
}
public void resetRequests()
{
 requestList.reset();
}
public void reset()
{
 receivedList.reset();
```

```
}
public void addRequest(Organization o, Entity entity, double quantity)
{
 RequestDonation rd = new RequestDonation(entity, quantity);
 requestList.add(o, rd, this);
}
public void commit()
{
 requestList.commit(this);
}
public void showMenu(Menu menu)
{
 menu.beneficiaryMenu(this);
}
 //Getters
public Requests getRequests()
{
 return requestList;
}
public int getNoPersons()
{
 return noPersons;
}
}
```

E. ENTITY

```
public class Entity {

 //Initial Variables
 private static int count = 0;
```

```
private String name, description;
private int id;

//Constructors
public Entity(String name, String description)
{
 this.name = name;
 this.description = description;
 id = count++;
}

//Methods
public String getEntityInfo()
{
 return name+ "("+ id + ") " + description;
}
public String getDetails()
{
 return "";
}

@Override
public String toString() {
 return getEntityInfo() + " - " + getDetails();
}
 //Getters
public int getID()
{
 return id;
```

```
 }
}
```

F. MATERIAL

```
public class Material extends Entity {

 //Initial Variabes
 private double level1, level2, level3;

 //Constructors
 public Material(String name, String description, double level1, double level2, double level3)
 {
  super(name, description);
  this.level1 = level1;
  this.level2 = level2;
  this.level3 = level3;
 }

 //Methods
 public String getDetails()
 {
  return "Material (1: " + level1 + ", 2: " + level2 + ", 3: " + level3 + ")";
 }
  //Getters
 public double getLevel1()
 {
  return level1;
 }
 public double getLevel2()
 {
```

```
 return level2;
}
public double getLevel3()
{
 return level3;
}
}
```

G. SERVICE

```
public class Service extends Entity {

//Constructors
public Service(String name, String description)
{
 super(name, description);
}

//Methods
public String getDetails()
{
 return "Service";
}
}
```

H. REQUESTDONATION

```
public class RequestDonation{

//Initial Variables
private Entity entity;
private double quantity;
private int ID;
```

```
public int getID()
{
 return ID;
}

//Constructors
public RequestDonation(Entity entity, double quantity)
{
 this.entity = entity;
 ID = entity.getID();
 this.quantity = quantity;
}

//Methods
public int compare(RequestDonation rd)
{
 if(rd.getID() == getID())
  return 1;
  return 0;
}

@Override
public String toString()
{
 return ID + ": " + entity.toString() + " x " + quantity;
}
 //Setters
public void setQuantity(double quantity)
```

```
{
 this.quantity = quantity;
}
 //Getters
public double getQuantity()
{
 return quantity;
}
public Entity getEntity()
{
 return entity;
}
}
```

I. ORANIZATION

```
import java.util.ArrayList;

public class Organization {

 //Initial Variables
 String name;
 Admin admin;
 ArrayList<Entity> entityList;
 ArrayList<Donator> donatorList;
 ArrayList<Beneficiary> beneficiaryList;
 RequestDonationList currentDonations;

 //Constructors
 public Organization()
 {
```

```
 entityList = new ArrayList<Entity>();
 donatorList = new ArrayList<Donator>();
 beneficiaryList = new ArrayList<Beneficiary>();
 currentDonations = new RequestDonationList();
}

//Methods
boolean hasEnough(RequestDonation rd)
{
 double quan = rd.getQuantity();
 if(currentDonations.get(rd.getID()).getQuantity() >= quan)
  return true;
 else return false;
}
void resetBeneficiariesLists()
{
 for(Beneficiary b : beneficiaryList)
 {
  b.reset();
 }
}
void listBeneficiaries()
{
 for(Beneficiary b : beneficiaryList)
 {
  System.out.print("\t");
  System.out.println(b.toString());
 }
}
```

```
void listDonators()
{
 for(Donator d : donatorList)
 {
  System.out.print("\t");
  System.out.println(d.toString());
 }
}
void listEntities()
{
 printListedServices();
 printListedMaterials();
}
void printListedServices()
{
 int i = 0;
 System.out.println("Services:");
 for(Entity s : entityList)
 {
  if(s.getDetails().matches("Service"))
  {
   i += 1;
   System.out.println(i + "- " + s.toString() + " / " + currentDonations.get(s.getID()).getQuantity() + "
left") ;
  }
 }
}
void printListedMaterials()
{
 int i = 0;
```

```
 System.out.println("Materials:");
 for(Entity s : entityList)
 {
  if(!s.getDetails().matches("Service"))
  {
   i += 1;
    System.out.println(i + "- " + s.toString() + " / " + currentDonations.get(s.getID()).getQuantity() + "
left") ;
  }
 }
}
 User findUser(String phone)
{
 if(admin.getPhone().matches(phone))
  return admin;
 for(Donator d : donatorList)
  if(d.getPhone().matches(phone))
   return d;
 for(Beneficiary b : beneficiaryList)
  if(b.getPhone().matches(phone))
   return b;

 return null;
}
 void addEntity(Entity e, double q)
{
 for(Entity en : entityList)
 {
  if(en.getEntityInfo().matches(e.getEntityInfo()))
  {
```

```
    new ExceptionError("addEntity error: This entity already exists.");
    return;
   }
  }

  entityList.add(e);
  RequestDonation rd = new RequestDonation(e,q);
  currentDonations.add(rd);
 }
 void removeEntity(int id)
 {
  for(Entity e : entityList)
  {
   if(e.getID() == id) {
    currentDonations.remove(currentDonations.get(id));
    entityList.remove(e);
   }
  }
 }
 void insertDonator(Donator d)
 {
  donatorList.add(d);
 }
 void removeDonator(String phoneNumber)
 {
  for(Donator d : donatorList)
  {
   if(d.getPhone().matches(phoneNumber))
   {
```

```
   donatorList.remove(d);
   break;
  }
 }
}
void insertBeneficiary(Beneficiary b)
{
 beneficiaryList.add(b);
}
void removeBeneficiary(String phoneNumber)
{
 for(Beneficiary b : beneficiaryList)
 {
  if(b.getPhone().matches(phoneNumber))
  {
   donatorList.remove(b);
   break;
  }
 }
}
 //Setters
void setAdmin(Admin a)
{
 admin = a;
}
 //Getters
Admin getAdmin()
{
 return admin;
```

```
}
Entity getEntity(int id)
{
 for(Entity e : entityList)
 {
  if(e.getID() == id)
  {
   return e;
  }
 }

 return null;
}
}
```

J. REQUESTDONATIONLIST

```
import java.util.ArrayList;

public class RequestDonationList {

 //Initial Variables
 private ArrayList<RequestDonation> rdEntities;

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

 //Methods
```

```
public RequestDonation get(int id)
{
 for(RequestDonation rd : rdEntities)
 {
  if(rd.getID() == (id))
   return rd;
 }

 return null;
 // return rdEntities.get(id)
}
public void add(RequestDonation rd)
{
 for(RequestDonation req : rdEntities)
 {
  if(req.equals(rd))
  {
   modify(req, req.getQuantity() + rd.getQuantity());
   return;
  }
 }

 rdEntities.add(rd); //enimerosi
}
public void remove(RequestDonation rd)
{
 rdEntities.remove(rd);
}
public void modify(RequestDonation rd, double quantity)
```

```
{
 rdEntities.get(rd.getID()).setQuantity(quantity);
}
public void monitor()
{
 for(RequestDonation rd : rdEntities)
 {
  System.out.println(rd.toString());
 }
}
public void reset()
{
 rdEntities.clear();
}
 //Getters
public ArrayList<RequestDonation> getrdEntities()
{
 return rdEntities;
}
}
```

K. OFFERS

```
public class Offers extends RequestDonationList {


 //Constructors
 public Offers()
 {
  super();
 }
```

```
 //Methods
 void commit()
 {
  for(RequestDonation rd : getrdEntities())
  {
   System.out.println("Successfully handled " + rd.toString());
   getrdEntities().remove(rd);
  }
 }
}
```

L. REQUESTS

```
public class Requests extends RequestDonationList {


 //Constructors
 public Requests()
 {
  super();
 }


 //Methods
 void add(Organization o, RequestDonation rd, Beneficiary b)
 {
  if(!o.hasEnough(rd))
  {
   new ExceptionError("Organization doesn't have enough of this!");
   return;
  }
  if(validRequestDonation(rd, b))
   super.add(rd); //enimerosi
```

```
 else
  new ExceptionError("You don't have a valid request!");
}
public void modify(Organization o, RequestDonation rd, double quantity, Beneficiary b)
{
 if(!o.hasEnough(rd))
 {
  new ExceptionError("Organization doesn't have enough of this!");
  return;
 }
 if(validRequestDonation(rd, b))
  super.get(rd.getID()).setQuantity(quantity);
 else
  new ExceptionError("You don't have a valid request!");
}
boolean validRequestDonation(RequestDonation rd, Beneficiary b)
{
 Entity e = rd.getEntity();
 if(e.getDetails().matches("Service"))
  return true;

 int noPersons = b.getNoPersons();
 if(noPersons == 1)
  if(rd.getQuantity() <= ((Material) e).getLevel1())
   return true;
 else if(noPersons > 1 && noPersons < 5)
  if(rd.getQuantity() <= ((Material) e).getLevel2())
   return true;
 else if(noPersons > 5)
```

```
  if(rd.getQuantity() <= ((Material) e).getLevel3())
   return true;
 return false;
}
void commit(Beneficiary b)
{
 for(RequestDonation rd : getrdEntities())
 {
  if(validRequestDonation(rd, b))
  {
   System.out.println("Successful handling of " + rd.toString());
   getrdEntities().remove(rd);
  }
 }
}
}
```

M. MENU

```
import java.util.Scanner;


public class Menu {


 //Initial Variables
 private Organization org;


 //Constructors
 public Menu(Organization org)
 {
  this.org = org;
  initMenu();
```

```
}

//Methods
String scanForPhone()
{
 System.out.print("Type your phone number: ");

 Scanner in = new Scanner ( System.in );
 String phone = in.nextLine();

 return phone;
}
String scanForName()
{
 System.out.print("Type your name: ");

 Scanner in = new Scanner ( System.in );
 String name= in.nextLine();

 return name;
}
User completeRegister(Organization org, String name, String phone)
{
 System.out.print("Choose your role, 1-Donator 2-Beneficiary: ");

 Scanner in = new Scanner ( System.in );
 int selection = in.nextInt();

 switch (selection)
```

```
 {
  case 1:
   Donator u = new Donator(name, phone);
   org.insertDonator((Donator) u);
   return u;
  case 2:
   Beneficiary b = new Beneficiary(name, phone);
   org.insertBeneficiary((Beneficiary) b);
   return b;
  default:
   System.out.println("Error choosing role, retrying.");
   return completeRegister(org, name, phone);
 }
}
public void greet(User u, String role)
{
 System.out.println("Welcome " + u.getName() + ", you are a(n) " + role);
}
private int scanNum()
{
 int selection;
 Scanner in = new Scanner ( System.in );
 selection = in.nextInt();
 return selection;
}
private int getMenuOption(int max)
{
 int selection;
 Scanner in = new Scanner ( System.in );
```

```
 selection = in.nextInt();
 if(selection > max || selection < 1)
 {
  System.out.println("You have to give an option from 1 to " + max + "!");
  return getMenuOption(max);
 }
 return selection;
}

 //Admin menu
public void adminViewMenu(Admin a)
{
 System.out.println("View\n\t1-Material\n\t2-Services\n\t3-Back\n");
 int selection = getMenuOption(3);
 switch(selection)
 {
  case 1:
   org.printListedMaterials();
   System.out.println("View\n\tType -1 to go back, or type the ID of a specific material\n");
   int mid = scanNum();
   if(mid == -1) adminViewMenu(a);
    System.out.println(org.getEntity(mid).toString());
    break;
  case 2:
   org.printListedServices();
   System.out.println("View\n\tType -1 to go back, or type the ID of a specific service\n");
   int sid = scanNum();
   if(sid == -1) adminViewMenu(a);
    System.out.println(org.getEntity(sid).toString());
```

```
     break;
   case 3:
    adminMenu(a);
    break;
  }

  adminViewMenu(a);
 }
 public void adminMonitorMenu(Admin a)
 {
  System.out.println("Monitor Organization:\n\t1-List Beneficiaries\n\t2-List Donators\n\t3-Reset
Beneficiaries Lists\n\t4-Back\n");
  int selection = getMenuOption(4);
  switch(selection)
  {
   case 1:
    System.out.println("Listing Beneficiaries");
    org.listBeneficiaries();
    System.out.println("View\n\tType -1 to go back, or type the phone number of a beneficiary to delete
him.\n");
    Integer bid = scanNum();
    if(bid == -1) adminMonitorMenu(a);
     org.removeBeneficiary(bid.toString());
     break;
   case 2:
    System.out.println("Listing Donators");
    org.listDonators();
    System.out.println("View\n\tType -1 to go back, or type the phone number of a donator to delete
him.\n");
    Integer sid = scanNum();
    if(sid == -1) adminMonitorMenu(a);
```

```
    org.removeDonator(sid.toString());
    break;
  case 3:
   org.resetBeneficiariesLists();
   break;
  case 4:
   adminMenu(a);
   break;
 }

 adminMonitorMenu(a);
}
public void adminMenu(Admin a)
{
 greet(a, "admin");
 System.out.println("1-View\n2-Monitor Organization\n3-Back\n4-Logout\n5-Exit\n");
 int selection = getMenuOption(5);
 switch(selection)
 {
  case 1:
   adminViewMenu(a);
   break;
  case 2:
   adminMonitorMenu(a);
   break;
  case 3:
   adminMenu(a);
   break;
  case 4:
```

```
  initMenu();
   break;
  case 5:
   System.exit(0);
   break;
 }

 adminMenu(a);
}

public void beneficiaryRequest(Beneficiary b, String s)
{
 System.out.println("View\n\tType -1 to go back, or type the "+ s +" ID you want to see.\n");
 Integer bid = scanNum();
 Material toOffer = (Material) org.getEntity(bid);
 if(bid == -1) beneficiaryAddRequest(b);
  System.out.println("Do you want to request " + toOffer.getEntityInfo() + "? 1-yes/2-no(Back)");

 int sel = getMenuOption(2);
 if(sel == 1)
 {
  System.out.println("What quantity do you need? (number, type -1 to cancel): ");
  int quan = scanNum();
  if(quan == -1) beneficiaryAddRequest(b);
   b.addRequest(org, toOffer, quan);
 } else beneficiaryAddRequest(b);
}
public void beneficiaryAddRequest(Beneficiary b)
{
```

```
 System.out.println("Add Request\n\t1-Material\n\t2-Services\n\t3-Back");
 int selection = getMenuOption(3);
 switch(selection){
  case 1:
   org.printListedMaterials();
   beneficiaryRequest(b, "material");
   break;
  case 2:
   org.printListedServices();
   beneficiaryRequest(b, "service");
   break;
  case 3:
   beneficiaryMenu(b);
   break;
 }


 beneficiaryAddRequest(b);
}
 public void beneficiaryShowRequests(Beneficiary b)
{
 //Xeirismos twn ekseresewn ths modify
 b.getRequests().monitor();
 System.out.println("View\n\tType -1 to go back, Type -2 to remove all offers, Type -3 to commit, or
type the offer ID you want to see.\n");
 Integer oid = scanNum();
 if(oid == -1) beneficiaryMenu(b);
 if(oid == -2) {
  b.resetRequests();
  beneficiaryMenu(b);
 }
```

```
 System.out.println("View " + oid + " id\n\t1-Delete Offer, 2-Edit Offer Quantity, 3-Back\n");
 int selc = scanNum();
 if(selc == 3) beneficiaryMenu(b);
 else if(selc == 2) {
  System.out.println("\"View \" + oid + \" id -> Edit Quantity (-1 to go back): ");
  int quan = scanNum();
   if(quan == -1) beneficiaryMenu(b);
   else {
    b.editRequest(org, oid, quan);
   }
 } else if(selc == 1)
  {
    b.removeRequest(oid);
  }
}
public void beneficiaryCommit(Beneficiary b)
{
 //Xeirismos ekserseseon wste na typwthoun katallila diagnwstika minimata kai na enimerothei o
 //xristis poia aithmata tou exoun ikanopoihthei epitixws kai poia den pliroun tis proypotheseis
 b.commit();
}
 //Beneficiary menu
public void beneficiaryMenu(Beneficiary b)
{
 greet(b, "beneficiary");
 System.out.println("1-Add Request\n2-Show Requests\n3-Commit\n4-Back\n5-Logout\n6-Exit\n");
 int selection = getMenuOption(6);
 switch(selection)
```

```
 {
  case 1:
   beneficiaryAddRequest(b);
   break;
  case 2:
   beneficiaryShowRequests(b);
   break;
  case 3:
   beneficiaryCommit(b);
   break;
  case 4:
   beneficiaryMenu(b);
   break;
  case 5:
   initMenu();
   break;
  case 6:
   System.exit(0);
   break;
 }

 beneficiaryMenu(b);
}
public void donatorAddOffer(Donator d)
{
 System.out.println("Add Offer\n\t1-Material\n\t2-Services\n\t3-Back");
 int selection = getMenuOption(3);
 switch(selection){
  case 1:
```

```
   org.printListedMaterials();
   System.out.println("View\n\tType -1 to go back, or type the material ID you want to see.\n");
   Integer bid = scanNum();
   Entity toOffer = org.getEntity(bid);
   if(bid == -1) donatorAddOffer(d);
    System.out.println("Do you want to offer " + toOffer.getEntityInfo() + "? 1-yes/2-no(Back)");

   int sel = getMenuOption(2);
   if(sel == 1)
   {
    System.out.println("What quantity do you want to offer? (number, type -1 to cancel): ");
    int quan = scanNum();
    if(quan == -1) donatorAddOffer(d);
     d.addOffer(toOffer, quan);
   } else donatorAddOffer(d);

   break;
  case 2:
   org.printListedServices();
   break;
   case 3:
   donatorMenu(d);
   break;
 }

 donatorAddOffer(d);
}
public void donatorShowOffers(Donator d)
{
```

```
 d.getOffers().monitor();
 System.out.println("View\n\tType -1 to go back, Type -2 to remove all offers, Type -3 to commit, or type the offer ID you want to see.\n");
 Integer oid = scanNum();
 if(oid == -1) donatorMenu(d);
 if(oid == -2) {
  d.resetOffers();
  donatorMenu(d);
 }

 System.out.println("View " + oid + " id\n\t1-Delete Offer, 2-Edit Offer Quantity, 3-Back\n");
 int selc = scanNum();
 if(selc == 3) donatorMenu(d);
 else if(selc == 2) {
  System.out.println("\"View \" + oid + \" id -> Edit Quantity (-1 to go back): ");
  int quan = scanNum();
  if(quan == -1) donatorMenu(d);
  else {
   d.editOffer(oid, quan);
  }
 } else if(selc == 1)
   {
    d.removeOffer(oid);
   }
}
public void donatorCommit(Donator d)
{
 d.commit();
}
 //Donator menu
```

```
public void donatorMenu(Donator d)
{
 greet(d, "donator");
 System.out.println("1-Add Offer\n2-Show Offers\n3-Commit\n4-Back\n5-Logout\n6-Exit\n");
 int selection = getMenuOption(6);
 switch(selection)
 {
  case 1:
   donatorAddOffer(d);
   break;
  case 2:
   donatorShowOffers(d);
   break;
  case 3:
   donatorCommit(d);
   break;
  case 4:
   donatorMenu(d);
   break;
  case 5:
   initMenu();
   break;
  case 6:
   System.exit(0);
   break;
 }

 donatorMenu(d);
}
```

```
 public void initMenu()
 {
  System.out.println("Welcome to the menu\n");
  String phone = scanForPhone();
  User user = org.findUser(phone);

  //Register in case of not found user.
  if(user == null)
  {
   System.out.println("Phone not found, please register.");
   String name = scanForName();
   user = completeRegister(org, name, phone);
  }

  user.showMenu(this);
 }
}
```

N. EXCEPTIONERROR

```
class ExceptionError {

 //Constructors
 public ExceptionError(String hey) {
  System.out.println("Error: " + hey);
 }
}
```

O. MAIN

```
public class Main {

 //Methods
```

```
public static void main(String[] args) {
 Organization org = new Organization();
 //Add the entities requested
 org.addEntity(new Material("milk", "Your favourite dairy product!", 1, 2.5, 4), 100);
 org.addEntity(new Material("sugar", "Don't overeat it!", 0.5, 1, 2.5), 100);
 org.addEntity(new Material("rice", "Chinese food?", 2, 5, 10), 100);
 org.addEntity(new Service("MedicalSupport", "Dentist and More Doctors"), 100);
 org.addEntity(new Service("NurserySupport", "Nursing and Care"), 100);
 org.addEntity(new Service("BabySitting", "Caring for Children"), 100);

 //Set an admin
 org.setAdmin(new Admin("Admin", "911"));

 //Add two beneficiaries, the one with more than 1 type of request
 Beneficiary guy = new Beneficiary("Guy", "100", 1);
 org.insertBeneficiary(guy);

 Beneficiary fella = new Beneficiary("Fella", "166", 2);
 org.insertBeneficiary(fella);

 //Add a donator and give him supply
 org.insertDonator(new Donator("Elon Musk", "200"));

 new Menu(org);
 }
}
```