Αναφορά Εργασίας Οντοκεντρικού Προγρ/σμου 2020-2021

Όνομα: Δημήτριος
Επίθετο: Στασινός
ΑΜ: 1084643
Email: up1084643@upnet.gr

Όνομα: Αλέξανδρος
Επίθετο: Καρράς
ΑΜ: 1084645
Email: up1084645@upnet.gr

Σύνδεσμος Google Drive

https://drive.google.com/drive/folders/1OO4fyOhtXD7A0XAK5mKOpWtDGPFPZVp-?usp=sharing

Κλάση User

Η κλάση αυτή αναπαριστά έναν χρήστη του προγράμματος. Αρχικά ορίζουμε την κλάση ως abstract αφού δεν θα δημιουργήσουμε στιγμιότυπα της. Οι δύο μεταβλητές, phone και name , είναι private ώστε να έχουμε πρόσβαση μόνο από τις μεθόδους getName και getPhone οι οποίες επιστρέφουν τις τιμές αυτών των μεταβλητών. Επίσης οι μεταβλητές είναι και final ώστε να μην μπορούν να αλλαχτούν οι τιμές τους. Ο constructor δέχεται δύο ορίσματα ώστε να γίνεται η αρχικοποίηση των μεταβλητών κατά την δημιουργία αντικείμενων των υποκλάσεων της. Η μέθοδος getUserInfo επιστρέφει κατάλληλα μορφοποιημένα τις πληροφορίες των μεταβλητών, υπάρχει η δήλωση της getDetails (abstract) η οποία υλοποιείτε στις υποκλάσεις και τέλος η toString καλεί τις δύο μεθόδους και τυπώνει κατάλληλα μορφοποιημένες τις συνολικές διαθέσιμες πληροφορίες.

Κλάση Admin

Η κλάση αυτή είναι υποκλάση της User και έχει επιπλέων μια private μεταβλητή isAdmin. Αυτή είναι final ώστε να μην μπορεί να γίνει αλλαγή της τιμής της, η οποία είναι πάντα true και έχουμε πρόσβαση σε αυτή μέσα από την μέθοδο isAdmin η οποία επιστρέφει την τιμή της. Ο constructor δέχεται δύο ορίσματα και μέσα από το super χρησιμοποιεί τον constructor της User ώστε να αρχικοποιείσει τις μεταβλητές. Επιπλέον θέτει την μεταβλητή isAdmin true και τέλος η getDetails υλοποιείτε επιστρέφοντας το αλφαριθμητικό “Admin”.

Κλάση Beneficiary

Η κλάση αυτή είναι υποκλάση της User και επιπλέον έχει: μια private και final μεταβλητή noPersons η οποία δηλώνει πόσα άτομα έχει η οικογένεια, ένα πεδίο receivedList τύπου RequestDonationList το οποίο περιέχει τα αντικείμενα και τις ποσότητες που έχει λάβει και ένα πεδίο requestsList τύπου Request η οποία περιέχει τα αντικείμενα που ζητά να λάβει ο Beneficiary. Επιπλέον υπάρχουν δύο constructor που με την χρήση της super χρησιμοποιούν τον constructor της User ώστε να αρχικοποιείσει τις μεταβλητές. Οι διαφορά μεταξύ τους είναι ότι ο ένας δέχεται τρία ορίσματα και μπορεί να θέσει τιμές και στις τρεις μεταβλητές (name, phone, noPersons) ενώ ο άλλος δέχεται δύο μεταβλητές και ορίζει αυτόματα την noPersons=1. Τέλος η μέθοδος getNoPersons επιστρέφει την τιμή της μεταβλητής noPersons και η getDetail υλοποιείτε επιστρέφοντας το αλφαριθμητικό “Beneficiary”.

Κλάση Donator

Η κλάση αυτή είναι υποκλάση της User και επιπλέον έχει ένα πεδίο offersList τύπου Offers στο οποίο περιέχονται τα είδη που θέλει να δωρίσει ο Donator. Ο constructor δέχεται δύο ορίσματα και με το super αρχικοποιεί τις μεταβλητές που υπάρχουν στην υπερκλάση User. Τέλος η getDetails υλοποιείτε επιστρέφοντας το αλφαριθμητικό “Donator”.

Κλάση Entity

Η κλάση αυτή αναπαριστά ένα είδος που είναι διαθέσιμο στον οργανισμό. Δεν θα δημιουργήσουμε αντικείμενα αυτής της κλάσεις και έτσι την ορίζουμε ως abstract. Περιέχει τρεις private και final μεταβλητές, name, description, id και έναν constructor ο οποίος δέχεται τρία ορίσματα και θέτει τιμές στις μεταβλητές κατά την δημιουργία των αντικειμένων των υποκλάσεων. Ακόμα οι μέθοδοι getName και getId επιστρέφουν τις τιμές των δύο μεταβλητών (name, id) αφού δεν έχουμε απευθείας πρόσβαση σε αυτές. Η μέθοδος getEntityInfo επιστρέφει κατάλληλα μορφοποιημένα τις πληροφορίες των μεταβλητών, υπάρχει η δήλωση της getDetails (abstract) η οποία υλοποιείτε στις υποκλάσεις και τέλος η toString καλεί τις δύο μεθόδους και τυπώνει κατάλληλα μορφοποιημένες τις συνολικές διαθέσιμες πληροφορίες.

Κλάση Material

Η κλάση αυτή είναι υποκλάση της User και επιπλέον έχει τρεις private και final μεταβλητές level1, level2, level3 οι οποίες αντιπροσωπεύουν την ποσότητα που δικαιούται ένας Beneficiary από ένα Material. Ο constructor δέχεται έξι ορίσματα και μέσα από το super αρχικοποιεί τις τρεις μεταβλητές της Entity και τέλος τις τρεις μεταβλητές της Material. Ακόμα οι μέθοδοι getlevel1, getlevel2, getlevel3 επιστρέφουν την τιμή που έχουμε θέσει σε αυτές τις μεταβλητές και τέλος υλοποιείτε η getDetails η οποία επιστρέφει κατάλληλα μορφοποιημένα την τιμή κάθε level με την χρήση των getters και ότι το αντικείμενο είναι Material.

Κλάση Service

Η κλάση αυτή είναι υποκλάση της User. Έχει ένα constructor ο οποίος δέχεται τρία ορίσματα και αρχικοποιεί με την χρήση του super τις μεταβλητές που κληρονομεί από την Entity. Τέλος υλοποιείτε η getDetails η οποία επιστρέφει το αλφαριθμητικό “Service”.

Κλάση CustomException

Είναι υποκλάση της κλάσης Exception και περιέχει έναν constructor ο οποίος παίρνει ως όρισμα μια ακολουθία χαρακτήρων. Με αυτήν την κλάση διαχειριζόμαστε τα exception που μπορεί να προκύψουν από την λάθος χρήση του προγράμματος από τους χρήστες ή την μη

δυνατότητα ολοκλήρωσης μια ενέργειας. Κάθε φορά που γίνεται throw εμφανίζεται και το κατάλληλο μήνυμα χωρίς να σταματήσει η λειτουργία του προγράμματος.

Κλάση Organization

Η κλάση αυτή αναπαριστά τον οργανισμό μέσα από τον οποίο θα διακινούνται τα Materials, Services και είναι εγγεγραμμένοι οι donators και beneficiaries . Έχει την μεταβλητή name, private και final και την admin private. Ακόμα έχει μια λίστα entityList τύπου Entity η οποία περιέχει τα είδη που διακινούνται στον οργανισμό, μια λίστα donatorList τύπου Donator η οποία περιέχει τους donators οι οποίοι είναι εγγεγραμμένοι στον οργανισμό, μια λίστα beneficiaryList τύπου Beneficiary η οποία περιέχει τους beneficiary που είναι εγγεγραμμένοι στον οργανισμό και τέλος ένα πεδίο currantDonations τύπου RequestDonations το οποίο θα περιλαμβάνει τις διαθέσιμες προσφορές και ποσότητες των ειδών. Έπειτα έχει ένα constructor ο οποίος δέχεται δύο ορίσματα και αρχικοποιεί τις μεταβλητές name και admin. Οι μέθοδοι getName και getAdmin επιστρέφουν τις τιμές των μεταβλητών (name, admin) και η setAdmin καταχωρεί έναν Admin στον οργανισμό.

Μέθοδος addEntity (void)

Παίρνει ως όρισμα ένα Entity και περιέχει μια μεταβλητή search (boolean) η οποία είναι false. Αρχικά γίνεται έλεγχος αν υπάρχει το αντικείμενο στην entityList με την βοήθεια της getId (Entity) και αν βρεθεί ίδιο id, αφού είναι μοναδικό για κάθε αντικείμενο, η μεταβλητή γίνεται true, ο έλεγχος σταματάει, γίνεται διαχείριση της εξαίρεσης και εμφανίζεται μήνυμα προς τον χρήστη. Εάν δεν βρεθεί ίδιο id η search παραμένει false και το αντικείμενο προστίθεται στην λίστα entityList.

Μέθοδος removeEntity (void)

Παίρνει δύο ορίσματα, Entiy και Admin. Κάνει έλεγχο για το αν ο χρήστης είναι admin με την μέθοδο isAdmin (Admin) και αν ο έλεγχος είναι true γίνεται αφαίρεση του entity από την λίστα. Αλλιώς προκύπτει εξαίρεση που εμφανίζει κατάλληλο μήνυμα.

Μέθοδος insertDonator (void)

Παίρνει ως όρισμα έναν Donator και έχει μεταβλητές boolean οι οποίες είναι false. Αρχικά γίνεται έλεγχος για το αν υπάρχει ο donator στην donatorList με την χρήση της μεθόδου getPhone (User) και συγκρίνει τα τηλέφωνα των εγγεγραμμένων με αυτόν που θέλουμε να προσθέσουμε. Έπειτα ελέγχει στην beneficiaryList για παρόμοιο τηλέφωνο και τέλος τον admin του οργανισμού. Η σύγκριση αλφαριθμητικών γίνεται με το equal και αν βρεθεί ταυτόσημο τηλέφωνο η boolean μεταβλητή γίνεται true και ο έλεγχος σταματάει. Γίνεται throw η Custom exception και τέλος γίνεται διαχείριση της και εμφανίζεται κατάλληλο μήνυμα. Εάν δεν βρεθεί ταυτόσημος user, οι boolean μεταβλητές παραμένουν false και ο donator προστίθεται στην λίστα.

Μέθοδος removeDonator (void)

Παίρνει δύο ορίσματα έναν Donator και έναν Admin. Κάνει έλεγχο για το αν ο χρήστης είναι admin με την μέθοδο isAdmin (Admin) και αν ο έλεγχος είναι true γίνεται αφαίρεση του donator από την λίστα. Αλλιώς προκύπτει εξαίρεση και γίνεται διαχείριση της εμφανίζοντας κατάλληλο μήνυμα.

Μέθοδος insertBeneficiary (void)

Παίρνει ως όρισμα έναν Beneficiary και έχει μεταβλητές boolean οι οποίες είναι false. Αρχικά γίνεται έλεγχος για το αν υπάρχει ο beneficiary στην beneficiaryList με την χρήση της μεθόδου getPhone (User) και συγκρίνει τα τηλέφωνα των εγγεγραμμένων με αυτόν που θέλουμε να προσθέσουμε. Έπειτα ελέγχει στην donatorList για παρόμοιο τηλέφωνο και τέλος τον admin του οργανισμού. Η σύγκριση αλφαριθμητικών γίνεται με το equal και αν βρεθεί ταυτόσημο τηλέφωνο η boolean μεταβλητή γίνεται true και ο έλεγχος σταματάει. Γίνεται throw η Custom exception και τέλος γίνεται διαχείριση της και εμφανίζεται κατάλληλο μήνυμα. Εάν δεν βρεθεί ταυτόσημος user, οι boolean μεταβλητές παραμένουν false και ο beneficiary προστίθεται στην λίστα.

Μέθοδος removeBeneficiary (void)

Παίρνει δύο ορίσματα έναν Beneficiary και έναν Admin. Κάνει έλεγχο για το αν ο χρήστης είναι admin με την μέθοδο isAdmin (Admin) και αν ο έλεγχος είναι αληθής γίνεται αφαίρεση του Beneficiary από την λίστα. Αλλιώς προκύπτει εξαίρεση και γίνεται διαχείριση της εμφανίζοντας κατάλληλο μήνυμα.

Μέθοδος listEntities (int)

Παίρνει ως όρισμα ένα String, περιέχει δύο μεταβλητές και επιστρέφει μια ακέραια μεταβλητή. Εάν το όρισμα String είναι “1” εμφανίζει μόνο τα materials του οργανισμού με τον έλεγχο ότι το Entity είναι Material αλλιώς αν το όρισμα είναι “2” εμφανίζει τα services του οργανισμού. Επίσης σε κάθε Entity που εμφανίζει, δείχνει και την ποσότητα που περιέχετε στον οργανισμό καλώντας την getQuantity του οργανισμού. Τέλος σε κάθε εύρεση κάποιου Entity το count αυξάνεται κατά ένα και τέλος αυτή η τιμή επιστρέφεται.

Μέθοδος getMaterial (Material)

Παίρνει ως όρισμα ένα Entity και έχει μια μεταβλητή Material. Αυτή ελέγχει κάθε Entity της entityList του οργανισμού και όταν βρει το ίδιο Id με αυτό που δώσαμε αποθηκεύει το Entity στο material κάνοντας το cast. Τέλος επιστρέφει το material που βρήκε.

Μέθοδος getEntity (Entity)

Παίρνει ως όρισμα ένα String και ένα int. Ακόμα περιλαμβάνει μια μεταβλητή Entity και μια ακέραια μεταβλητή count. Εάν το String ισούται με “1” γίνεται έλεγχος στην entityList του οργανισμού και βρίσκει μόνο τα Materials ενώ εάν ισούται με “2” βρίσκει μόνο τα Services. Ανάλογα με τι αναζητεί, Material ή Service, όταν βρει κάποιο αυξάνει το count κατά ένα και

αν αυτό ισούται με τον ακέραιο που έχει δοθεί ως όρισμα, αποθηκεύει το συγκεκριμένο Entity στην μεταβλητή entity και την επιστρέφει.

Μέθοδος getQuantity (double)

Παίρνει ως όρισμα ένα Entity και περιλαμβάνει δύο μεταβλητές quantity (double) και search (boolean). Αυτή η μέθοδος ελέγχει την currentDonations του οργανισμού για να βρει το Entity που του δώσαμε ως όρισμα. Αν τη λίστα που περιέχει η currentDonations έχει μέγεθος 0 δηλαδή δεν περιέχει κάποιο αντικείμενο, το quantity παίρνει την τιμή 0. Αλλιώς ψάχνει την λίστα και αν βρει το αντικείμενο με την χρήση της getId (User) αποθηκεύει την ποσότητα στην quantity με την βοήθεια της getQuantity (RequestDonationsList) η search παίρνει τιμή true και ο έλεγχος σταματάει. Αν δεν βρεθεί το αντικείμενο το quantity παίρνει τιμή 0 και τέλος επιστρέφει αυτήν την τιμή.

Μέθοδος listBeneficiaries (int)

Έχει μια μεταβλητή count (int) ώστε να εμφανίζει αριθμημένα τους Beneficiaries. Αν η λίστα που περιέχει τους Beneficiaries του οργανισμού δεν έχει μέγεθος ίσο με 0, θα εμφανίζει αριθμημένα τα ονόματά τους και θα αυξάνει το count κατά ένα. Εάν η λίστα έχει μέγεθος ίσο με 0 θα εμφανίζει κατάλληλο μήνυμα και τέλος επιστρέφει το count.

Μέθοδος getBeneficiary (Beneficiary)

Έχει μια μεταβλητή Beneficiary και έχει ως όρισμα ένα ακέραιο αριθμό. Η μέθοδος αυτή παίρνει το στοιχείο της beneficiaryList που βρίσκεται στην θέση του ακεραίου που παίρνει ως όρισμα μείον ένα, τον αποθηκεύει στην μεταβλητή beneficiary και την επιστρέφει. Αυτό συμβαίνει γιατί η λίστα ξεκινάει από την θέση 0 ενώ η αρίθμηση στον χρήστη από τον αριθμό 1. Έτσι για να τρέξει σωστά η μέθοδος πρέπει από την επιλογή του χρήστη να αφαιρέσουμε 1. Ακόμα υπερφορτώνουμε την μέθοδο που τώρα παίρνει ως όρισμα ένα String και έχει μια μεταβλητή beneficiary. Αυτή η μέθοδος ψάχνει την beneficiaryList και κάνει έλεγχο ώστε να βρει τον Beneficiary με το ίδιο τηλέφωνο τον οποίο όταν βρει τον αποθηκεύει στην μεταβλητή beneficiary και την επιστρέφει.

Μέθοδος listDonators (int)

Έχει μια μεταβλητή count (int) ώστε να εμφανίζει αριθμημένα τους Donators. Αν η λίστα που περιέχει τους Donators του οργανισμού δεν έχει μέγεθος ίσο με 0, θα εμφανίζει αριθμημένα τα ονόματά τους και θα αυξάνει το count κατά ένα. Εάν η λίστα έχει μέγεθος ίσο με 0 θα εμφανίζει κατάλληλο μήνυμα και τέλος επιστρέφει το count.

Μέθοδος getDonator (Donator)

Έχει μια μεταβλητή Donator και έχει ως όρισμα ένα ακέραιο αριθμό. Η μέθοδος αυτή παίρνει το στοιχείο της donatorList που βρίσκεται στην θέση του ακεραίου που παίρνει ως όρισμα μείον ένα, τον αποθηκεύει στην μεταβλητή donator και την επιστρέφει. Αυτό συμβαίνει γιατί η λίστα ξεκινάει από την θέση 0 ενώ η αρίθμηση στον χρήστη από τον αριθμό 1. Έτσι για να τρέξει σωστά η μέθοδος πρέπει από την επιλογή του χρήστη να αφαιρέσουμε 1. Ακόμα

υπερφορτώνουμε την μέθοδο που τώρα παίρνει ως όρισμα ένα String και έχει μια μεταβλητή donator. Αυτή η μέθοδος ψάχνει την donatorList και κάνει έλεγχο ώστε να βρει τον Donator με το ίδιο τηλέφωνο τον οποίο όταν βρει τον αποθηκεύει στην μεταβλητή donator και την επιστρέφει.

Μέθοδος resetBeneficiariesLists (void)

Για κάθε beneficiary που βρίσκεται στην beneficiaryList καθαρίζει την rdEntities λίστα της receivedList και τέλος εμφανίζει μήνυμα ότι η διαδικασία ολοκληρώθηκε.

Κλάση RequestDonation

Αναπαριστά ένα αίτημα για δωρεά ή παροχή σε κάποιον User που είναι εγγεγραμμένος στον οργανισμό. Έχει ένα πεδίο entity τύπου Entity και μια private μεταβλητή quantity. Ο constructor δέχεται δύο ορίσματα και αρχικοποιεί τις μεταβλητές. Η μέθοδος getEntity επιστρέφει το Entity , η getQuantity επιστρέφει το quantity και η setQuantity δέχεται ένα double όρισμα και το αποθηκεύει στην μεταβλητή quantity.

Κλάση RequestDonationList

Η κλάση αυτή περιέχει μια λίστα τύπου RequestDonation και μεθόδους ώστε να γίνεται διαχείριση αυτής της λίστας.

Μέθοδος add (void)

Δέχεται ως όρισμα ένα RequestDonation και περιέχει μια μεταβλητή search (boolean) με τιμή false. Αν η λίστα έχει μέγεθος ίσο με 0 τότε προσθέτει το requestDonation αλλιώς ψάχνει την λίστα μέχρι να βρει το συγκεκριμένο requestDonation. Ο έλεγχος γίνεται για κάθε αντικείμενο με την βοήθεια της getEntity (RequestDonation) και της getId (Entity). Όταν ο έλεγχος γίνει αληθής, θέτουμε search ίσο με true και για το αντικείμενο που βρέθηκε στην λίστα καλείτε η setQuantity (RequestDonation) και βάζουμε ως όρισμα σε αυτήν, την τιμή που είχε το συγκεκριμένο requestDonation με την χρήση της getQuantity (RequestDonation) και προσθέτουμε το quantity που έχει η RequestDonation που περάσαμε ως όρισμα με την χρήση της getQuantity (RequestDonation). Αν δεν βρεθεί το requestDonation στη λίστα το search παραμένει false και το requestDonation προστίθεται στην λίστα.

Μέθοδος monitor (int)

Η μέθοδος αυτή υπερφορτώνεται παίρνοντας την μια φορά ως όρισμα έναν Donator και την άλλη έναν Beneficiary. Και οι δύο έχουν μια ακέραια μεταβλητή count ίση με 1. Αν η λίστα rdEntities της κλάσης για τον κάθε User έχει μέγεθος διάφορο του 0 εμφανίζει κατάλληλα μηνύματα και αριθμημένα τα ονόματα των Entities μαζί με την ποσότητα που ζητάνε να δώσουν ή να παραλάβουν. Αλλιώς αν το μέγεθος της λίστας είναι ίσο με 0 εμφανίζει κάθε φορά κατάλληλο μήνυμα.

Μέθοδος get(RequestDonation)

Δέχεται ως όρισμα έναν ακέραιο αριθμό και επιστρέφει το requestDonation που βρίσκεται στην θέση του ακέραιου αριθμού μείον ένα. Αυτό συμβαίνει γιατί η λίστα ξεκινάει από την θέση 0 ενώ η αρίθμηση στον χρήστη από τον αριθμό 1. Έτσι για να τρέξει σωστά η μέθοδος πρέπει από την επιλογή του χρήστη να αφαιρέσουμε 1.

Μέθοδος remove (RequestDonation)

Δέχεται ως όρισμα ένα requestDonation και το αφαιρεί από την λίστα rdEntities.

Μέθοδος modify (void)

Δέχεται ως όρισμα δύο ακέραιους αριθμούς και παίρνει από την rdEntities το στοιχείο που βρίσκεται στην θέση του ενός ακεραίου που δόθηκε ως όρισμα πλην ένα και σε αυτό καλείτε η setQuantity (RequestDonation) και θέτει την ποσότητα του Entity ίσο με το δεύτερο όρισμα (quantity).

Μέθοδος reset (void)

Καθαρίζει την λίστα rdEntities

Κλάση Offers

Είναι υποκλάση της RequestDonationList και έχει μια επιπλέον μέθοδο. Περιέχει τις δωρεές που Θέλει να πραγματοποιήσει ένας Donator.

Μέθοδος commit (void)

Έχει μια μεταβλητή search (boolean) και δέχεται ως όρισμα ένα Organization. Αρχικά αν η λίστα rdEntities της currentDonations του organization έχει μέγεθος ίσο με 0 τοποθετεί όλα τα στοιχεία της rdEntities του Donator στην rdEntities της currentDonations του οργανισμού. Αλλιώς για κάθε στοιχείο που περιέχει η λίστα που κληρονόμησε από την RequestDonationList ελέγχει την λίστα της currentDonation του οργανισμού και αν βρεθεί το ίδιο Entity με την βοήθεια των id προσθέτει το quantity του στοιχείου στην λίστα του οργανισμού. Αν δεν βρεθεί, το search παραμένει false και το requestDonation προστίθεται στην currentDonation. Τέλος καθαρίζει την rdEntities του Donator και εμφανίζει μήνυμα ότι η ενέργεια ολοκληρώθηκε.

Κλάση Requests

Είναι υποκλάση της RequestDonationList και περιέχει το σύνολο των ειδών που ζητά ο Beneficiary.

Μέθοδος add (void)

Έχει δύο μεταβλητές approval_1 και approval_2 (boolean) δηλωμένες ως false και δέχεται τρία ορίσματα (RequestDonation, Organization, Beneficiary). Αρχικά αν το Entity του requestDonation είναι Material ψάχνει να το βρει στην rdEntities του currentDonations και όταν βρεθεί αλλάζει την τιμή της approval_1 μόνο όταν η ποσότητα που ζητάει είναι μικρότερη ή ίση με αυτή που υπάρχει στον οργανισμό αλλιώς προκύπτει εξαίρεση που εμφανίζει κατάλληλο μήνυμα. Εάν περάσει τον πρώτο έλεγχο καλείτε η μέθοδος validRequestDonation (με τα τρία ορίσματα) και αν δεν προκύψει κάποιο exception και να το κάνει διαχείριση καλείτε η add που κληρονομεί από την υπερκλάση και προσθέτει το requestDonation στην λίστα. Αν το Entity είναι Service το πρόγραμμα ελέγχει μόνο εάν η ποσότητα που ζητάει είναι διαθέσιμη στον οργανισμό αλλιώς κάνει throw CustomException με το κατάλληλο μήνυμα.

Μέθοδος validRequestDonation (void)

Την μέθοδο αυτή την υπερφορτώνουμε. Η πρώτη δέχεται τρία ορίσματα (Organization, Beneficiary, RequestDonation), έχει μεταβλητές δύο boolean approval και search, μια double level και ένα Material στο οποίο αποθηκεύεται το material που περιέχει το requestDonation. Τώρα ελέγχει τα άτομα που έχει η οικογένεια του beneficiary και έτσι αποθηκεύει το σωστό level του Material. Έπειτα ψάχνει στα αντικείμενα που έχει παραλάβει και αν βρει το συγκεκριμένο αντικείμενο ελέγχει αν η ποσότητα που έχει συν αυτή που ζητάει δεν ξεπερνάνε το όριο και έτσι περνάει τον έλεγχο. Αν δεν βρει το αντικείμενο τότε ελέγχει αν δικαιούται αυτήν την ποσότητα, το approval γίνεται true και δεν προκύπτει κάποιο exception. Αν δεν δικαιούται την συγκεκριμένη ποσότητα γίνεται throw custom exception με κατάλληλο μήνυμα. Η δεύτερη δέχεται τέσσερα ορίσματα (Organization, Beneficiary, RequestDonation, int), έχει δύο μεταβλητές approval (boolean) και level (double) , βρίσκει το material του requestDonation και ελέγχει αν την καινούργια ποσότητα που ζητάει ο Beneficiary είναι μέσα στα όρια που του επιτρέπεται. Εάν δεν την δικαιούται προκύπτει εξαίρεση με κατάλληλο μήνυμα.

Μέθοδος modify (void)

Έχει τρεις μεταβλητές approval_1=false, approval_2=true και approval_3=false και δέχεται τέσσερα ορίσματα (RequestDonation, Organization, Beneficiary, int). Αρχικά ελέγχει αν το requestDonation είναι Material ή Service. Αν είναι Material ψάχνει στην currentDonation του οργανισμού να το βρει μέσω της getId (Entity) και ελέγχει αν η νέα ποσότητα που ζητάει είναι διαθέσιμη. Εάν δεν είναι διαθέσιμη προκύπτει εξαίρεση αλλιώς ο έλεγχος προχωράει στην validRequestDonation (με τα τέσσερα ορίσματα) και διαχειρίζεται το exception που μπορεί να προκύψει. Εάν περάσει και τους δύο ελέγχους τότε τροποποιείτε η ποσότητα του είδους που ζητάει ο Beneficiary. Αν το requestDonation είναι Service ελέγχει μόνο αν η ποσότητα που ζητάει είναι διαθέσιμη στον οργανισμό και ανανεώνει το requestDoantion αλλιώς προκύπτει εξαίρεση.

Μέθοδος commit(void)

Δέχεται δύο ορίσματα (Beneficiary, Organization), έχει δύο μεταβλητές Boolean approval_1=false και approval_2=true. Ακόμα έχει μια λίστα τύπου RequestDonation στην οποία τοποθετούνται τα request που έχουν γίνει δεκτά. Τώρα για κάθε στοιχείο που βρίσκεται στην requestsList του beneficiary ελέγχει πάλι αν δικαιούται αυτήν την ποσότητα και αν περάσει τους ελέγχους το στοιχείο προστίθεται στην receivedList του beneficiary, η ποσότητα αφαιρείτε από τον οργανισμό, το RequestDonation προστίθεται στην λίστα approved και εμφανίζεται κατάλληλο μήνυμα. Εάν δεν περάσει τους ελέγχους το approval_1 παραμένει false και το approval_2 γίνεται false, εμφανίζοντας κατάλληλο μήνυμα. Τέλος για κάθε RequestDonation που έγινε δεκτό και είναι αποθηκευμένο στην approved, διαγράφεται από την requestsList του Beneficiary.

Κλάση Menu

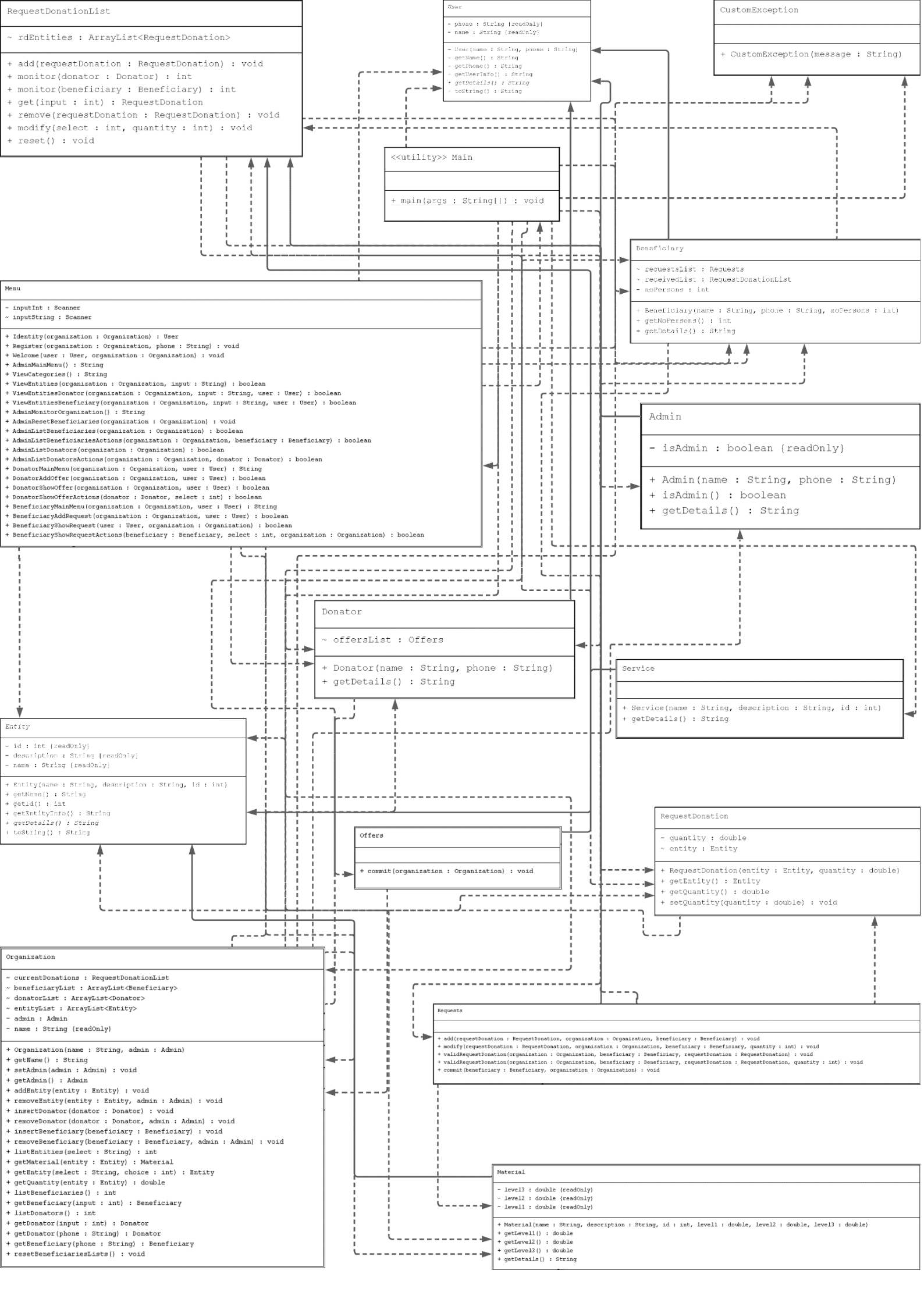
Περιέχει δύο αντικείμενα τύπου Scanner, ένα για ακεραίους και έναν για String. Η μέθοδος Identity διαχειρίζεται την ταυτοποίηση κάθε χρήστη και βρίσκει τον συγκεκριμένο User. Εάν δεν τον βρει καλεί την μέθοδο Register ώστε να μπορέσει να εγγραφεί. Εκεί του δίνεται η επιλογή να εγγραφεί ως Donator ή Beneficiary και προστίθεται στον οργανισμό. Σε κάθε περίπτωση που ζητείται από τον χρήστη να εισάγει κάτι, γίνεται έλεγχος για την σωστή εισαγωγή αλλιώς γίνεται διαχείριση εξαίρεσης και μέσα από βρόχο do-while ζητάει πάλι να εισάγει σωστά δεδομένα. Η περιήγηση στα επίπεδα του menu γίνεται με τον χρήστη να διαλέγει το ανάλογο αριθμό που εμφανίζεται στην κονσόλα και με τον βρόγχο do-while να πηγαίνει προς τα πίσω ή να παραμένει στο ίδιο επίπεδο. Κάθε επίπεδο έχει την δικιά του μέθοδο ώστε να είναι εύκολη η διαχείριση του menu και τέλος κάθε μέθοδος έχει τις μεταβλητές που χρειάζεται και τα ορίσματα ώστε να εμφανίζονται σωστά τα δεδομένα και να είναι εύκολη η ανάγνωσή τους από τον χρήστη.

Κλάση main

Στην κλάση αυτή δημιουργούμε τα αντικείμενα και καλούμε τις μεθόδους ώστε το πρόγραμμα να είναι λειτουργικό και να περιέχει αρκετά δεδομένα. Αρχικά φτιάχνουμε ένα αντικείμενο Admin και θέτουμε τα ορίσματα που χρειάζεται ώστε όταν δημιουργήσουμε αντικείμενο του Organization να τον βάλουμε ως όρισμα. Επιπλέον δημιουργούμε τα Materials και τα τοποθετούμε στον οργανισμό. Το ίδιο κάνουμε και για τα Services και τοποθετούμε κάποια από αυτά στην λίστα currentDonations του οργανισμού ώστε να υπάρχουν διαθέσιμες ποσότητες. Έπειτα φτιάχνουμε δυο Beneficiaries, τους τοποθετούμε στον οργανισμό και δημιουργούμε Requests για έναν από αυτόν. Τέλος δημιουργούμε έναν donator, τον τοποθετούμε στον οργανισμό και του προσθέτουμε παροχές. Για να αρχίσει το πρόγραμμα πρέπει να δημιουργήσουμε ένα αντικείμενο της menu και να καλέσουμε τις μεθόδους της. Πρώτα καλούμε από την menu την Identity η οποία επιστρέφει τον User και από την main γίνεται ο έλεγχος για το αν είναι Admin,donator ή Beneficiary εμφανίζοντας κατάλληλο μήνυμα καλωσορίσματος. Αν ο User είναι Admin, όλοι οι μέθοδοι της menu καλούνται από την Main και εκεί γίνεται η διαχείριση των εξαιρέσεων. Για τους Donator και Beneficiary καλείτε μόνο η DonatorMainMenu ή BeneficiaryMainMenu αντίστοιχα και μέσα

από αυτές καλούνται οι επόμενοι μέθοδοι ώστε ο χρήστης να μπορεί να περιηγείται στο menu. Τέλος κάθε φορά που αποσυνδέεται ένας χρήστης το πρόγραμμα επιστρέφει στο Login(μέθοδο Idenity) και περιμένει να του δοθεί κάποιο τηλέφωνο ώστε να συνδεθεί κάποιος χρήστης και αν γίνει τερματισμός προγράμματος εμφανίζεται κατάλληλο μήνυμα.

Διάγραμμα κλάσεων (UML)


RequestDonationList
~ rdEntities : ArrayList<RequestDonation>
+ add(requestDonation : RequestDonation) : void
+ monitor(donator : Donator) : int
+ monitor(beneficiary : Beneficiary) : int
+ get(input : int) : RequestDonation
+ remove(requestDonation : RequestDonation) : void
+ modify(select : int, quantity : int) : void
+ reset() : void
User
- phone : String {readOnly}
- name : String {readOnly}
- User(name : String, phone : String)
- getName() : String
- getPhone() : String
- getUserInfo() : String
+ getDetails() : String
- toString() : String
CustomException
+ CustomException(message : String)
<<utility>> Main
+ main(args : String[]) : void
Beneficiary
~ requestsList : Requests
~ receivedList : RequestDonationList
- noPersons : int
+ Beneficiary(name : String, phone : String, noPersons : int)
+ getNoPersons() : int
+ getDetails() : String
Menu
~ inputInt : Scanner
~ inputString : Scanner
+ Identity(organization : Organization) : User
+ Register(organization : Organization, phone : String) : void
+ Welcome(user : User, organization : Organization) : void
+ AdminMainMenu() : String
+ ViewCategories() : String
+ ViewEntities(organization : Organization, input : String) : boolean
+ ViewEntitiesDonator(organization : Organization, input : String, user : User) : boolean
+ ViewEntitiesBeneficiary(organization : Organization, input : String, user : User) : boolean
+ AdminMonitorOrganization() : String
+ AdminResetBeneficiaries(organization : Organization) : void
+ AdminListBeneficiaries(organization : Organization) : boolean
+ AdminListBeneficiariesActions(organization : Organization, beneficiary : Beneficiary) : boolean
+ AdminListDonators(organization : Organization) : boolean
+ AdminListDonatorsActions(organization : Organization, donator : Donator) : boolean
+ DonatorMainMenu(organization : Organization, user : User) : String
+ DonatorAddOffer(organization : Organization, user : User) : boolean
+ DonatorShowOffer(organization : Organization, user : User) : boolean
+ DonatorShowOfferActions(donator : Donator, select : int) : boolean
+ BeneficiaryMainMenu(organization : Organization, user : User) : String
+ BeneficiaryAddRequest(organization : Organization, user : User) : boolean
+ BeneficiaryShowRequest(user : User, organization : Organization) : boolean
+ BeneficiaryShowRequestActions(beneficiary : Beneficiary, select : int, organization : Organization) : boolean
Admin
- isAdmin : boolean {readOnly}
+ Admin(name : String, phone : String)
+ isAdmin() : boolean
+ getDetails() : String
Donator
~ offersList : Offers
+ Donator(name : String, phone : String)
+ getDetails() : String
Service
+ Service(name : String, description : String, id : int)
+ getDetails() : String
Entity
- id : int {readOnly}
- description : String {readOnly}
- name : String {readOnly}
+ Entity(name : String, description : String, id : int)
+ getName() : String
+ getId() : int
+ getEntityInfo() : String
+ getDetails() : String
- toString() : String
Offers
+ commit(organization : Organization) : void
RequestDonation
- quantity : double
~ entity : Entity
+ RequestDonation(entity : Entity, quantity : double)
+ getEntity() : Entity
+ getQuantity() : double
+ setQuantity(quantity : double) : void
Organization
~ currentDonations : RequestDonationList
~ beneficiaryList : ArrayList<Beneficiary>
~ donatorList : ArrayList<Donator>
~ entityList : ArrayList<Entity>
~ admin : Admin
- name : String {readOnly}
+ Organization(name : String, admin : Admin)
+ getName() : String
+ setAdmin(admin : Admin) : void
+ getAdmin() : Admin
+ addEntity(entity : Entity) : void
+ removeEntity(entity : Entity, admin : Admin) : void
+ insertDonator(donator : Donator) : void
+ removeDonator(donator : Donator, admin : Admin) : void
+ insertBeneficiary(beneficiary : Beneficiary) : void
+ removeBeneficiary(beneficiary : Beneficiary, admin : Admin) : void
+ listEntities(select : String) : int
+ getMaterial(entity : Entity) : Material
+ getEntity(select : String, choice : int) : Entity
+ getQuantity(entity : Entity) : double
+ listBeneficiaries() : int
+ getBeneficiary(input : int) : Beneficiary
+ listDonators() : int
+ getDonator(input : int) : Donator
+ getDonator(phone : String) : Donator
+ getBeneficiary(phone : String) : Beneficiary
+ resetBeneficiariesLists() : void
Requests
+ add(requestDonation : RequestDonation, organization : Organization, beneficiary : Beneficiary) : void
+ modify(requestDonation : RequestDonation, organization : Organization, beneficiary : Beneficiary, quantity : int) : void
+ validRequestDonation(organization : Organization, beneficiary : Beneficiary, requestDonation : RequestDonation) : void
+ validRequestDonation(organization : Organization, beneficiary : Beneficiary, requestDonation : RequestDonation, quantity : int) : void
+ commit(beneficiary : Beneficiary, organization : Organization) : void
Material
- level3 : double {readOnly}
- level2 : double {readOnly}
- level1 : double {readOnly}
+ Material(name : String, description : String, id : int, level1 : double, level2 : double, level3 : double)
+ getLevel1() : double
+ getLevel2() : double
+ getLevel3() : double
+ getDetails() : String

Κώδικας Project

Κλάση User

```
public abstract class User {

   //Ιδιωτικές και σταθερές μεταβλητές
   private final String name;
   private final String phone;

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

   //Μέθοδος που επιστρέφει το όνομα του χρήστη
   public String getName() {
      return name;
   }

   //Μέθοδος που επιστρέφει το τηλέφωνο του χρήστη
   public String getPhone() {
      return phone;
   }

   /*Μέθοδος που επιστρέφει κατάλληλα μορφοποιημένα
   το όνομα και τηλέφωνο του χρήστη*/
   public String getUserInfo() {
      return "Name: " + name + "\nMobile Phone: " + phone;
   }

   //Δήλωση μεθόδου
   public abstract String getDetails();

   /*Μέθοδος που επιστρέφει κατάλληλα μορφοποιημένα
   το σύνολο των διαθέσιμων πληροφοριών του χρήστη*/
   public String toString() {
      return getDetails() + "\n" + getUserInfo();
   }

}
```

Κλάση Admin

```
public class Admin extends User{

   //Ιδιωτική και σταθερή μεταβλητή
   private final boolean isAdmin;

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

   /*Μέθοδος που επιστρέφει
   την τιμή της μεταβλητής isAdmin*/
   public boolean isAdmin() {
      return isAdmin;
   }

   //Ηλοποίηση της μεθόδου
   public String getDetails(){
      return "Admin";
   }

}
```

Κλάση Beneficiary

```
public class Beneficiary extends User{

   //Ιδιωτική μεταβλητή
   private int noPersons=1;

   /*Δημιουργία αντικειμένου της RequestDonationList που αναπαραστά τα
   είδη που έχει παραλάβει ο Beneficiary*/
   RequestDonationList receivedList = new RequestDonationList();

   /*Δημιουργία αντικειμένου της Requests που αναπαραστά
   τα είδη που θέλει να παραλάβει ο Beneficiary*/
   Requests requestsList = new Requests();

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

   /*Μέθοδος που επιστρέφει
   την τιμή της μεταβλητής noPersons*/
   public int getNoPersons() {
      return noPersons;
   }
```

```
    //Ηλοποίηση της μεθόδου
    public String getDetails(){
        return "Beneficiary";
    }
}
```

Κλάση Donator

```
public class Donator extends User {

    /*Δημιουργία αντικειμένου της Offers
    που αναπαραστά τα είδη που θέλει
    να δωρήσει ο Donator*/
    Offers offersList= new Offers();

    //constructor
    public Donator(String name, String phone) {
        super(name, phone);
    }

    //Ηλοποίηση της μεθόδου
    public String getDetails(){
        return "Donator";
    }
}
```

Κλάση Entity

```
public abstract class Entity {

    //Ιδιωτικές και σταθερές μεταβλητές
    private final String name;
    private final String description;
    private final int id;

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

    /*Μέθοδος που επιστρέφει
    το όνομα του είδους*/
    public String getName() {
        return name;
    }

    /*Μέθοδος που επιστρέφει
    το Id του είδους*/
    public int getId() {
```

```
        return id;
    }

    /*Μέθοδος που επιστρέφει κατάλληλα μορφοποιημένα
    το όνομα, περιγραφή και Id του είδους*/
    public String getEntityInfo() {
        return "Name: " + name + "\nDescription: " + description + "\nID: " + id;
    }

    //Δήλωση μεθόδου
    public abstract String getDetails();

    /*Μέθοδος που επιστρέφει κατάλληλα μορφοποιημένα
    το σύνολο των διαθέσιμων πληροφοριών του είδους*/
    public String toString() {
        return getEntityInfo() + "\n" + getDetails();
    }


}
```

Κλάση Material

```
public class Material extends Entity {

    //Ιδιωτικές και σταθερές μεταβλητές
    private final double level1;
    private final double level2;
    private final double level3;

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

    /*Μέθοδος που επιστρέφει την τιμή
    του level1*/
    public double getLevel1() {
        return level1;
    }

    /*Μέθοδος που επιστρέφει την τιμή
    του level2*/
    public double getLevel2() {
        return level2;
    }

    /*Μέθοδος που επιστρέφει την τιμή
    του level3*/
```

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

    //Ηλοποίηση της μεθόδου
    public String getDetails() {
        return "Level1=" + getLevel1() + "\tLevel2=" + getLevel2() + "\tLevel3=" + getLevel3() + "\nMaterial";
    }
}
```

Κλάση Service

```
public class Service extends  Entity{

    //constructor
    public Service(String name, String description, int id) {
        super(name, description, id);
    }

    //Ηλοποίηση της μεθόδου
    public String getDetails(){
        return "Service";
    }

}
```

Κλάση CustomException

```
class CustomException extends Exception {

    public CustomException(String message) {
        super(message);
    }
}
```

Κλάση RequestDonation

```
public class RequestDonation {

    Entity entity;
    private double quantity;

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

    //Μέθοδος που επιστρέφει το Entity
    public Entity getEntity() {
        return entity;
```

```
    }

    //Μέθοδος που επιστρέφει το quantity
    public double getQuantity() {
        return quantity;
    }

    //Μέθοδος που θέτει νέα τιμή στο quantity
    public void setQuantity(double quantity) {
        this.quantity = quantity;
    }
}
```

Κλάση RequestDoantionList

```
import java.util.ArrayList;

public class RequestDonationList {

   //Λίστα που περιέχει αντικείμενα τύπου RequestDonation
   ArrayList<RequestDonation> rdEntities = new ArrayList<>();

   //Μέθοδος η οποία τοποθετεί ένα συγκεκριμένο requestDonation στην rdEntities
   public void add(RequestDonation requestDonation) {
      boolean search = false;

      if (rdEntities.size() == 0) {                  //Αν το μέγεθος της λίστας
         rdEntities.add(requestDonation);//είναι 0 προσθέτει το RequestDonation
      } else {
         for (var v : rdEntities) { //Έλεγχος για το αν το RequestDonation υπάρχει στην λίστα
            if (v.getEntity().getId() == requestDonation.getEntity().getId()) {
               v.setQuantity(v.getQuantity() + requestDonation.getQuantity());
               search = true;  //Προσθέτει την επιπλέον ποσότητα
            }
         }
         if (!search) { //Αν δεν βρεθεί το αντικείμενο,
                        //προσθέτει στην λίστα το RequestDonation
            rdEntities.add(requestDonation);
         }
      }
   }

   /*Μέθοδος που εμφανίζει τα είδη που βρίσκονται στην λίστα μαζί
   με τις ποσότητες τους για έναν Donator*/
   public int monitor(Donator donator) {
      int count = 1;
      if (rdEntities.size() != 0) {
         System.out.println(donator.getName() + " offers:"); //Εμφάνιση των ειδών
         for (var v : rdEntities) {                          //με κατάλληλη μορφωποίηση
            System.out.println(count + "." + v.getEntity().getName() + " [" + v.getQuantity() + "]");
            count++;
         }
      } else { //Εμφάνιση μηνύματος εάν η λίστα έχει μέγεθος 0
         System.out.println("You are not offering something");
      }
      return count; //Επιστρέφει τον αριθμό των ειδών που περιέχει η λίστα
   }

   /*Μέθοδος που εμφανίζει τα είδη που βρίσκονται στην λίστα μαζί
   με τις ποσότητες τους για έναν Beneficiary*/
   public int monitor(Beneficiary beneficiary) {
      int count = 1;
      if (rdEntities.size() != 0) {
         System.out.println(beneficiary.getName() + " requests:"); //Εμφάνιση των ειδών με
         for (var v : rdEntities) {                                //κατάλληλη μορφωποίηση
```

```
            System.out.println(count + "." + v.getEntity().getName() + " [" + v.getQuantity() + "]");
            count++;
        }
    } else { //Εμφάνιση μηνύματος εάν η λίστα έχει μέγεθος 0
        System.out.println("You are not requesting something");
    }
    return count; //Επιστρέφει τον αριθμό των ειδών που περιέχει η λίστα
}

/*Μέθοδος που επιστρέφει ένα RequestDonation
ανάλογα με την επιλογή του χρήστη*/
public RequestDonation get(int input) {
    return rdEntities.get(input - 1); //Η λίστα ξεκινάει απο την θέση 0,
}                                     //επιστροφή του RequestDonation

//Μέθοδος που διαγράφει ένα RequestDonation από την λίστα
public void remove(RequestDonation requestDonation) {
    rdEntities.remove(requestDonation);
}

/*Μέθοδος που τροποποιεί την ποσότητα ενός
RequestDonation που βρίσκεται στην λίστα*/
public void modify(int select, int quantity) {          //Επιλέγει ένα RequestDonation από την λίστα
    rdEntities.get(select - 1).setQuantity(quantity); //και καλεί μέθοδο ώστε να θέσει την καινούργια ποσότητα
}

//Μέθοδος που καθαρίζει την λίστα
public void reset() {
    rdEntities.clear();
}

}
```

Κλάση Requests

```
import java.util.ArrayList;

public class Requests extends RequestDonationList {

    //Μέθοδος που ελέγχει αν ο Beneficiary δικαιούτε τα είδη
    public void add(RequestDonation requestDonation, Organization organization, Beneficiary beneficiary) throws
CustomException {
        boolean approval_1 = false, approval_2 = false;

        //Ελεγχος αν το requestDonation είναι Material
        if (!(requestDonation.getEntity() instanceof Material)) {
            for (var v : organization.currentDonations.rdEntities) {
                if (requestDonation.getEntity().getId() == v.getEntity().getId()) {
                    if (requestDonation.getQuantity() <= v.getQuantity()) {
                        approval_1 = true;    //Πρώτος έλεγχος για το αν η ποσότητα που
                    }                         //ζητάει ο Beneficiary είναι διαθέσιμη στον οργανισμό
```

```
            }
        }
        if (!approval_1) { //Εάν αποτύχει ο πρώτος έλεγχος προκύπτει
                    // εξαίρεση που εμφανίζει κατάλληλο μήνυμα
            throw new CustomException("The quantity you are requesting is not available");
        }

        //Δεύτερος έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
        try {
            validRequestDonation(organization, beneficiary, requestDonation);
        } catch (CustomException e) {
            System.out.println(e.getMessage()); //Διαχείρηση της εξαίρεσης που μπορεί να προκύψει
            System.out.println("----------------------------");
        }
        add(requestDonation); //Εάν περάσει και τους δύο ελέγχους καλείτε η add της RequestDonationList
    } else { //Εάν το requestDonation είναι Service
        for (var v : organization.currentDonations.rdEntities) {
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) {
                if (requestDonation.getQuantity() <= v.getQuantity()) { //Έλεγχος για το αν η ποσότητα που
                    add(requestDonation);                    //ζητάει ο Beneficiary είναι
                    approval_2 = true;                       //διαθέσιμη στον οργανισμό
                }
            }
        }
        if (!approval_2) { //Εάν αποτύχει ο έλεγχος προκύπτει εξαίρεση που εμφανίζει κατάλληλο μήνυμα
            throw new CustomException("The quantity you are requesting is not available\nPlease enter another one");
        }
    }
  }

  //Μέθοδος που ελέγχει αν ο Beneficiary δικαιούτε την νέα ποσότητα που ζητάει
  public void modify(RequestDonation requestDonation, Organization organization, Beneficiary beneficiary, int
quantity) throws CustomException {
    boolean approval_1 = false, approval_3 = false, approval_2 = true;

    //Έλεγχος αν το requestDonation είναι Material
    if(requestDonation.getEntity() instanceof Material){
        for (var v : organization.currentDonations.rdEntities) {
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) { //Πρώτος έλεγχος για το αν η ποσότητα που
                if (quantity <= v.getQuantity()) {                  //ζητάει ο Beneficiary είναι
                    approval_1 = true;                              //διαθέσιμη στον οργανισμό
                }
            }
        }
        if (!approval_1) {  //Εάν αποτύχει ο πρώτος έλεγχος προκύπτει εξαίρεση που εμφανίζει κατάλληλο μήνυμα
            throw new CustomException("The quantity you are requesting is not available\nPlease enter another one");
        }

        //Δεύτερος έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
        try {
            validRequestDonation(organization, beneficiary, requestDonation, quantity);
        } catch (CustomException e) {
            approval_2 = false;
```

```
            System.out.println(e.getMessage()); //Διαχείρηση της εξαίρεσης που μπορεί να προκύψει
            System.out.println("----------------------------");
        }
        if (approval_2) {
            requestDonation.setQuantity(quantity); //Εάν περάσει και τους δύο ελέγχους καλείτε μέθοδος για να αναθέσει
        }                                   //την νέα τιμή στο quantity
    } else { //Εάν το requestDonation είναι Service
        for (var v : organization.currentDonations.rdEntities) {
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) {
                if (quantity <= v.getQuantity()) {  //Έλεγχος για το αν η ποσότητα που
                                                    //ζητάει ο Beneficiary είναι διαθέσιμη στον οργανισμό
                    requestDonation.setQuantity(quantity);
                    approval_3 = true;
                }
            }
        }
        if (!approval_3) { //Εάν αποτύχει ο έλεγχος προκύπτει εξαίρεση που εμφανίζει κατάλληλο μήνυμα
            throw new CustomException("The quantity you are requesting is not available\nPlease enter another one");
        }
    }
}

//Μέθοδοι που ελέγχουν αν η ποσότητα που ζητάει ο Beneficiary είναι μέσα στο επιτρεπτό όριο ανάλογα με τον αριθμό των ατόμων της οικογένειας
//Καλέιται στην add
public void validRequestDonation(Organization organization, Beneficiary beneficiary, RequestDonation requestDonation) throws CustomException {
    boolean approval = false, search = false;
    double level;
    Material material = organization.getMaterial(requestDonation.getEntity());

    //Έλεγχος για το μέγεθος της οικογέγειας
    if (beneficiary.getNoPersons() == 1) {
        level = material.getLevel1(); //Αποθηκεύεται το level του Material
        for (var v : beneficiary.receivedList.rdEntities) {
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
                if (v.getQuantity() + requestDonation.getQuantity() <= level) {
                    approval = true;
                    search = true;
                }
            }
        }
        if (!search) {
            if (requestDonation.getQuantity() <= level) {
                approval = true;
            }
        }
    } else if (beneficiary.getNoPersons() >= 2 && beneficiary.getNoPersons() <= 4) {
        level = material.getLevel2(); //Αποθηκεύεται το level του Material
        for (var v : beneficiary.receivedList.rdEntities) {
```

```
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
                if (v.getQuantity() + requestDonation.getQuantity() <= level) {
                    approval = true;
                    search = true;
                }
            }
        }
        if (!search) {
            if (requestDonation.getQuantity() <= level) {
                approval = true;
            }
        }
    } else if (beneficiary.getNoPersons() >= 5) {
        level = material.getLevel3(); //Αποθηκεύεται το level του Material
        for (var v : beneficiary.receivedList.rdEntities) {
            if (requestDonation.getEntity().getId() == v.getEntity().getId()) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
                if (v.getQuantity() + requestDonation.getQuantity() <= level) {
                    approval = true;
                    search = true;
                }
            }
        }
        if (!search) {
            if (requestDonation.getQuantity() <= level) {
                approval = true;
            }
        }
    }
    if (!approval) { //Προκύπτει εξαίρεση με κατάλληλο μήνυμα εάν δεν δικαιούτε την ποσότητα
        throw new CustomException("You are no valid for this quantity");
    }

}

//Καλείται στην modify
public void validRequestDonation(Organization organization, Beneficiary beneficiary, RequestDonation requestDonation, int quantity) throws CustomException {
    boolean approval = false;
    double level;
    Material material = organization.getMaterial(requestDonation.getEntity());

    //Έλεγχος για το μέγεθος της οικογένειας
    if (beneficiary.getNoPersons() == 1) {
        level = material.getLevel1(); //Αποθηκεύεται το level του Material
        if (quantity <= level) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
            approval = true;

        }
    } else if (beneficiary.getNoPersons() >= 2 && beneficiary.getNoPersons() <= 4) {
        level = material.getLevel2(); //Αποθηκεύεται το level του Material
        if (quantity <= level) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
```

```
            approval = true;
        }
    } else if (beneficiary.getNoPersons() >= 5) {
        level = material.getLevel3(); //Αποθηκεύεται το level του Material
        if (quantity <= level) { //Έλεγχος για το αν δικαιούτε την ποσότητα που ζητάει ο Beneficiary
            approval = true;

        }
    }
    if (!approval) { //Προκύπτει εξαίρεση με κατάλληλο μήνυμα εάν δεν δικαιούτε την ποσότητα
        throw new CustomException("You are no valid for this quantity");
    }

}

//Μέθοδος που ενημερώνει τα currentDonations του οργανισμού ανάλογα με τα είδη που ζητάει ο Beneficiary
public void commit(Beneficiary beneficiary, Organization organization) {
    boolean approval_1, approval_2;
    ArrayList<RequestDonation> approved = new ArrayList<>();

    //Ελέγχει αν ο Beneficiary δικαούτε την ποσότητα που ζητάει
    if (beneficiary.requestsList.rdEntities.size() != 0) {
        for (var v : beneficiary.requestsList.rdEntities) {
            approval_1 = false;
            approval_2 = true;
            if (v.getEntity() instanceof Material) {
                for (var t : organization.currentDonations.rdEntities) {
                    if (v.getEntity().getId() == t.getEntity().getId()) {
                        try {
                            if (v.getQuantity() <= t.getQuantity()) {
                                approval_1 = true;
                            }
                            validRequestDonation(organization, beneficiary, v);
                        } catch (CustomException e) {
                            approval_2 = false;
                        }
                        if (approval_1 && approval_2) {        //Εάν περάσει τους ελέγχους εμφανίζει κατάλληλο μήνυμα
                            beneficiary.receivedList.add(v);    //και αφαιρεί την ποσότητα από τον οργανισμό
                            approved.add(v);
                            t.setQuantity(t.getQuantity() - v.getQuantity());
                            System.out.println("-> " + v.getEntity().getName() + " (Request approved) \u2714\n");
                        }
                    }
                }
            } else {
                for (var k : organization.currentDonations.rdEntities) {
                    if (v.getEntity().getId() == k.getEntity().getId()) {
                        if (v.getQuantity() <= k.getQuantity()) {
                            beneficiary.receivedList.add(v);
                            k.setQuantity(k.getQuantity() - v.getQuantity()); //Εάν περάσει τους ελέγχουσ εμφανίζει κατάλληλο
μήνυμα
                            approved.add(v);                            //και αφαιρεί την ποσότητα από τον οργανισμό
                            System.out.println("-> " + v.getEntity().getName() + " (Request approved) \u2714\n");
```

```
                    approval_1 = true;
                    approval_2 = true;
                }
            }
        }
        //Εμφανίζει κατάλληλο μήνυμα ανάλογα με τον έλεγχο που απέτυχε
        if (!approval_1 && approval_2) {
            System.out.println("-> " + v.getEntity().getName() + " (Request rejected) X\nThe quantity you are requesting is not available\n");
        } else if (approval_1 && !approval_2) {
            System.out.println("-> " + v.getEntity().getName() + " (Request rejected) X\nYou are no valid for this quantity\n");
        } else if (!approval_1) {
            System.out.println("-> " + v.getEntity().getName() + " (Request rejected) X\nThe quantity you are requesting is not available\nYou are no valid for this quantity\n");
        }
      }
    } else { //Εμφανίζει μήνυμα εάν δεν ζητάει τίποτα
      System.out.println("You have no request to commit");
    }

    for (var v : approved) {
      beneficiary.requestsList.rdEntities.remove(v); //Αφαίρεση όσον Requests πραγματοποιήθηκαν με επιτυχία
    }

    approved.clear();
    System.out.println("----------------------------");
  }

}
```

Κλάση Offers

```
public class Offers extends RequestDonationList {

  /*Μέθοδος που Ενημερώνει τα currentDonations του οργανισμού
  με τις προσφορές που περιέχονται στη λίστα rdEntities*/
  public void commit(Organization organization) {
    boolean search;


    if (rdEntities.size() != 0) {
      if (organization.currentDonations.rdEntities.size() == 0) {
        organization.currentDonations.rdEntities.addAll(rdEntities);
      } else {
        for (var v : rdEntities) {
          search = false;
          for (var t : organization.currentDonations.rdEntities) {  //Ελεγχος για το αν υπάρχει στην currentDonation
            if (v.getEntity().getId() == t.getEntity().getId()) { //το RequestDonation

              t.setQuantity(t.getQuantity() + v.getQuantity()); //Ανάθεση νέας τιμής στην ποσότητα
```

```
                    search = true;
                }
            }
            if (!search) { //Εάν δεν υπάρχει προσθέτει το RequestDonation
                organization.currentDonations.rdEntities.add(v);
            }
        }
    }
    rdEntities.clear(); //Καθαρισμός λίστας και εμφάνιση μηνύματος ότι η ενέργεια ολοκληρώθηκε
    System.out.println("Your offers have been delivered");
    System.out.println("----------------------------");
  } else{ //Εμφάνιση μηνύματος εάν η λίστα έχει μέγεθος 0
    System.out.println("You have no offers to deliver");
    System.out.println("----------------------------");
  }

 }

}
```

Κλάση Organization

```
import java.util.ArrayList;

public class Organization {

  //Ιδιωτική και σταθερή μεταβλητή
  private final String name;

  //Ιδιωτική μεταβλητή
  private Admin admin;

  /*Δημιουργία λίστας τύπου Entity που περιέχει
  τα είδη που διακινούνται στον οργανισμό*/
  ArrayList<Entity> entityList = new ArrayList<>();

  /*Δημιουργία λίστας τύπου Donator που περιέχει
  τους Donators του οργανισμού*/
  ArrayList<Donator> donatorList = new ArrayList<>();

  /*Δημιουργία λίστας τύπου Beneficiary που περιέχει
  τους Beneficiaries του οργανισμού*/
  ArrayList<Beneficiary> beneficiaryList = new ArrayList<>();
  RequestDonationList currentDonations = new RequestDonationList();

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

  /*Μέθοδος που επιστρέφει
```

```
το όνομα του οργανισμού*/
public String getName() {
   return name;
}

//Μέθοδος που θέτει έναν Admin στον οργανισμό
public void setAdmin(Admin admin) {
   this.admin = admin;
}

//Μέθοδος που επιστρέφει τον Admin του οργανισμού
public Admin getAdmin() {
   return admin;
}

//Μέθοδος που προσθέτει ένα είδος στον οργανισμό
public void addEntity(Entity entity) {
   boolean search = false;

   try { //Έλεγχος για το αν υπάρχει το συγκεκριμένο είδος στον οργανισμό
      for (var v : entityList) {
         if (v.getId() == entity.getId()) { //Εύρεση μέσω του Id
            search = true;
            break;
         }
      }
      if (!search) {
         entityList.add(entity); //Προσθήκη είδους εάν δεν υπάρχει αλλιώς προκύπτει εξαίρεση
      } else throw new CustomException("The " + entity.getName() + " already exists in the organization");
   } catch (CustomException e) { //Διαχείρηση εξαίρεσης
      System.out.println(e.getMessage());
   }
}

//Μέθοδος που αφαιρεί είδος απο τον οργανισμό
public void removeEntity(Entity entity, Admin admin) {
   try {
      if (admin.isAdmin()) { //Έλεγχος για το αν ο χρήστης είναι Admin αλλιώς προκύπτει εξαίρεση
         entityList.remove(entity); //Αφαίρεση είδους
      } else throw new CustomException("You do not have the necessary permissions to perform this action");
   } catch (CustomException e) { //Διαχείρηση εξαίρεσης
      System.out.println(e.getMessage());
   }
}

//Μέθοδος που προσθέτει έναν Donator στον οργανισμό
public void insertDonator(Donator donator) {
   boolean search_donator = false;
   boolean search_beneficiary = false;
   boolean search_admin = false;
   try { //Έλεγχος για το αν υπάρχει άλλος χρήστης με το ίδιο τηλέφωνο
      for (var v : donatorList) {
         if (v.getPhone().equals(donator.getPhone())) {
```

```
            search_donator = true;
            break;
          }
        }
        for (var t : beneficiaryList) { //Έλεγχος για το αν υπάρχει Beneficiary με το ίδιο τηλέφωνο
          if (t.getPhone().equals(donator.getPhone())) {
            search_beneficiary = true;
            break;
          }
        }
        if (donator.getPhone().equals(admin.getPhone())) {
          search_admin = true; //Έλεγχος του admin
        }
        if (!search_admin && !search_donator && !search_beneficiary) {
          donatorList.add(donator); //Προσθήκη του Donator εάν δεν υπάρχει αλλιώς προκύπτει εξαίρεση
        } else if (search_donator) { //Εξαιρέσεις με κατάλληλα μηνύματα
          throw new CustomException("The user already exists in the organization as Donator");
        } else if (search_beneficiary) {
          throw new CustomException("The user already exists in the organization as Beneficiary");
        } else if (search_admin) {
          throw new CustomException("The user already exists in the organization as Admin");
        }
      } catch (CustomException e) { //Διαχείρηση εξαίρεσης
        System.out.println(e.getMessage());
      }
    }

    //Μέθοδος που αφαιρεί έναν Donator από τον οργανισμό
    public void removeDonator(Donator donator, Admin admin) {
      try {
        if (admin.isAdmin()) { //Έλεγχος για το αν ο χρήστης είναι Admin αλλιώς ποκύπτει εξαίρεση
          donatorList.remove(donator);
        } else throw new CustomException("You do not have the necessary permissions to perform this action");
      } catch (CustomException e) { //Διαχείρηση εξαίρεσης
        System.out.println(e.getMessage());
      }
    }

    //Μέθοδος που προσθέτει έναν Beneficiary στον οργανισμό
    public void insertBeneficiary(Beneficiary beneficiary) {
      boolean search_donator = false;
      boolean search_beneficiary = false;
      boolean search_admin = false;
      try { //Έλεγχος για το αν υπάρχει άλλος Beneficiary με το ίδιο τηλέφωνο
        for (var v : beneficiaryList) {
          if (v.getPhone().equals(beneficiary.getPhone())) {
            search_beneficiary = true;
            break;
          }
        }
        for (var t : donatorList) { //Έλεγχος για το αν υπάρχει κάποιος Donator με το ίδιο τηλέφωνο
          if (t.getPhone().equals(beneficiary.getPhone())) {
            search_donator = true;
```

```
            break;
          }
        }
        if (beneficiary.getPhone().equals(admin.getPhone())) {
          search_admin = true; //Έλεγχος του Admin
        }
        if (!search_admin && !search_donator && !search_beneficiary) {
          beneficiaryList.add(beneficiary); //Προσθήκη του Beneficiary εάν δεν υπάρχει αλλιώς προκύπτει εξαίρεση
        } else if (search_donator) { //Εξαιρέσεις με κατάλληλα μηνύματα
          throw new CustomException("The user already exists in the organization as Donator");
        } else if (search_beneficiary) {
          throw new CustomException("The user already exists in the organization as Beneficiary");
        } else if (search_admin) {
          throw new CustomException("The user already exists in the organization as Admin");
        }
      } catch (CustomException e) { //Διαχείρηση εξαίρεσης
        System.out.println(e.getMessage());
      }
    }

    //Μέθοδος που αφαιρεί έναν Beneficiary απο τον οργανισμό
    public void removeBeneficiary(Beneficiary beneficiary, Admin admin) {
      try {
        if (admin.isAdmin()) { //Έλεγχος για το αν ο χρήστης είναι Admin αλλιώς προκύπτει εξαίρεση
          beneficiaryList.remove(beneficiary);
        } else throw new CustomException("You do not have the necessary permissions to perform this action");
      } catch (CustomException e) { //Διαχείρηση εξαίρεσης
        System.out.println(e.getMessage());
      }
    }

    //Μέθοδος που εμφανίζει τα είδη του Οργανισμού με τις διαθέσιμες ποσότητες
    public int listEntities(String select) {
      int count = 1;

      //Εμφανίζει τα Materials
      if (select.equals("1")) {
        System.out.println("Materials");
        for (var v : entityList) {
          if(v instanceof Material){ //Εύρεση των Materials απο την λίστα
            System.out.print(count + "." + v.getName() + " (");
            System.out.println(getQuantity(v) + ")");
            count++;
          }
        }
        //Εμφανίζει τα Services
      } else if (select.equals("2")) {
        System.out.println("Services");
        for (var v : entityList) {
          if (v instanceof Service) { //Εύρεση των Services από την λίστα
            System.out.print(count + "." + v.getName() + " (");
            System.out.println(getQuantity(v) + ")");
            count++;
```

```
            }
        }
    }
    return count; //Επιστροφή της ποσότητας των ειδών που περιέχει η επιλεγμένη κατηγορία
}

//Μέθοδος εύρεσης ενός Material μέσα από ένα Entity
public Material getMaterial(Entity entity) {
    Material material = null;

    for (var v : entityList) { //Εύρεση του entity
        if (entity.getId() == v.getId()) {
            material = (Material) v; //Αποθήκευση του Entity στο Material
        }
    }

    return material; //Επιστρέφει Material
}

//Μέθοδος εύρεσης ενός Entity μέσα απο τις επιλογές του χρήστη
public Entity getEntity(String select, int choice) {
    Entity entity = null;
    int count = 1;

    //Εύρεση του Entity εάν αυτό ειναι τύπου Material
    if (select.equals("1")) {
        for (var v : entityList) {
            if (v instanceof Material) { //Εύρεση των Materials από την λίστα
                if (choice == count) {
                    entity = v; //Αποθήκευση του Entity
                    break;
                }
                count++;
            }
        }
        //Εύρεση του Entity εάν αυτό είναι τύπου Service
    } else if (select.equals("2")) {
        for (var v : entityList) {
            if (v instanceof Service) { //Εύρεση των Services από την λίστα
                if (choice == count) {
                    entity = v; //Αποθήκευση του Entity
                    break;
                }
                count++;
            }
        }
    }

    return entity; //Επιστροφή του Entity
}

//Μέθοδος εύρεσης της διαθέσιμης ποσότητας ενός Entity στον οργανισμό
public double getQuantity(Entity entity) {
```

```
    double quantity = 0;
    boolean search = false;

    if (currentDonations.rdEntities.size() == 0) {
        quantity = 0;
    } else {
        for (var v : currentDonations.rdEntities) { //Εύρεση του Entity στην λίστα του οργανισμού μέσω του iD
            if (entity.getId() == v.getEntity().getId()) {
                quantity = v.getQuantity(); //Αποθήκευση της ποσότητας
                search = true;
                break;
            }
        }
        if (!search) { //Εάν δεν βρεθέι αναθέτει τιμή 0
            quantity = 0;
        }
    }

    return quantity; //Επιστρέφει την ποσότητα
}

//Μέθοδος που εμφανίζει τους Beneficiaries του οργανισμού
public int listBeneficiaries() {
    int count = 1;
    if (beneficiaryList.size() != 0) { //Εμφάνιση των beneficiaries με κατάλληλη μορφοποίηση
        for (var v : beneficiaryList) {
            System.out.println(count + "." + v.getName());
            count++;
        }
    } else { //Εμφάνιση μηνύματος εάν δεν υπάρχουν Beneficiaries
        System.out.println("There are no beneficiaries\n");

    }

    return count; //Επιστρέφει τον αριθμό των Beneficiaries
}

//Μέθοδος που επιστρέφει έναν Beneficiary ανάλογα με την επιλογή του χρήστη
public Beneficiary getBeneficiary(int input) {
    Beneficiary beneficiary;
    beneficiary = beneficiaryList.get(input - 1); //Η λίστα ξεκινάει απο την θέση 0

    return beneficiary; //Επιστρέφει τον Beneficiary
}

//Μέθοδος που εμφανίζει τους Donators του οργανισμού
public int listDonators() {
    int count = 1;
    if (donatorList.size() != 0) { //Εμφάνιση των donators με κατάλληλη μορφοποίηση
        for (var v : donatorList) {
            System.out.println(count + "." + v.getName());
            count++;
        }
```

```
    } else { //Εμφάνιση μηνύματος εάν δεν υπάρχουν Donators
        System.out.println("There are no donators\n");
    }

    return count;   //Επιστρέφει τον αριθμό των Donators
}

//Μέθοδος που επιστρέφει έναν Donator ανάλογα με την επιλογή του χρήστη
public Donator getDonator(int input) {
    Donator donator;
    donator = donatorList.get(input - 1); //Η λίστα ξεκινάει απο την θέση 0

    return donator; //Επιστρέφει τον Donator
}

//Μέθοδος που επιστρέφει έναν Donator μέσω του τηλεφώνου του
public Donator getDonator(String phone) {
    Donator donator = null;
    for (var v : donatorList) { //Ευρεση του Donator με την χρήση του τηλεφώνου
        if (phone.equals(v.getPhone())) {
            donator = v; //Αποθήκευση του Donator
        }
    }
    return donator; //Επιστρέφει τον Donator
}

//Μέθοδος που επιστρέφει έναν Beneficiary μέσω του τηλεφώνου του
public Beneficiary getBeneficiary(String phone) {
    Beneficiary beneficiary = null;
    for (var v : beneficiaryList) { //Εύρεση του Beneficiary με την χρήση του τηλεφώνου
        if (phone.equals(v.getPhone())) {
            beneficiary = v; //Αποθήκευση του Beneficiary
        }
    }
    return beneficiary; //Επιστρέφει τον Beneficiary
}

/*Μέθοδος που καθαρίζει την receivedList κάθε
Beneficiary που βρίσκεται στον οργανισμό*/
public void resetBeneficiariesLists() {
    for (var v : beneficiaryList) {
        v.receivedList.rdEntities.clear();
    }
    System.out.println("Beneficiaries receivedList cleared."); //Μήνυμα ότι η ενέργεια ολοκληρώθηκε
    System.out.println("----------------------------");
}


}
```

Κλάση Menu

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

public class Menu {

   //Δημιουργία αντικειμένων Scanner ώστε να δέχεται τιμές από το χρήστη
   Scanner inputString = new Scanner(System.in);
   Scanner inputInt = new Scanner(System.in);

   //Μέθοδος που αναγνωρίζει τον χρήστη
   public User Identity(Organization organization) throws CustomException {
      User user = null;
      int number;
      String phone;
      boolean search = false;
      try {
         System.out.println("Login");
         System.out.print("Enter your mobile phone (3 Digits): "); //Περιμένει να βάλει ο χρήστης το τηλέφωνο του
         number = inputInt.nextInt();
         System.out.println("----------------------------");
         if(number<=0){
            throw new CustomException("The mobile number is invalid.");
         }else if(number/100<=0 || number/100>=10){
            throw new CustomException("The mobile number is invalid.");
         }
      } catch (InputMismatchException e) { //Διαχείρηση της εξαίρεσης που μπορεί να προκύψει
         throw new CustomException("----------------------------\nThe mobile number is invalid.");
      }finally {
         inputInt.nextLine();
      }

      phone = String.valueOf(number); //Μετατροπή του int σε String

      //Εύρεση του χρήστη
      if (phone.equals(organization.getAdmin().getPhone())) { //Έλεγχος εάν ο χρήστης είναι Admin
         user = organization.getAdmin(); //Αποθήκευση του χρήστη
         search = true;
      } else {
         for (var v : organization.donatorList) { //Έλεγχος εάν ο χρήστης είναι Donator
            if (phone.equals(v.getPhone())) {
               user = v; //Αποθήκευση του χρήστη
               search = true;
               break;
            }
         }
         if (!search) {
            for (var v : organization.beneficiaryList) { //Έλεγχος εάν ο χρήστης είναι Beneficiary
               if (phone.equals(v.getPhone())) {
                  user = v; //Αποθήκευση του χρήστη
                  search = true;
```

```
                break;
            }
        }
    }
}

//Εάν δεν βρεθεί ο χρήστης τον προτρέπει να εγγραφεί
if (!search) {
    boolean error;
    do {
        error = false;
        try {
            Register(organization, phone); //Καλεί την μέθοδο για την εγγραφή του χρήστη
        } catch (CustomException e) {
            error = true;
            System.out.println(e.getMessage());
            System.out.println("----------------------------");
        }
    } while (error);
}

return user;
}

//Μέθοδος που εγγράφει έναν νέο χρήστη
public void Register(Organization organization, String phone) throws CustomException {
    String choice, name, input;
    boolean error = false, back;
    int noPersons;

    do {
        back = false;
        System.out.print("Do you want to register (Y/N): "); //Μήνυμα για το αν θέλει να εγγραφεί
        choice = inputString.nextLine();
        if (choice.equals("Y") || choice.equals("y")) {
            System.out.println("----------------------------");
            System.out.println("Registration"); //Έναρξη εγγραφής
            System.out.println("Do you want to register as:\n1.Donator\n2.Beneficiary\n3.Back");
            System.out.print("choose: ");
            input = inputString.nextLine();
            System.out.println("----------------------------");
            if (input.equals("1")) { //Εγγραφή ως Donator
                do {
                    try {
                        System.out.print("Name:");
                        name = inputString.nextLine();
                        if (name.matches(".*\\d.*")) {
                            throw new CustomException("Invalid name\nPlease enter a valid name");
                        }
                        organization.insertDonator(new Donator(name, phone)); //Εκχώρηση του Donator στον οργανισμό
                        System.out.println("You have register"); //Μήνυμα για την επιτυχίας της εγγραφής
                        System.out.println("----------------------------");
                    } catch (CustomException e) {
```

```
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
                error = true;
              }
            } while (error);
          } else if (input.equals("2")) { //Εγγραφή ως Beneficiary
            do {
              try {
                System.out.print("Name:");
                name = inputString.nextLine();
                if (name.matches(".*\\d.*")) {
                  throw new CustomException("Invalid name\nPlease enter a valid name");
                }
                try {                                   //Διαχείριση των εξαιρέσεων από την λάθος είσοδο του χρήστη
                  System.out.print("Number of persons in the family:");
                  noPersons = inputInt.nextInt();
                } catch (InputMismatchException e) {
                  throw new CustomException("The mobile number is invalid.");
                }
                organization.insertBeneficiary(new Beneficiary(name, phone, noPersons)); //Εκχώρηση του Donator στον οργανισμό
                System.out.println("You have register"); //Μήνυμα για την επιτυχίας της εγγραφής
                System.out.println("----------------------------");
              } catch (CustomException e) {
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
                error = true;
              }
            } while (error);
          } else if (input.equals("3")) {
            back = true;
          } else {
            throw new CustomException("[" + input + "]" + "is invalid\nYou must enter a valid number.");
          }
        } else if (choice.equals("N") || choice.equals("n")) {
          System.out.println("----------------------------");
        } else {
          throw new CustomException("----------------------------\nInvalid choice\nPlease select Y/N");
        }
      } while (back);

    }

    //Μέθοδος που εμφανίζει τα στοιχεία του χρήστη και τις πληροφορίες του
    public void Welcome(User user, Organization organization) {
      System.out.print("Welcome ");
      System.out.println(user.toString());
      System.out.println("Organization: " + organization.getName());
      System.out.println("----------------------------");
    }

    //Μέθοδος που εμφανίζει κατάλληλα το κεντρικό menu του Admin
    public String AdminMainMenu() throws CustomException {
```

```
    String input;
    String input_2;
    boolean back,error;

    do {
        back=false;
        System.out.println("1.View\n2.Monitor Organization\n3.Logout\n4.Exit");
        System.out.print("Choose: ");
        input = inputString.nextLine();
        System.out.println("----------------------------");
        if (!input.equals("1") && !input.equals("2") && !input.equals("3") && !input.equals("4")) {
            throw new CustomException("[" + input + "]" + " is invalid\nYou must enter a valid number.");
        }
        if (input.equals("4")) {
            do {
                error=false;
                try {
                    System.out.print("Do you want to exit (Y/N):");
                    input_2 = inputString.nextLine();
                    System.out.println("----------------------------");
                    if (input_2.equals("N") || input_2.equals("n")) {
                        back = true;
                    } else if (!input_2.equals("Y") && !input_2.equals("y")) {
                        throw new CustomException("Invalid choice\nPlease select Y/N");
                    }
                } catch (CustomException e) {
                    error=true;
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                }
            }while (error);
        }
    }while (back);
    return input;
}

//Μέθοδος που εμφανίζει τις δύο κατηγορίες των Entities
public String ViewCategories() throws CustomException {
    String input;

    System.out.println("1.Material\n2.Services\n3.Back");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------------------------");
    if (!input.equals("1") && !input.equals("2") && !input.equals("3")) {
        throw new CustomException("[" + input + "] is invalid\nYou must enter a valid number.");
    }
    return input;
}

//Μέθοδος που εμφανίζει τα Entities για έναν Admin
public boolean ViewEntities(Organization organization, String input) throws CustomException {
    int count;
```

```
    int choice;
    boolean back = false;
    Entity entity;
    try {
        count = organization.listEntities(input); //Εμφανίζει τα Material ή τα Services
        System.out.println(count + ".Back");
        System.out.print("Choice: ");
        choice = inputInt.nextInt();
        if (choice == count) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
            System.out.println("----------------------------");
            back = true;
        }
        if (choice < 1 || choice > count) { //Δημιουργία εξαίρεσης
            throw new CustomException("The input is invalid\nYou must enter a valid number");
        }
        if (choice != count) { //Εμφάνιση πληροφοριών για κάποιο Entity
            System.out.println("----------------------------");
            entity = organization.getEntity(input, choice); //Εύρεση Entity
            System.out.println(entity.toString());
            System.out.println("Quantity: " + organization.getQuantity(entity));
            System.out.println("----------------------------");
        }
    } catch (InputMismatchException e) {
        throw new CustomException("The input is invalid\nYou must enter a valid number");
    } finally {
        inputInt.nextLine();
    }


    return back;
}

//Μέθοδος που εμφανίζει τα Entities για έναν Donator
public boolean ViewEntitiesDonator(Organization organization, String input, User user) throws CustomException {
    int count, choice, quantity;
    String select = "-1";
    boolean back = false, error;
    Entity entity;
    Donator donator;

    try {
        count = organization.listEntities(input); //Εμφανίζει τα Materials ή τα Services
        System.out.println(count + ".Back");
        System.out.print("Choice: ");
        choice = inputInt.nextInt();
        if (choice == count) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
            System.out.println("----------------------------");
            back = true;
        }
        if (choice < 1 || choice > count) {
            throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
        }
```

```
        if (choice != count) { //Εμφάνιση πληροφοριών για κάποιο Entity
            System.out.println("----------------------------");
            entity = organization.getEntity(input, choice);
            System.out.println(entity.toString());
            System.out.println("Quantity: " + organization.getQuantity(entity));
            System.out.println("----------------------------");
            do {
                error = false;
                try {
                    System.out.println("Do you want to offer " + entity.getName() + "?"); //Προσθήκη δωρεάς από τον
Donator
                    System.out.print("Choose (Y/N): ");
                    select = inputString.nextLine();
                    System.out.println("----------------------------");
                    if (!select.equals("Y") && !select.equals("y") && !select.equals("N") && !select.equals("n")) {
                        throw new CustomException("[" + select + "] is invalid\nYou must enter Y/N");
                    }
                } catch (CustomException e) {
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                    error = true;
                }
            } while (error);
            if (select.equals("Y") || select.equals("y")) {
                do {
                    error = false;
                    try {
                        System.out.print("Quantity: "); //Προσθήκη της ποσότητας για κάποια δωρεά
                        quantity = inputInt.nextInt();
                        System.out.println("----------------------------");
                        if (quantity <= 0) {
                            throw new CustomException("The input is invalid\nYou must enter a valid number");
                        }
                        donator = organization.getDonator(user.getPhone()); //Εύρεση του Donator
                        donator.offersList.add(new RequestDonation(entity, quantity)); //Προσθήκη δωρεάς στην λίστα του
Donator
                    } catch (InputMismatchException e) {
                        System.out.println("----------------------------");
                        System.out.println("The input is invalid\nYou must enter a valid number");
                        System.out.println("----------------------------");
                        inputInt.nextLine();
                        error = true;
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        inputInt.nextLine();
                        error = true;
                    }
                } while (error);
            }
        }
    } catch (InputMismatchException e) {
        throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
```

```
        } finally {
            inputInt.nextLine();
        }


        return back;
    }

    //Μέθοδος που εμφανίζει τα Entities για έναν Beneficiary
    public boolean ViewEntitiesBeneficiary(Organization organization, String input, User user) throws CustomException
{
        int count;
        int choice;
        String select = "-1";
        boolean back = false, error;
        int quantity;
        Entity entity;
        Beneficiary beneficiary;
        try {
            count = organization.listEntities(input); //Εμφάνιση των Materials ή Entities
            System.out.println(count + ".Back");
            System.out.print("Choice: ");
            choice = inputInt.nextInt();
            if (choice == count) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
                System.out.println("----------------------------");
                back = true;
            }
            if (choice < 1 || choice > count) {
                throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
            }
            if (choice != count) { //Εμφάνιση πληροφοριών για κάποιο Entity
                System.out.println("----------------------------");
                entity = organization.getEntity(input, choice);
                System.out.println(entity.toString());
                System.out.println("Quantity: " + organization.getQuantity(entity));
                System.out.println("----------------------------");
                do {
                    error = false;
                    try {
                        System.out.println("Do you want to receive " + entity.getName() + "?"); //Προσθήκη είδους που ζητάει
o Beneficiary
                        System.out.print("Choose (Y/N): ");
                        select = inputString.nextLine();
                        System.out.println("----------------------------");
                        if (!select.equals("Y") && !select.equals("y") && !select.equals("N") && !select.equals("n")) {
                            throw new CustomException("[" + select + "] is invalid\nYou must enter Y/N");
                        }
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        error = true;
                    }
                } while (error);
```

```
            if (select.equals("Y") || select.equals("y")) {
                do {
                    error = false;
                    try {
                        System.out.print("Quantity: "); //Προσθήκη της ποσότητας τοθ Request
                        quantity = inputInt.nextInt();
                        System.out.println("----------------------------");
                        if (quantity <= 0) {
                            throw new CustomException("The input is invalid\nYou must enter a valid number");
                        }
                        beneficiary = organization.getBeneficiary(user.getPhone()); //Εύρεση του Beneficiary
                        beneficiary.requestsList.add(new RequestDonation(entity, quantity), organization, beneficiary);
//Προσθήκη του Request στην requestList
                    } catch (InputMismatchException e) {
                        System.out.println("----------------------------");
                        System.out.println("The input is invalid\nYou must enter a valid number");
                        System.out.println("----------------------------");
                        inputInt.nextLine();
                        error = true;
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        inputInt.nextLine();
                        error = true;
                    }
                } while (error);
            }
        }
    } catch (InputMismatchException e) {
        throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
    } finally {
        inputInt.nextLine();
    }


    return back;
}

//Μέθοδος που εμφανίζει κατάλληλα τις επιλογές του MonitorOrganization
public String AdminMonitorOrganization() throws CustomException {
    String input;

    System.out.println("1.List Beneficiaries\n2.List Donators\n3.Reset Beneficiaries");
    System.out.println("4.Back");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------------------------");
    if (!input.equals("1") && !input.equals("2") && !input.equals("3") && !input.equals("4")) {
        throw new CustomException("[" + input + "]" + "is invalid\nYou must enter a valid number.");
    }

    return input;
}
```

```
  //Μέθοδος που καθαρίζει την receivedList όλων των Beneficiary μετά από ερώτηση του χρήστη
  public void AdminResetBeneficiaries(Organization organization) throws CustomException {
    String input;

    System.out.println("Confirmation, do you want to proceed (Y/N): ");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------------------------");
    if (input.equals("Y") || input.equals("y")) {
      organization.resetBeneficiariesLists();
    } else if (!input.equals("N") && !input.equals("n")) {
      throw new CustomException("[" + input + "]" + "is invalid\nYou must enter a valid number.");
    }
  }

  //Μέθοδος που εμφανίζει τους Beneficiaries του οργανισμού για έναν Admin
  public boolean AdminListBeneficiaries(Organization organization) throws CustomException {
    boolean back, back_1;
    int count;
    int input;
    Beneficiary beneficiary;
    boolean error;
    do {
      back = false;
      back_1 = false;
      try {
        count = organization.listBeneficiaries(); //Εμφάνιση των Beneficiaries
        System.out.println(count + ".Back");
        System.out.print("Choose: ");
        input = inputInt.nextInt();
        System.out.println("----------------------------");
        if (input == count) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
          back = true;
        }
        if (input < 1 || input > count) {
          throw new CustomException("The input is invalid\nYou must enter a valid number");
        }
        if (input != count) {
          beneficiary = organization.getBeneficiary(input); //Εύρεση του beneficiary
          do {
            error = false;
            try {
              back_1 = AdminListBeneficiariesActions(organization, beneficiary); //Μέθοδος με ενέργειες για έναν
Beneficiary
            } catch (CustomException e) {
              System.out.println(e.getMessage());
              System.out.println("----------------------------");
              error = true;
            }
          } while (error || !back_1);
        }
      } catch (InputMismatchException e) {
```

```
            throw new CustomException("----------\nThe input is invalid\nYou must enter a valid number");
        } finally {
            inputInt.nextLine();
        }
    } while (back_1);

    return back;
}

//Μέθοδος που εμφανίζει τις ενέργειες που μπορούν να γίνουν σε έναν Beneficiary από έναν Admin
public boolean AdminListBeneficiariesActions(Organization organization, Beneficiary beneficiary) throws CustomException {
    String input;
    boolean back = false;
    System.out.println(beneficiary.toString()); //Εμφάνιση του Beneficiary
    if (beneficiary.receivedList.rdEntities.size() != 0) {
        System.out.println("\nHe has received:"); //Εμφάνιση ειδών που έχει παραλάβει
        for (var v : beneficiary.receivedList.rdEntities) {
            System.out.println("-> " + v.getEntity().getName() + " [" + v.getQuantity() + "]");
        }
    } else {
        System.out.println("He has not taken anything");
    }

    //Ενέργειες που μπορούν να γίνουν στον Beneficiary
    System.out.println("\n1.Clear receivedList\n2.Remove Beneficiary");
    System.out.println("3.Back");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------");
    if (input.equals("3")) {
        back = true; //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
    } else if (input.equals("1")) {
        beneficiary.receivedList.rdEntities.clear(); //Καθαρίζει την receivedList του Beneficiary
        System.out.println("ReceivedList cleared");
        System.out.println("----------");
    } else if (input.equals("2")) {
        organization.removeBeneficiary(beneficiary, organization.getAdmin()); //Διαγράφει τον Beneficiary από
        System.out.println("Beneficiary removed");                          //τον οργανισμό
        System.out.println("----------");
        back = true;
    }
    if (!input.equals("1") && !input.equals("2") && !input.equals("3")) {
        throw new CustomException("[" + input + "] is not valid\nYou must enter a valid number");
    }

    return back;
}

//Μέθοδος που εμφανίζει τους Donators του οργανισμού για έναν Admin
public boolean AdminListDonators(Organization organization) throws CustomException {
    boolean back, back_1;
    int count, input;
```

```
    Donator donator;
    boolean error;
    do {
      back_1 = false;
      back = false;
      try {
        count = organization.listDonators(); //Εμφάνιση των Donators
        System.out.println(count + ".Back");
        System.out.print("Choose: ");
        input = inputInt.nextInt();
        System.out.println("----------------------------");
        if (count == input) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
          back = true;
        }
        if (input < 1 || input > count) {
          throw new CustomException("The input is invalid\nYou must enter a valid number");
        }
        if (count != input) {
          donator = organization.getDonator(input); //Εύρεση του Donator
          do {
            error = false;
            try {
              back_1 = AdminListDonatorsActions(organization, donator); //Μέθοδος με ενέργειες για έναν
Donator
            } catch (CustomException e) {
              System.out.println(e.getMessage());
              System.out.println("----------------------------");
              error = true;
            }
          } while (error || !back_1);
        }
      } catch (InputMismatchException e) {
        throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
      } finally {
        inputInt.nextLine();
      }
    } while (back_1);

    return back;
  }

  //Μέθοδος που εμφανίζει τις ενέργειες που μπορούν να γίνουν σε έναν Donator από έναν Admin
  public boolean AdminListDonatorsActions(Organization organization, Donator donator) throws CustomException {
    String input;
    boolean back = false;
    System.out.println(donator.toString()); //Εμφάνιση του Donator
    if (donator.offersList.rdEntities.size() != 0) { //Εμφάνιση ειδών που θέλει να δώσει
      System.out.println("\nHe want to offer:");
      for (var v : donator.offersList.rdEntities) {
        System.out.println("-> " + v.getEntity().getName() + " [" + v.getQuantity() + "]");
      }
```

```
    } else {
        System.out.println("He has not offer anything");

    }
    System.out.println("\n1.Delete Donator");
    System.out.println("2.Back");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------------------------");
    if (input.equals("2")) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
        back = true;
    } else if (input.equals("1")) {
        organization.removeDonator(donator, organization.getAdmin()); //Διαγράφει τον Donator από τον οργανισμό
        System.out.println("Donator deleted");
        System.out.println("----------------------------");
        back = true;
    }
    if (!input.equals("1") && !input.equals("2")) {
        throw new CustomException("[" + input + "] is not valid\nYou must enter a valid number");
    }


    return back;
}

//Μέθοδος που εμφανίζει κατάλληλα το κεντρικό menu ενός Donator
public String DonatorMainMenu(Organization organization, User user) throws CustomException {
    String input,input_2;
    boolean back, error;
    Donator donator;

    do {
        back = false;
        System.out.println("1.Add Offer\n2.Show Offers\n3.Commit\n4.Logout\n5.Exit");
        System.out.print("Choose: ");
        input = inputString.nextLine();
        System.out.println("----------------------------");
        if (input.equals("1")) {
            back = DonatorAddOffer(organization, user); //Καλειται μέθοδος για το επόμενο επίπεδο του menu
        } else if (input.equals("2")) {
            do {
                error = false;
                try {
                    back = DonatorShowOffer(organization, user); //Καλειται μέθοδος για το επόμενο επίπεδο του menu
                } catch (CustomException e) {
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                    error = true;
                }
            } while (error);
        } else if (input.equals("3")) {
            donator = organization.getDonator(user.getPhone()); //Εύρεση του Donator
            donator.offersList.commit(organization); //Καλείτε μέθοδος για την εκτέλεση της δωρεάς
```

```
            back = true;
        }else if (input.equals("5")) {
            do {
                error=false;
                try {
                    System.out.print("Do you want to exit (Y/N):");
                    input_2 = inputString.nextLine();
                    System.out.println("----------------------------");
                    if (input_2.equals("N") || input_2.equals("n")) {
                        back=true;
                    } else if (!input_2.equals("Y") && !input_2.equals("y")) {
                        throw new CustomException("Invalid choice\nPlease select Y/N");
                    }
                } catch (CustomException e) {
                    error=true;
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                }
            }while (error);
        }
        if (!input.equals("1") && !input.equals("2") && !input.equals("3") && !input.equals("4") && !input.equals("5"))
{
            throw new CustomException("[" + input + "]" + " is invalid\nYou must enter a valid number.");
        }


    } while (back);

    return input;
  }

  //Μέθοδος που εμφανίζει το menu για την προσθήκη μιας δωρεάς
  public boolean DonatorAddOffer(Organization organization, User user) {
    boolean back, error, back_1;
    String input;

    do {
      error = false;
      back = false;
      back_1 = false;
      try {
        input = ViewCategories(); //Εμφάνιση κατηγοριών του Entity
        if (!input.equals("3")) {
          do {
            error = false;
            try {
              back_1 = ViewEntitiesDonator(organization, input, user); //Εμφάνιση ειδών για δωρεά
            } catch (CustomException e) {                          //και εισαγωγή της ποσότητας
              System.out.println(e.getMessage());
              System.out.println("----------------------------");
              error = true;
            }
          } while (!back_1 || error);
```

```
            }
            if (input.equals("3")) {
                back = true; //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
            }
        } catch (CustomException e) {
            System.out.println(e.getMessage());
            System.out.println("----------------------------");
            error = true;
        }
    } while (error || back_1);

    return back;
}

//Μέθοδος που εμφανίζει τις δωρεές ειδών που θέλει να κάνει ο Donator
public boolean DonatorShowOffer(Organization organization, User user) throws CustomException {
    boolean back = false, back_1;
    int count, input;
    Donator donator;

    do {
        back_1 = false;
        try {
            donator = organization.getDonator(user.getPhone()); //Εύρεση Donator
            count = donator.offersList.monitor(donator); //Εμφάνιση των ειδών που διατιθετε να δώσει

            System.out.println("\n" + count + ".Clear all offers");
            System.out.println(count + 1 + ".Commit");
            System.out.println(count + 2 + ".Back");
            System.out.print("Choose: ");
            input = inputInt.nextInt();
            System.out.println("----------------------------");
        } catch (InputMismatchException e) {
            throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
        } finally {
            inputInt.nextLine();
        }
        if (input == count + 2) {
            back = true; //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
        } else if (input < 1 || input > count + 2) {
            throw new CustomException("The input is invalid\nYou must enter a valid number");
        } else if (input < count) {
            back_1 = DonatorShowOfferActions(donator, input); //Εμφάνιση ενεργειών για κάποιο είδος προς δωρεά
        } else if (input == count) {
            donator.offersList.reset(); //Καθαρίζει την offerList του Donator
            System.out.println("Offers cleared");
            System.out.println("----------------------------");
            back = true;
        } else if (input == count + 1) {
            donator.offersList.commit(organization); //Καλείται μέθοδος για την εκτέλεση της δωρεάς
            back = true;
        }
    } while (back_1);
```

```
        return back;
    }

    //Μέθοδος που εμφανίζει τις ενέργειες που μπορούν να γίνουν σε ένα είδος προς δωρεά από τον Donator
    public boolean DonatorShowOfferActions(Donator donator, int select) {
        String input;
        boolean back = false, error, back_1;
        int quantity;
        RequestDonation requestDonation = donator.offersList.get(select); //Εύρεση του requestDonation
        do {
            try {
                do {
                    error = false;
                    back_1 = false;
                    back = false;
                    System.out.println("Offer: " + requestDonation.getEntity().getName() + " [" +
requestDonation.getQuantity() + "]");
                    System.out.println("1.Delete offer\n2.Modify offer\n3.Back"); //Εμφάνιση της δωρεάς και των ενεργειών
                    System.out.print("Choose: ");
                    input = inputString.nextLine();
                    System.out.println("----------------------------");
                    if (input.equals("1")) {
                        donator.offersList.remove(requestDonation); //Αφαιρεί την RequestDonation από την λίστα του
Donator
                        back = true;
                        back_1 = false;
                    } else if (input.equals("2")) {
                        do {
                            error = false;
                            back_1 = false;
                            try {
                                System.out.print("Quantity: ");
                                quantity = inputInt.nextInt();
                                System.out.println("----------------------------");
                                donator.offersList.modify(select, quantity); //Καλέιται μέθοδος ώστε να γίνει αλλαγή της
ποσότητας
                                back_1 = true;                          //ενός Requestdonation
                                if (quantity <= 0) {
                                    throw new CustomException("The input is invalid\nYou must enter a valid number");
                                }
                            } catch (InputMismatchException e) {
                                System.out.println("----------------------------");
                                System.out.println("The input is invalid\nYou must enter a valid number");
                                System.out.println("----------------------------");
                                error = true;
                            } catch (CustomException e) {
                                System.out.println(e.getMessage());
                                System.out.println("----------------------------");
                                error = true;
                            } finally {
                                inputInt.nextLine();
                            }
```

```
                } while (error);
            } else if (input.equals("3")) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
                back = true;
                back_1 = false;
            }
            if (!input.equals("1") && !input.equals("2") && !input.equals("3")) {
                throw new CustomException("[" + input + "]" + " is invalid\nYou must enter a valid number.");
            }
        } while (back_1);
    } catch (CustomException e) {
        System.out.println(e.getMessage());
        System.out.println("----------------------------");
        error = true;
    }
  } while (error);

  return back;
}

//Μέθοδος που εμφανίζει κατάλληλα το κεντρικό menu ενός Beneficiary
public String BeneficiaryMainMenu(Organization organization, User user) throws CustomException {
  String input,input_2;
  boolean back,error;
  Beneficiary beneficiary;

  do {
    back = false;
    System.out.println("1.Add request\n2.Show requests\n3.Commit\n4.Logout\n5.Exit");
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------------------------");
    if (input.equals("1")) {
        back = BeneficiaryAddRequest(organization, user); //Καλειται μέθοδος για το επόμενο επίπεδο του menu
    } else if (input.equals("3")) {
        beneficiary = organization.getBeneficiary(user.getPhone()); //Εύρεση Beneficiary
        beneficiary.requestsList.commit(beneficiary, organization); //Καλείτε μέθοδος για την εκτέλεση του αιτήματος
        back = true;
    } else if (input.equals("2")) {
        back = BeneficiaryShowRequest(user, organization); //Καλειται μέθοδος για το επόμενο επίπεδο του menu
    }else if (input.equals("5")) {
        do {
            error=false;
            try {
                System.out.print("Do you want to exit (Y/N):");
                input_2 = inputString.nextLine();
                System.out.println("----------------------------");
                if (input_2.equals("N") || input_2.equals("n")) {
                    back=true;
                } else if (!input_2.equals("Y") && !input_2.equals("y")) {
                    throw new CustomException("Invalid choice\nPlease select Y/N");
                }
            } catch (CustomException e) {
```

```
                error=true;
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
              }
            }while (error);
        }
        if (!input.equals("1") && !input.equals("2") && !input.equals("3") && !input.equals("4") && !input.equals("5"))
{
            throw new CustomException("[" + input + "] is invalid\nPlease enter a valid number");
        }
      } while (back);

      return input;
    }

    //Μέθοδος που εμφανίζει το menu για την προσθήκη ενός Request
    public boolean BeneficiaryAddRequest(Organization organization, User user) {
      boolean back, error, back_1;
      String input;

      do {
        error = false;
        back = false;
        back_1 = false;
        try {
          input = ViewCategories(); //Εμφάνιση κατηγοριών του Entity
          if (!input.equals("3")) {
            do {
              error = false;
              try {
                back_1 = ViewEntitiesBeneficiary(organization, input, user); //Εμφάνιση των ειδών για αίτηση
              } catch (CustomException e) {                          //και εισαγωγή της ποσότητας
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
                error = true;
              }
            } while (!back_1 || error);
          }
          if (input.equals("3")) {
            back = true; //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
          }
        } catch (CustomException e) {
          System.out.println(e.getMessage());
          System.out.println("----------------------------");
          error = true;
        }
      } while (error || back_1);

      return back;
    }

    //Μέθοδος που εμφανίζει των αιτημάτων που θέλει να κάνει ο Beneficiary
    public boolean BeneficiaryShowRequest(User user, Organization organization) throws CustomException {
```

```
    boolean back = false, back_1;
    int count, input;
    Beneficiary beneficiary;
    do {
      back_1 = false;
      try {
        try {
          beneficiary = organization.getBeneficiary(user.getPhone()); //Εύρεση του Beneficiary
          count = beneficiary.requestsList.monitor(beneficiary); ///Εμφάνιση των αιτημάτων που θέλει να κάνει
          System.out.println("\n" + count + ".Clear all requests");
          System.out.println(count + 1 + ".Commit");
          System.out.println(count + 2 + ".Back");
          System.out.print("Choose: ");
          input = inputInt.nextInt();
          System.out.println("----------------------------");
        } catch (InputMismatchException e) {
          throw new CustomException("----------------------------\nThe input is invalid\nYou must enter a valid number");
        }
        if (input == count + 2) {
          back = true; //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
        } else if (input < 1 || input > count + 2) {
          throw new CustomException("The input is invalid\nYou must enter a valid number");
        } else if (input < count) {
          back_1 = BeneficiaryShowRequestActions(beneficiary, input, organization); //Εμφάνιση ενεργειών για κάποιο αίτημα
        } else if (input == count) {
          beneficiary.requestsList.reset(); //Καθαριζει την requestsList του Beneficiary
          System.out.println("Requests cleared");
          System.out.println("----------------------------");
          back = true;
        } else if (input == count + 1) {
          beneficiary.requestsList.commit(beneficiary, organization); //Καλέιται η μέθοδος για την εκτέλεση του αιτήματος
          back_1 = false;
          back = true;
        }
      } catch (CustomException e) {
        back_1 = true;
        System.out.println(e.getMessage());
        System.out.println("----------------------------");
      } finally {
        inputInt.nextLine();
      }
    } while (back_1);
    return back;
  }

  //Μέθοδος που εμφανίζει τις ενέργειες που μπορούν να γίνουν σε ένα αίτημα του Beneficiary
  public boolean BeneficiaryShowRequestActions(Beneficiary beneficiary, int select, Organization organization) {
    String input;
    boolean back, error, back_1;
    int quantity;
```

```
RequestDonation requestDonation = beneficiary.requestsList.get(select); //Εύρεση του requestDonation

do {
  back_1 = false;
  back = false;
  error = false;
  try {
    System.out.println("Request: " + requestDonation.getEntity().getName() + " [" + requestDonation.getQuantity() + "]");
    System.out.println("1.Delete request\n2.Modify request\n3.Back"); //Εμφάνιση του αιτήματος και των ενεργειών
    System.out.print("Choose: ");
    input = inputString.nextLine();
    System.out.println("----------");
    if (input.equals("1")) {
      beneficiary.requestsList.remove(requestDonation); //Αφαιρεί του αιτήματος από την requestList του Beneficiary
      System.out.println("Request deleted");
      System.out.println("----------");
      back = true;
    } else if (input.equals("2")) {
      do {
        error = false;
        try {
          requestDonation = beneficiary.requestsList.get(select);
          System.out.print("Quantity: ");
          quantity = inputInt.nextInt();
          System.out.println("----------");
          if (quantity <= 0) {
            throw new CustomException("The input is invalid\nYou must enter a valid number");
          }
          try {
            beneficiary.requestsList.modify(requestDonation, organization, beneficiary, quantity); //Καλέιτε μέθοδος ώστε να γίνει
            back_1 = true;                                                    //αλλαγή της ποσότητας ενός αιτήματος
          } catch (CustomException msg) {
            back_1=true;
            System.out.println(msg.getMessage());
            System.out.println("----------");
          }
        } catch (InputMismatchException e) {
          System.out.println("----------");
          System.out.println("The input is invalid\nYou must enter a valid number");
          System.out.println("----------");
          inputInt.nextLine();
          error = true;
        } catch (CustomException e) {
          System.out.println(e.getMessage());
          System.out.println("----------");
          inputInt.nextLine();
          error = true;
        }
      } while (error);
```

```
            } else if (input.equals("3")) { //Πηγαίνει στο menu ένα επίπεδο προς τα πάνω
                back = true;
                back_1 = false;
            }
            if (!input.equals("1") && !input.equals("2") && !input.equals("3")) {
                throw new CustomException("[" + input + "]" + " is invalid\nYou must enter a valid number.");
            }
        } catch (CustomException e) {
            error = true;
            System.out.println(e.getMessage());
            System.out.println("----------------------------");
        }
    } while (back_1 || error);

    return back;
  }

}
```

Κλάση Main

```
public class Main {

  public static void main(String[] args) {

    //Δημιουργία αντικειμένου Admin
    Admin admin = new Admin("Dimitris", "698");

    //Δημιουργία αντικειμένου Organization
    Organization organization = new Organization("CEID", admin);

    //Δημιουργία αντικειμένων Materials
    Material milk = new Material("Milk", "Cow's", 100, 2, 4, 6);
    Material sugar = new Material("Sugar", "Brown sugar", 101, 2, 4, 6);
    Material rice = new Material("Rice", "White", 102, 2, 4, 6);

    //Προσθήκη ειδών στον οργανισμό
    organization.addEntity(milk);
    organization.addEntity(sugar);
    organization.addEntity(rice);

    //Δημιουργία αντικειμένων Services
    Service MedicalSupport = new Service("MedicalSupport", "Medicines", 200);
    Service NurserySupport = new Service("NurserySupport", "School", 201);
    Service BabySitting = new Service("BabySitting", "For children", 202);

    //Προσθήκη ειδών στον οργανισμό
    organization.addEntity(MedicalSupport);
    organization.addEntity(NurserySupport);
    organization.addEntity(BabySitting);

    //Προσθήκη διαθέσιμων ειδών στον οργανισμό
```

```
organization.currentDonations.add(new RequestDonation(milk, 6));
organization.currentDonations.add(new RequestDonation(rice, 7));
organization.currentDonations.add(new RequestDonation(sugar,2));
organization.currentDonations.add(new RequestDonation(MedicalSupport, 18));
organization.currentDonations.add(new RequestDonation(BabySitting, 2));

//Δημιουργία αντικειμένου Beneficiary
Beneficiary beneficiary_1 = new Beneficiary("Konstantinos", "697", 2);

//Προσθήκη του Beneficiary στον οργανισμό
organization.insertBeneficiary(beneficiary_1);

//Προσθήκη ειδών που ζητά ο Beneficiary
beneficiary_1.requestsList.add(new RequestDonation(milk, 3));
beneficiary_1.requestsList.add(new RequestDonation(sugar, 2));
beneficiary_1.requestsList.add(new RequestDonation(NurserySupport,7));

//Δημιουργία αντικειμένου Beneficiary
Beneficiary beneficiary_2 = new Beneficiary("Giorgos", "693", 6);

//Προσθήκη του Beneficiary στον οργανισμό
organization.insertBeneficiary(beneficiary_2);

beneficiary_2.receivedList.add(new RequestDonation(rice,4));
beneficiary_2.receivedList.add(new RequestDonation(MedicalSupport,3));
//Δημιουργία αντικειμένου Donator
Donator donator_1 = new Donator("Alexandros", "695");

//Προσθήκη του Donator στον Οργανισμό
organization.insertDonator(donator_1);

//Προσθήκη ειδών που θέλει να δωρίσει ο Donator
donator_1.offersList.add(new RequestDonation(sugar, 4));
donator_1.offersList.add(new RequestDonation(BabySitting, 1));

//Δημιουργία αντικειμένου Menu
Menu menu = new Menu();

User user = null;
String input, input_2, input_4, input_6, input_7, input_8;
boolean input_3, input_5;
boolean error;

do {
   input = "-1";
   input_6 = "-1";
   input_7 = "-1";
   do {
      error = false;
      try {
         user = menu.Identity(organization); //Αναγνώρηση χρήστη
      } catch (CustomException e) {
         System.out.println(e.getMessage());
```

```
            System.out.println("----------------------------");
            error = true;
        }
    } while (error || user == null);
    if (user instanceof Admin) { //Είσοδος ως Admin
        menu.Welcome(user, organization); //Χαιρετισμός του Admin
        do {
            do {
                input_2 = "-1";
                input_4 = "-1";
                error = false;
                try {
                    input = menu.AdminMainMenu(); //Κεντρικό menu του Admin
                } catch (CustomException e) {
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                    error = true;
                }
            } while (error);
            if (input.equals("1")) {
                do {
                    input_3 = false;
                    do {
                        error = false;
                        try {
                            input_2 = menu.ViewCategories(); //Εμφάνισει δύο κατηγοριών (Materials, Entities)
                        } catch (CustomException e) {
                            System.out.println(e.getMessage());
                            System.out.println("----------------------------");
                            error = true;
                        }
                    } while (error);
                    if (input_2.equals("1") || input_2.equals("2")) {
                        do {
                            error = false;
                            try {
                                input_3 = menu.ViewEntities(organization, input_2); //Εμφανισει ειδών μιας κατηργορίας
                            } catch (CustomException e) {
                                System.out.println("----------------------------");
                                System.out.println(e.getMessage());
                                System.out.println("----------------------------");
                                error = true;
                            }
                        } while (error || !input_3);
                    }
                } while (input_3);
            } else if (input.equals("2")) {
                do {
                    input_5 = false;
                    do {
                        error = false;
                        try {
                            input_4 = menu.AdminMonitorOrganization(); //Επιλογές για την διαχείρηση του οργανισμού
```

```
                } catch (CustomException e) {
                    System.out.println(e.getMessage());
                    System.out.println("----------------------------");
                    error = true;
                }
            } while (error);
            if (input_4.equals("1")) {
                do {
                    error = false;
                    try {
                        input_5 = menu.AdminListBeneficiaries(organization); //Εμφανίζει του Beneficiaries
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        error = true;
                    }
                } while (error);
            } else if (input_4.equals("2")) {
                do {
                    error = false;
                    try {
                        input_5 = menu.AdminListDonators(organization); //Εμφανίζει τους Donators
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        error = true;
                    }
                } while (error);
            } else if (input_4.equals("3")) {
                do {
                    error=false;
                    input_5=false;
                    try {
                        menu.AdminResetBeneficiaries(organization); //Καθαρίζει την receivedList κάθε Beneficiary
                    } catch (CustomException e) {
                        System.out.println(e.getMessage());
                        System.out.println("----------------------------");
                        error = true;
                    }
                    input_5=true;
                } while (error);
            }
        } while (input_5);
    }
  } while (input_2.equals("3") || input_4.equals("4"));
} else if (user instanceof Donator) { //Είσοδος ως Donator
  menu.Welcome(user, organization); //Χαιρετισμός του Donator
  do {
    error = false;
    input_6 = "-1";
    try {
      input_6 = menu.DonatorMainMenu(organization, user); //Κενρικό menu του Donator
    } catch (CustomException e) {
```

```
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
                error = true;
              }
            } while (error);
          } else if (user instanceof Beneficiary) { //Είσοδος ως Beneficiary
            menu.Welcome(user, organization); //Χαιρετισμός του Beneficiary
            do {
              error = false;
              input_7 = "-1";
              try {
                input_7 = menu.BeneficiaryMainMenu(organization, user); //Κεντρικό menu του Donator
              } catch (CustomException e) {
                System.out.println(e.getMessage());
                System.out.println("----------------------------");
                error = true;
              }
            } while (error);
          }

        } while (input.equals("3") || input_6.equals("4") || input_7.equals("4"));

        System.out.println("Exit.....");
        System.out.println("----------------------------");


    }
}
```