**Περιληψη:**

Η κλαση Organization αποτελεί τον οργανισμό που είναι υπεύθυνος για τις δωρεές.

Οι χρηστες (abstract κλάσης User) του συστήματος αυτού είναι είτε δωρητές (υποκλάσης Donator) είτε επωφελούμενου (υποκλάσης Beneficiary) είτε διαχειριστές (υποκλάσης Admin).

Οι παροχές του οργανισμού (abstract κλάσης Entities) αφορούν είτε υλικά (υποκλάσης Materials) είτε υπηρεσίες (υποκλάσης Services).

Κάθε αίτημα για δωρεά ή αίτηση παροχων και η επεξεργασία αυτών γίνεται μέσω της κλάσης RequestDonation.

Η κλάση RequestDonationList αποτελεί μια ομαδοποίηση αιτημάτων και προσφέρει τρόπους για την επεξεργασία τους. Οι υποκλάσεις της Offers και Requests αφορούν ομάδες αιτημάτων που ανήκουν σε δωρητές και σε επωφελούμενους αντίστοιχα.

Η κλάση Input χρησιμοποιείται ως wrapper της Scanner ώστε οι εισαγόμενες πληροφορίες να γίνονται δεκτές μόνο όταν πληρούν ορισμένα κριτήρια. Αυτό επιτυγχάνεται με αμυντικό προγραμματισμό. Έτσι, η Input αντικαθιστά την Scanner επειδή είναι πιο αξιόπιστη.

Η κλάση Menu αποτελεί την επικοινωνία ανάμεσα στο χρήστη και το πρόγραμμα. Έχει μεθόδους που παίρνουν πληροφορίες απτο χρήστη, τις επεξεργάζονται και αλλάζουν κατάλληλα την κατάσταση του προγράμματος.

Η κλάση Main αρχικοποιεί τον οργανισμό, τους χρήστες και τις παροχές του.

Οι κλάσεις των εξαιρέσεων ελέγχουν ακραίες ή ανεπιθύμητες καταστάσεις. Αυτές είναι: η εισαγωγή άκυρης τιμής απτο χρήστη, η προσθήκη ενός υπάρχοντος αντικειμένου το οποίο δεν μπορεί να εμφανίζεται πάνω από μια φορά, η αίτηση υπερβολικής ποσότητας από ένα προϊόν, η αίτηση ποσότητας που υπερβαίνει το απόθεμα του οργανισμού, η αδυναμία εύρεσης ενός αιτήματος δωρεάς ή αίτησης παροχής, ή η γενικότερη αδυναμία αντιμετώπισης κάποιου αιτήματος δωρεάς ή αίτησης παροχής.

**Σύντομη επεξήγηση κλάσεων:**

* User

→ Η κλάση αυτή είναι abstract (χωρίς στιγμιότυπα) και αναπαριστά έναν χρήστη του συστήματος του οργανισμού.

→ Έχει 2 πεδία.

1) Το πεδίο name τύπου String είναι το όνομα του χρήστη.

2) Το πεδίο phone τύπου String είναι το κινητό τηλέφωνο του χρήστη που χρησιμοποιείται για την ταυτοποίηση του.

→ Έχει 1 δημιουργό που παίρνει δύο ορίσματα String name και String phone με τα οποία αρχικοποιεί τα πεδία του χρήστη.

→ Έχει 2 μεθόδους.

1) Την getPhone() η οποία επιστρέφει το κινητό του χρήστη.

2) Την getName()η οποία επιστρέφει το όνομα του χρήστη.

* Donator

→ Υποκλάση της User. Αναπαριστά έναν δωρητή του οργανισμού.

→ Έχει1 πεδίο, το offersList τύπου Offers το οποίο αναπαριστά μια λίστα με τις δωρεές που προτίθεται να προσφέρει στον οργανισμό ο χρήστης.

→Έχει έναν δημιουργό ο οποίος χρησιμοποιεί μονάχα τον δημιουργό της User.

→ Έχει 7 μεθόδους.

Αυτές είναι οι addOffer(), removeOffer(), modifyOffer(), monitorOffers(), commitOffers(), resetOffers() οι οποίες είναι wrappers μεθόδων της κλάσης Offers και χειρίζονται τις λειτουργίες προσθήκης, αφαίρεσης και τροποποίησης δωρεών, εμφάνισης όλων των δωρεών, εξυπηρέτησης τους και εκκαθάρισης της λίστας.

Επίσης η μέθοδος getOffersList() η οποία επιστρέφει τo offersList.

* Beneficiary

→Υποκλάση της User. Αναπαριστά έναν επωφελούμενο του οργανισμού.

→ Έχει 3 πεδία.

1) Το πεδίο noPersons τύπου int το οποίο αναπαριστά τον αριθμό των ατόμων στην οικογένεια του χρήστη.

2) Το πεδίο receivedList τύπου RequestDonationList το οποίο αναπαριστά μια λίστα με τις δωρεές που έχει παραλάβει ο χρήστης.

3) Το πεδίο requestsList τύπου Requests το οποίο αναπαριστά μια λίστα με τις δωρεές που θέλει να παραλάβει ο χρήστης.

→Έχει έναν δημιουργό ο οποίος χρησιμοποιεί μονάχα τον δημιουργό της User.

→Έχει 15 μεθόδους.

Είναι οι addReceived(), removeReceived(), modifyReceived, monitorReceived(), resetReceivedList() καιaddRequest(), removeRequest(), modifyRequest(), monitorRequests(), resetRequests(), commitRequests() οιοποίες εκτέλουν αντίστοιχες λειτουργείες με τις wrappers της κλάσης Donator, με την διαφορά ότι είναι wrappers των RequestDonationList (για το receivedList) και Requests (για το requestsList).

Επίσης είναι οι getNoPersons() και setNoPersons(), οι οποίες επιστρέφουν και αλλάζουν αντίστοιχα τον αριθμό των μελών της οικογένειας του Beneficiary, και οι getReceivedList() και getRequestsList() οι οποίες επιστρέφουν το receivedList και requestsList αντίστοιχα.

* Admin

→ Υποκλάση της User. Αναπαριστά τον διαχειριστή του συστήματος.

→ Έχει 1 πεδίο isAdmin τύπου Boolean με τιμή true.

→Έχει 1 δημιουργό ο οποίος χρησιμοποιεί μονάχα τον δημιουργό της User.

→Δεν έχει καμία μέθοδο (πέρα από αυτές που κληρονομεί).

* Entity

→ Η κλάση αυτή είναι abstract(χωρίς στιγμιότυπα) και αναπαριστά ένα είδος δωρεάς.

→ Έχει 4 πεδία.

1) Το πεδίο name τύπου String το οποίο αναπαριστά το όνομα της δωρεάς.

2) Το πεδίο description τύπου String το οποίο είναι η περιγραφή της δωρεάς.

3) Το πεδίο id τύπου int το οποίο αποτελεί τον αριθμό ταυτότητας της δωρεάς.

4) Το πεδίο category τύπου int το οποίο ανάλογα με την τιμή του δηλώνει μια διαφορετική κατηγορία δωρεάς (για τις υλοποιημένες κατηγορίες Material και Service είναι 0 και 1 αντίστοιχα).

→ Έχει 2 δημιουργούς.

Ο πρώτος δεν παίρνει κανένα όρισμα και είναι κενός, δεν κάνει αρχικοποίηση με τιμές.

Ο δεύτερος έχει ένα όρισμα για κάθε πεδίο ώστε να τα αρχικοποιεί με τα αντίστοιχα ορίσματα.

→ Έχει 11 μεθόδους.

Από αυτές, οι 8 είναι setters και getters για τα 4 πεδία της κλάσης.

Οι υπόλοιπες τρεις είναι η abstractgetDetails(), η getEntityInfo() και η toString(). Η πρώτη επιστρέφει διαφορετικές πληροφορίες ανάλογα με τον τύπο του Entity, ενώ η δεύτερη επιστρέφει στοιχεία που είναι κοινά για όλα τα Entities. Η toString() τις συνδυάζει για να επιστρέψει σε Stringμια πλήρη επισκόπηση του Entity.

* Material

→Υποκλάση της Entity. Αναπαριστά μια υλική δωρεά.

→ Έχει 3 πεδία, ταlevel1, level2, level3 τα οποία αντιστοιχούν στην ποσότητα που δικαιούται να λάβει από τον οργανισμό ένας επωφελούμενος ανάλογα με τον αριθμό των μελών της οικογένειάς του.

→ Έχει έναν δημιουργό με 7 ορίσματα, 4 από τα οποία αντιστοιχούνται στο constructor της υπερκλάσης μέσω της super() και τα άλλα τρία αντιστοιχούνται στα level.

→Έχει 4 μεθόδους: 3 getters για τα level και υλοποίηση της getDetails() από την Entity.

* Service

→Υποκλάση της Entity. Αναπαριστά μια δωρεά υπηρεσίας.

→ Δεν έχει κανένα πεδίο (πέρα από αυτά που κληρονομεί).

→ Έχει 1 δημιουργό ο οποίος χρησιμοποιεί μονάχα τον δημιουργό της Entity.

→ Έχει 1 μέθοδο, η οποία αποτελεί υλοποίηση της getDetails() της Entity.

* RequestDonation

→ Η κλάση αυτή αναπαριστά ένα αίτημα/παροχή ενός Entity.

→Έχει 2 πεδία.

1) Το πεδίο entity τύπου Entity το οποίο είναι το Entity που αφορά το αίτημα/παροχή.

2) Το πεδίο quantity τύπου doubleτο οποίο αναπαριστά την ποσότητα του αιτήματος/παροχής.

→ Έχει 1 δημιουργό, ο οποίος με δύο ορίσματα entity τύπου Entity και quantity τύπου double αρχικοποιεί τα entity και quantity αντίστοιχα.

→ Έχει 3 μεθόδους: getters για τα entity και quantity και setter για το quantity.

* RequestDonationList

→Η κλάση αυτή αναπαριστά μια λίστα από RequestDonation.

→ Έχει 1 πεδίο rdEntities τύπου ArrayList<RequestDonation> που περιέχει τα αιτήματα/παροχές.

→ Δεν έχει constructor (πέρα από τον προκαθορισμένο).

→ Έχει 7 μεθόδους.

1) Την get() η οποία παίρνοντας ως όρισμα ένα int id ψάχνει την λίστα για ένα RequestDonation με το ίδιο id. Αν το βρει το επιστρέφει, αλλιώς εγείρεται εξαίρεση τύπου RequestDonationNotFoundException.

2) Την add() η οποία προσθέτει το RequestDonation rd όρισμά της στην λίστα εφόσον υποστηρίζεται το entity του από τον οργανισμό. Αν δεν υποστηρίζεται, εγείρεται RequestDonationNotFoundException. Μετά από την επιβεβαίωση ύπαρξης του entity στον οργανισμό, ελέγχεται αν το rd έχει το ίδιο id με κάποιο από τα RequestDonation του rdEntities. Αν ναι προστιθέται η ποσότητα του rd στο κατάλληλο RequestDonation του rdEntities, αλλιώς προστίθεται το ίδιο το rd στο rdEntities.

3) Την remove() η οποία παίρνοντας ως όρισμα ένα int id, αφαιρεί, αν υπάρχει, το RequestDonation που περιέχει entity με id ίσο με το όρισμα. Αν δεν υπάρχει, εξαιτίας της χρήσης της get(), εγείρεται εξαίρεση τύπου RequestDonationNotFoundException.

4) Την modify() η οποία παίρνοντας ως ορίσματα ένα int id και ένα double quantity βρίσκει το RequestDonationμε το κατάλληλο id και θέτει το quantity του να είναι ίσο με το όρισμα quantity. Αν δεν υπάρχει RequestDonationμε το κατάλληλο id, εξαιτίας της χρήσης της get(), εγείρεται εξαίρεση τύπου RequestDonationNotFoundException.

5) Την monitor() η οποία δημιουργεί ένα String,το οποίο αποτελεί κατάλληλα μορφοποιημένη λίστα με τα ονόματα και τις ποσότητες των RequestDonation του rdEntities, και το επιστρέφει.

6) Την reset() η οποία καθαρίζει το rdEntities.

7) Την getRdEntities() η οποία επιστρέφει το rdEntities.

* Offers

→Υποκλάση της RequestDonationList. Αναπαριστά μια λίστα παροχών RequestDonation.

→Έχει 1 πεδίο donor τύπου Donator, το οποίο δείχνει σε ποιον χρήστη/δωρητή αντιστοιχεί αυτή η λίστα. Αυτό χρησιμεύει στις μεθόδους που χρειάζεται να γνωρίζουν σε ποιόν ανήκει η λίστα, καθώς δεν χρειάζεται σε κάθε περίπτωση νέο όρισμα.

→ Έχει 1 δημιουργό, ο οποίος παίρνει ως όρισμα τον Donator στον οποίο αντιστοιχεί η λίστα και θέτει την τιμή του donor σε αυτό.

→ Έχει 1 μέθοδο, την commit(). Αυτή προσπαθεί να εξυπηρετήσει όλες τις παροχές τις λίστας, προσθέτοντας κάθε μία στο πεδίο currentDonations τύπου RequestDonationList με την χρήση της currentDonations.add(). Κάθε RequestDonationNotFoundException,το οποίο εγείρεται από την add() στην περίπτωση που κάποιο entity δεν υποστηρίζεται πια από τον οργανισμό, αντιμετωπίζεται με την εκτύπωση κατάλληλου μηνύματος στην οθόνη. Στο τέλος γίνεται καθαρισμός της λίστας παροχών του donator.

* Requests

→Υποκλάση της RequestDonationList. Αναπαριστά μια λίστα αιτημάτων RequestDonation.

→ Έχει 1 πεδίο benef τύπου Beneficiary, το οποίο δείχνει σε ποιον χρήστη/επωφελούμενο αντιστοιχεί αυτή η λίστα. Αυτό χρησιμεύει στις μεθόδους που χρειάζεται να γνωρίζουν σε ποιόν ανήκει η λίστα, καθώς δεν χρειάζεται σε κάθε περίπτωση νέο όρισμα.

→ Έχει 1 δημιουργό, ο οποίος παίρνει ως όρισμα τον Beneficiary στον οποίο αντιστοιχεί η λίστα και θέτει την τιμή του benef σε αυτό.

→ Έχει 5 μεθόδους.

1) Την overridden add() η οποία παίρνει ως όρισμα ένα RequestDonation rd και ελέγχει πρώτα αν ο οργανισμός έχει αρκετό απόθεμα για να το εξυπηρετήσει. Αν όχι εγείρεται InsufficientQuantityException. Ύστερα καλείται η validRequestDonation() με όρισμα το rdγια να ελεγχθεί αν ο Beneficiary δικαιούται το rd. Αν δεν εγείρει αυτή η μέθοδος το InvalidRequestException, τελικά καλείται η add() της υπερκλάσης (super.modify()).

2) Την overridden modify() η οποία παίρνει ως όρισμα ένα intid και ένα double quantity. Κάνοντας τους ίδιους ελέγχους με την παραπάνω add(), αλλά χρησιμοποιώντας την validRequestDonation() με ορίσματα id και quantity. Αν κανένας έλεγχος δεν εγείρει Exception, καλείται η modify() της υπερκλάσης (super.modify()).

3) Την validRequestDonation() με όρισμα RequestDonation rd. Σχεδιάστηκε για χρήση με την add(). Εφόσον το rd είναι Material, ελέγχει αν τοquantity του rd ξεπερνά το όριο που επιτρέπεται στον benef, συγκρίνοντας το quantity του rd + το quantity του RequestDonation με το ίδιο id που περιέχεται στην λίστα + το quantity του RequestDonation με το ίδιο id που περιέχεται στο receivedList του πεδίου benef με το κατάλληλο level του entity ανάλογα με τον αριθμό μελών της οικογένειας του Beneficiary (noPersonsτουbenef). Αν δεν επιτρέπεται εγείρεται InvalidRequestException.

4) Την validRequestDonation() με ορίσματα int id και double quantity (πολυμορφισμός). Σχεδιάστηκε για χρήση με την modify(). Ελέγχει πάλι αν επιτρέπεται το quantity στον benef με βάση την ποσότητα που έχει ήδη λάβει και το κατάλληλο level για το noPersons. Αν δεν επιτρέπεται εγείρει εξαίρεση InvalidRequestException.

5) Την commit() η οποία προσπαθεί να εξυπηρετήσει κάθε RequestDonationτης λίστας ελέγχοντας εκ νέου για κάθε ένα αν υπάρχει αρκετή ποσότητα στον οργανισμό και αν το δικαιούται ο benef. Για κάθε ένα που δεν εξυπηρετείται τυπώνεται κατάλληλο μήνυμα. Κάθε ένα που εξυπηρετείται αφαιρείται από την λίστα.

* Input

→ Η κλάση αυτή είναι υπεύθυνη για την εισαγωγή πληροφοριών από τον χρήστη ελέγχοντας οτι οι εισαγόμενες πληροφορίες τηρούν κάποια κριτήρια μέσω αμυντικού προγραμματισμού.

→ Έχει δυο πέδια.

Το πέδιο list είναι ένα ArrayList με στοιχεία τύπου String και αποτελεί την λίστα ελέγχου.

Το πεδίο pinf τυπου double είναι το άπειρο και είναι static ώστε να χρησιμοποιείται χωρίς την δημιουργία αντικειμένου. (Αυτό το πεδίο θα μπορούσε να πάει σε οποιαδήποτε κλάση και στην ουσία η έκφραση Double.POSITIVE\_INFINITY αντικαθίσταται με την Input.pinf) (Δεν χρησιμοποιείται μέσα στην κλάση Input)

→ Έχει τρεις δημιουργούς.

Ο πρώτος παίρνει ως όρισμα ένα String και δημιουργεί μια λίστα που την αναθέτει στην list με στοιχεία τα substrings που προκύπτουν αν τεμαχίσουμε το String στις θέσεις που έχει κενό. Έτσι, αν δώσουμε μια πρόταση ως παράμετρο τα στοιχεία της list θα είναι οι επιμέρους λέξεις.

Ο δεύτερος παίρνει ως όρισμα μια λίστα και την αναθέτει στην list.

Ο τρίτος λειτουργεί όπως ο πρώτος μόνο που έχει ένα παραπάνω όρισμα Stringsplitter που διευκρινίζει βάσει ποιου String θα τεμαχίστει το String str ώστε να προκύψει η λίστα.

→ Έχει 6 μεθόδους.

Η μέθοδος clearConsole() είναι στατική ώστε να μην απαιτείται η δημιουργία αντικειμένου για την χρήση της και χρησιμοποιείται για να καθαρίσει την οθόνη του προγράμματος. Δημιουργεί μια διαδικασία και καλεί την εντολή καθαρισμού της οθόνης σε μια γραμμή εντολών και συνδέεται με την κονσόλα που τρέχει το πρόγραμμα για να την καθαρίσει.

Η μέθοδος intInRange είναι στατική ώστε να μην απαιτείται η δημιουργία αντικειμένου για την χρήση της και παίρνει ως ορίσματα το μήνυμα που θα τυπώσει στον χρήστη, την ελάχιστη και την μέγιστη τίμη που μπορεί να βάλει ο χρήστης. Χρησιμοποιώντας ένα whileloop κάθε φορά που δεν μπορεί να φτάσει στην γραμμή return selection; επειδή έχει προκύψει exception επαναλαμβάνει την παρακάτω διαδικασία. Αρχικά τυπώνει το μήνυμα στον χρήστη, παίρνει την επόμενη γραμμή που εισάγει ο χρήστης από το πληκτρολόγιο και αν ο χρήστης προσπαθεί να κάνει interupt το πρόγραμμα πετάει NoSuchElementException. Ύστερα καθαρίζει την οθόνη. Μετά μετατρέπει την εισαγώμενη πληροφορία σε ακέραιο χρησιμοποιώντας την parseInt που πετάει NumberFormatException. Αν ο αριθμός που εισήγαγε ο χρήστης είναι μεγαλύτερος απτην αναμενόμενη μέγιστη τιμή ή μικρότερος απτην ελάχιστη τιμή τότε πετάει NotAcceptedValueException. Σε κάθε περίπτωση η κάθε εξαίρεση αντιμετωπίζεται ως NotAcceptedValueException και ειδοποιεί τον χρήστη μέσω την notifyMe() η οποία τυπώνει το κατάλληλο μήνυμα. Αν δεν προκύψει καμία εξαίρεση τότε επιστρέφεται ο αριθμός που έδωσε ο χρήστης. Αν προκύψει εξαίρεση η παραπάνω διαδικασία επαναλαμβάνεται.

Παρόμοια λειτουργεί και η strInput μόνο που επειδή πρόκειται για String δεν προκύπτουν εξαιρέσεις απτην μετατροπή του String σε int και δεν περιορίζουμε τις τιμές που μπορεί να πάρει το String.

Η checkStrInList χρησιμοποιείται ως μέθοδος αντικειμένου του οποίου έχουμε αρχικοποιήσει την list και παίρνει ως όρισμα το μήνυμα που τυπώνει στον χρήστη.

Ύστερα τυπώνει το μήνυμα, παίρνει την επόμενη γραμμή που εισάγει ο χρήστης από το πληκτρολόγιο και αν ο χρήστης προσπαθεί να κάνει interupt το πρόγραμμα πετάει NoSuchElementException. Στην συνέχεια καθαρίζει την οθόνη.

Μετά ελέγχει αν το String που έδωσε ο χρήστης περιέχεται στην list και αν όχι πετάει NotAcceptedValueException. Τέλος επιστρέφει το String που εισήγαγε ο χρήστης. Σε κάθε περίπτωση η κάθε εξαίρεση αντιμετωπίζεται ως NotAcceptedValueException και ειδοποιεί τον χρήστη μέσω την notifyMe() η οποία τυπώνει το κατάλληλο μήνυμα. Αν δεν προκύψει καμία εξαίρεση τότε επιστρέφεται το String που έδωσε ο χρήστης. Αν προκύψει εξαίρεση η παραπάνω διαδικασία επαναλαμβάνεται.

Παρόμοια λειτουργεί και η checkIntInList μόνο που πετάει ενα παραπάνω exception κατά την μετατροπή του String σε ακέραιο και επιστρέφει τον ακέραιο.

Η checkPhone τυπωνει το μήνυμα στο χρήστη παίρνει την επόμενη γραμμή που εισάγει ο χρήστης και αν αυτή είναι “exit” τότε επιστέφει “exit”. Αλλιώς ελέγχει αν αυτό που εισήγαγε ο χρήστης έχει μήκος 10 και αν δεν έχει τότε πετάει NotAcceptedValueException. Ύστερα χρησιμοποιεί το Long.parseLong ώστε αν αυτό που εισήγαγε ο χρήστης δεν μπορεί να μετατραπεί σε Long να πετάει NumberFormatException επειδή δεν μπορεί να είναι κινητό τηλέφωνο.

Σε κάθε περίπτωση η κάθε εξαίρεση αντιμετωπίζεται ως NotAcceptedValueException και ειδοποιεί τον χρήστη μέσω την notifyMe() η οποία τυπώνει το κατάλληλο μήνυμα. Αν δεν προκύψει καμία εξαίρεση τότε επιστρέφεται το String που έδωσε ο χρήστης. Αν προκύψει εξαίρεση η παραπάνω διαδικασία επαναλαμβάνεται.

* Menu

Έχει ενα στατικό πεδίο seconds το οποίο καθορίζει τον χρόνο που θα παγώνει η οθόνη όταν πετιέται notification.

→Έχει 28 μεθόδους.

1) Η showOffersOptions επιστρέφει την επιλογή ενός δωρητή όταν βλέπει αυτά που είναι διατεθειμένος να προσφέρει(κωδικοποιημένα).

2) Η showRequestsOptions επιστρέφει την επιλογή ενός επωφελούμενου όταν βλέπει αυτά που είναι διατεθειμένος να ζητήσει(κωδικοποιημένα).

3) Η modifyOffersGetName επιστρέφει το όνομα της προσφοράς που θέλει να τροποποιήσει ο δωρητής ή ότι θέλει να πάει στο προηγούμενο στάδιο του μενού.

4) Η modifyRequestsGetName επιστρέφει το όνομα της αίτησης που θέλει να τροποποιήσει ο δωρητής ή ότι θέλει να πάει στο προηγούμενο στάδιο του μενού.

5) Η getOfferEntityName ανάλογα με την κατηγορία που επιλέγουμε επιστρέφει είτε το όνομα του entity της κατηγορίας που ο χρήστης θέλει να προσφέρει είτε "back" για να δηλώσει την επιθυμία του χρήστη να πάει στο προηγούμενο στάδιο του μενού.

6) Η getRequestEntityName ανάλογα με την κατηγορία που επιλέγουμε επιστρέφει είτε το όνομα του entity της κατηγορίας που ο χρήστης θέλει να ζητήσει είτε "back" για να δηλώσει την επιθυμία του χρήστη να πάει στο προηγούμενο στάδιο του μενού.

7) Η confirm για να επιβεβαιώσει ο χρήστης τις προηγούμενες επιλογές του.

8) Η welcome αποτελεί το καλωσόρισμα προς τον χρήστη.

9) Η bye αποτελεί τον αποχαιρετισμό προς τον χρήστη

10) Η getPhone παίρνει το τηλέφωνο του χρήστη και το επιστρέφει.

11) Η register εγγράφει έναν χρήστη στο σύστημα και επιστρέφει 2 αν η εγγραφή έγινε επιτυχημένα ενώ επιστρέφει 0 αν όχι.

12) Η login επιστρέφει το είδος του λογαριασμού στον οποίο ανήκει το τηλέφωνο του ορίσματος (κωδικοποιημένα).

13) Η donatorStartMenu εμφανίζει το κυρίως μενού του δωρητή και επιστρέφει την επιλογή της επόμενης ενέργειας του (κωδικοποιημένα).

14) Η beneficiaryStartMenu εμφανίζει το κυρίως μενού του επωφελούμενου και επιστρέφει την επιλογή της επόμενης ενέργειας του (κωδικοποιημένα).

15) Η donatorAddOfferChooseCategory επιστρέφει την επιλογή κατηγορίας (υλικά ή υπηρεσίες) που επιθυμεί να προσφέρει ένας δωρητής (κωδικοποιημένα).

16) Η beneficiaryRequestDonationChooseCategory επιστρέφει την επιλογή κατηγορίας (υλικά ή υπηρεσίες) που επιθυμεί να ζητήσει ένας δωρητής (κωδικοποιημένα).

17) Η adminStartMenu εμφανίζει το κυρίως μενού του διαχειριστή και επιστρέφει την επιλογή της επόμενης ενέργειας του (κωδικοποιημένα)

18) Η adminChooseCategory επιστρέφει την κατηγορία που θέλει να επιλέξει ο διαχειριστής ή οτι θέλει να πάει στο προηγούμενο στάδιο του μενού (κωδικοποιημένα).

19) Η adminViewMaterials εμφανίζει τα υλικά στον διαχειριστή και επιστρέφει το υλικό που επιλέγει ή "back" αν θέλει να πάει στο προηγούμενο στάδιο του μενού.

20) Η adminViewServices εμφανίζει τις υπηρεσίες στον διαχειριστή και επιστρέφει το υλικό που επιλέγει ή "back" αν θέλει να πάει στο προηγούμενο στάδιο του μενού.

21) Η adminMonitorOrganization επιστρέφει την επιλογή του διαχειριστή όταν θέλει να κάνει monitor τον οργανισμό (κωδικοποιημένα).

22) Η adminMonitorBeneficiaries επιστρέφει το κινητό του επωφελούμενου που ο διαχειριστής θέλει να κάνει monitorή "back" αν θέλει να πάει στο προηγούμενο στάδιο του μενού.

23) Η adminManageBeneficiary επιστρέφει τι θέλει να κάνει ο admin όταν κάνει monitor έναν συγκεκριμένο επωφελούμενο (κωδικοποιημένα).

24) Η adminMonitorDonators επιστρέφει το κινητό του δωρητή που ο διαχειριστής θέλει να κάνει monitorή "back" αν θέλει να πάει στο προηγούμενο στάδιο του μενού.

25) Η adminManageDonator επιστρέφει τι θέλει να κάνει ο admin όταν κάνει monitor έναν συγκεκριμένο δωρητή (κωδικοποιημένα).

26) Η handleUnregistered επιστρέφει την επιλογή ενός μη εγγεγραμμένου χρήστη.

27) Η notification εμφανίζει στην οθόνη ενα μήνυμα προς τον χρήστη που διαρκεί μερικά δευτερόλεπτα.

28) Η startMenu είναι η βασική συνάρτηση που περιέχει το μενού της εφαρμογής.

Πιο συγκεκριμένα, μόλις εκτελούμε το πρόγραμμα καλωσορίζει τον χρήστη καλώντας την welcome. Ύστερα αρχικοποιεί την κατάσταση state στην οποία βρίσκεται το μενού στην τιμή 0 και το τηλέφωνο του εκάστοτε χρήστη στην τιμή "phone". Μετά υπάρχει ένα while loop που χρησιμοποιεί μια δομή switch η οποία έχει 6 καταστάσεις, τις 0, 1, 2, 3, 4, 5.

Το λουπ αυτό επαναλαμβάνεται όσο η κατάσταση του προγράμματος είναι διάφορη του -1.

Στην κατάσταση 0 το πρόγραμμα παίρνει το τηλέφωνο απτο χρήστη και αν αντί για το τηλέφωνο του δώσει "exit" η κατάσταση του προγράμματος γίνεται -1 και το πρόγραμμα τερματίζεται. Αν ο χρήστης δώσει έγκυρο τηλέφωνο τότε ελέγχεται αν το τηλέφωνο αυτό ανήκει σε ήδη εγγεγραμμένο χρήστη. Αν ναι τότε ξαναμπαίνει στο λουπ με state = 2. Αν όχι τότε τον ενημερώνει ότι δεν είναι εγγεγραμμένος και τον ρωτά αν θέλει να εγγραφεί, αν θέλει να εισάγει ένα διαφορετικό τηλέφωνο ή αν θέλει να βγει απτο πρόγραμμα. Αν ο χρήστης θέλει να εγγραφεί τότε η κατάσταση του προγράμματος γίνεται 1 και το λουπ ξανατρέχει με την καινούργια κατάσταση. Αν ο χρήστης θέλει να ξαναεισάγει το τηλέφωνο του τότε η κατάσταση γίνεται 0 αλλιώς γίνεται -1 και το πρόγραμμα τερματίζεται.

Στην κατάσταση 1 ο χρήστης εγγράφεται στον οργανισμό δίνοντας τα απαραίτητα στοιχεία και αν η εγγραφή ήταν επιτυχής η κατάσταση γίνεται 2 ενώ αν η εγγραφή αποτύχει η κατάσταση γίνεται 0.

Στην κατάσταση 2 αντιστοιχίζεται το τηλέφωνο του χρήστη με το είδος του λογαριασμού του. Αν ο χρήστης είναι δωρητής τότε η κατάσταση του προγράμματος γίνεται 3, αν είναι επωφελούμενος γίνεται 4, αν είναι διαχειριστής γίνεται 5 ενώ αν δεν είναι τίποτα από τα παραπάνω το πρόγραμμα τερματίζεται λόγω κάποιου απρόβλεπτου τεχνικού προβλήματος (δεν υπάρχει τέτοια περίπτωση).

Στην κατάσταση 3 αντιστοιχεί το μενού ενός δωρητή. Δημιουργείται ένα καινούργιο scope και ορίζεται η εσωτερική κατάσταση innerstate του προγράμματος και ορίζεται ο δωρητής που αντιστοιχεί στο τηλέφωνο που πήραμε από την κατάσταση 0. Ύστερα δηλώνεται το entity που ο δωρητής θέλει είτε να προσφέρει και αρχικοποιείται η ποσότητα του σε τιμή 0. Μετά τυπώνεται το μήνυμα του καλωσορίσματος του χρήστη μαζί με τα στοιχεία του. Ύστερα υπάρχει ένα while loop που χρησιμοποιεί μια δομή switch η οποία έχει 10 εσωτερικές καταστάσεις, τις 0, 1, 2, 10, 20, 21, 22, 40, 41, 42. Ο μόνος τρόπος για να βγει το πρόγραμμα από το λουπ του δωρητή είναι η εσωτερική κατάσταση να είναι -1 ή -2 (αρνητική) και τότε η κατάσταση του προγράμματος θα γίνει -1 ή 0 αντίστοιχα και αυτό θα συμβεί αν ο χρήστης αποφασίσει να τερματίσει το πρόγραμμα ή να κάνει logout αντίστοιχα.

Στην εσωτερική κατάσταση 0 τυπώνεται το βασικό μενού του δωρητή. Αν ο δωρητής επιλέξει να προσθέσει μια προσφορά η εσωτερική κατάσταση γίνεται 1, αν επιλέξει να δει τις προσφορές του γίνεται 2, αν επιλέξει να κάνει commit τις προσφορές του γίνεται 40, αν επιλέξει να κάνει logout γίνεται -1 ενώ αν επιλέξει να τερματίσει το πρόγραμμα γίνεται -2.

Στην εσωτερική κατάσταση 1 ο χρήστης επιλέγει αν θέλει να προσφέρει υλικά, υπηρεσίες ή αν θέλει να πάει στο προηγούμενο στάδιο του μενού. Αν θέλει να προσφέρει υλικά τότε η εσωτερική κατάσταση γίνεται 1, αν θέλει να προσφέρει υπηρεσίες γίνεται 10 ενώ αν θέλει να επιστρέψει πίσω γίνεται 0.

Στην εσωτερική κατάσταση 2 ο χρήστης εισάγει το όνομα του προϊόντος που θέλει να προσφέρει η "back" αν θέλει να πάει πίσω. Αν θέλει να πάει πίσω τότε η εσωτερική κατάσταση γίνεται 1. Αλλιώς ελέγχεται αν το όνομα του προϊόντος που θέλει να προσφέρει αντιστοιχεί σε προϊόν που υποστηρίζεται από τον οργανισμό και αν δεν υποστηρίζεται τότε πετάει NotAcceptedValueException και μπαίνει εκ νέου στην εσωτερική κατάσταση 2 μέχρι ο χρήστης να δώσει είτε έγκυρο όνομα είτε "back". Αν το όνομα που δώσει ο χρήστης είναι έγκυρο τότε εμφανίζει τις πληροφορίες του προϊόντος και ρωτά τον χρήστη αν θέλει να το προσφέρει. Αν όχι ξαναμπαίνει εκ νέου στην εσωτερική κατάσταση 2. Αν ναι τότε ζητάει από το χρήστη την ποσότητα που θέλει να προσφέρει και καλεί την addOffer του δωρητή για την συγκεκριμένη προσφορά και αν προκύψει εξαίρεση ειδοποιεί τον δωρητή. Ύστερα επιστρέφει στην εσωτερική κατάσταση 0 είτε η προσθήκη ήταν επιτυχής είτε όχι.

Στην εσωτερική κατάσταση 10 το πρόγραμμα λειτουργεί με ανάλογο τρόπο όπως στην 2 μόνο που αντί για προϊόντα ο δωρητής προσφέρει υπηρεσίες.

Στην εσωτερική κατάσταση 20 ελέγχεται αν η λίστα με αυτά που θέλει να προσφέρει ο δωρητής είναι άδεια και αν ναι τυπώνει το κατάλληλο μήνυμα. Αν όχι εμφανίζονται αυτά που θέλει να προσφέρει ο δωρητής και μπορεί να επιλέξει να τροποποιήσει τις προσφορές του, να τις διαγράψει, να τις κάνει commit ή να πάει ένα βήμα πίσω στο μενού. Αν επιλέξει το πρώτο η εσωτερική κατάσταση γίνεται 21, αν το δεύτερο 22, αν το τρίτο 40 και αν το τέταρτο 0.

Στην εσωτερική κατάσταση 21 ο χρήστης εισάγει το όνομα του προϊόντος την προσφερόμενη ποσότητα του οποίου θέλει να τροποποιήσει ή "back". Στην δεύτερη περίπτωση το πρόγραμμα πηγαίνει στην εσωτερική κατάσταση 20. Μετά μετατρέπει το όνομα του προϊόντος στο αντίστοιχο entity και επιβεβαιώνει οτι η μετατροπή ήταν επιτυχής. Αν δεν ήταν το πρόγραμμα μπαίνει εκ νέου στην εσωτερική κατάσταση 21. Ύστερα ο δωρητής επιλέγει αν θέλει να τροποποιήσει την προσφορά, να την διαγράψει ή να πάει ένα βήμα πίσω στο μενού. Αν επιλέξει το πρώτο τότε εισάγει την νέα ποσότητα που θέλει να προσφέρει και καλείται η modifyOffer του συγκεκριμένου δωρητή με την καινούργια προσφορά και η εσωτερική κατάσταση ξαναγίνεται 21. Αν επιλέξει το δεύτερο τότε καλείται η removeOffer του δωρητή ώστε να αφαιρεθεί η προσφορά του δωρητή σύμφωνα με το όνομα που έδωσε και η εσωτερική κατάσταση ξαναγίνεται 21. Αν επιλέξει το τρίτο η εσωτερική κατάσταση γίνεται 20.

Στην εσωτερική κατάσταση 22 καλείται η resetOffers του δωρητή ώστε να αδειάσει την λίστα προσφορών του και ειδοποιείται ότι οι προσφορές του διαγράφηκαν επιτυχώς. Ύστερα επιστρέφει στην εσωτερική κατάσταση 0.

Στην εσωτερική κατάσταση 40 γίνονται commit οι προσφορές του δωρητή. Αν η λίστα προσφορών του είναι άδεια ειδοποιείται ότι δεν έχει προσφορές αλλιώς ειδοποιείται ότι το commit έγινε επιτυγχώς και καλείται η commitOffers του. Σε κάθε περίπτωση επιστρέφει στην εσωτερική κατάσταση 0.

Στην εσωτερική κατάσταση 41 η εσωτερική κατάσταση γίνεται -1 ώστε να γίνει logout. Αποτελεί μια περιττή κατάσταση αλλά συμβάλει στη δομή του κώδικα.

Στην εσωτερική κατάσταση 42 η εσωτερική κατάσταση γίνεται -2 ώστε να τερματιστεί το πρόγραμμα. Αποτελεί μια περιττή κατάσταση αλλά συμβάλει στη δομή του κώδικα.

Στην κατάσταση 4 αντιστοιχεί το μενού ενος επωφελούμενου το οποίο είναι ανάλογο με αυτό του δωρητή με αντίστοιχες εσωτερικές καταστάσεις.

Στην κατάσταση 5 αντιστοιχεί το μενού του διαχειριστή.

Δημιουργείται ένα καινούργιο scope και ορίζεται η εσωτερική κατάσταση innerstate του προγράμματος και ορίζεται ο διαχειριστής του οργανισμού. Μετά αρχικοποιείται ένας δωρητής και ένας επωφελούμενος οι οποιοί αναφέρονται στον εκάστοτε χρήστη που ο διαχειριστής θέλει να επιβλέψει.

Ύστερα δηλώνεται το entity που ο διαχειριστής θέλει να διαχειριστεί και αρχικοποιείται η ποσότητα του σε τιμή 0.

Μετά τυπώνεται το μήνυμα του καλωσορίσματος του διαχειριστή μαζί με τα στοιχεία του. Ύστερα υπάρχει ενα while loop που χρησιμοποιεί μια δομή switch η οποία έχει 12 εσωτερικές καταστάσεις, τις 0, 1, 2, 10, 11, 20, 21, 22, 100, 110, 200, 210. Ο μόνος τρόπος για να βγει το πρόγραμμα από το λουπ του διαχειριστή είναι η εσωτερική κατάσταση να είναι -1 ή -2 (αρνητική) και τότε η κατάσταση του προγράμματος θα γίνει -1 ή 0 αντίστοιχα και αυτό θα συμβεί αν ο χρήστης αποφασίσει να τερματίσει το πρόγραμμα ή να κάνει logout αντίστοιχα.

Στην εσωτερική κατάσταση 0 τυπώνεται το βασικό μενού του διαχειριστή. Αν ο διαχειριστής επιλέξει να επιβλέψει τις υπάρχουσες προσφορές η εσωτερική κατάσταση γίνεται 1. Αν επιλέξει να επιβλέψει τους χρήστες του οργανισμού γίνεται 2. Αν επιλέξει να κάνει logout γίνεται -1. Αν επιλέξει να τερματίσει το πρόγραμμα γίνεται -2.

Στην εσωτερική κατάσταση 1 ο διαχειριστής επιλέγει την κατηγορία που θέλει να επιβλέψει(υλικά/υπηρεσίες). Αν θέλει να επιβλέψει τα υλικά η εσωτερική κατάσταση γίνεται 10, αν επιλέξει να επιβλέψει τις υπηρεσίες γίνεται 11 ενώ αν επιλέξει να πάει ένα βήμα πίσω στο μενού γίνεται 0.

Στην εσωτερική κατάσταση 2 ο διαχειριστής επιλέγει αν θέλει να επιβλέψει τους επωφελούμενους, αν θέλει να επιβλέψει τους δωρητές, αν θέλει να αδειάσει τις λίστες με όσα έχουν λάβει οι επωφελούμενοι ή αν θέλει να πάει ένα βήμα πίσω στο μενού. Έτσι, η εσωτερική κατάσταση γίνεται 20, 21, 22 ή 0 αντίστοιχα.

Στην εσωτερική κατάσταση 10 ο διαχειριστής εισάγει το όνομα του υλικού που θέλει να επιβλέψει. Αν το όνομα είναι έγκυρο η εσωτερική κατάσταση γίνεται 100 αλλιώς γίνεται 1.

Στην εσωτερική κατάσταση 11 ο διαχειριστής εισάγει το όνομα της υπηρεσίας που θέλει να επιβλέψει. Αν το όνομα είναι έγκυρο η εσωτερική κατάσταση γίνεται 110 αλλιώς γίνεται 1.

Στην εσωτερική κατάσταση 20 ο διαχειριστής εισάγει το τηλέφωνο του επωφελούμενου που θέλει να επιβλέψει το οποίο μετατρέπεται σε αντικείμενο επωφελούμενου. Αν το τηλέφωνο είναι έγκυρο η εσωτερική κατάσταση γίνεται 200 αλλιώς γίνεται 2.

Στην εσωτερική κατάσταση 21 ο διαχειριστής εισάγει το τηλέφωνο του δωρητή που θέλει να επιβλέψει το οποίο μετατρέπεται σε αντικείμενο δωρητή. Αν το τηλέφωνο είναι έγκυρο η εσωτερική κατάσταση γίνεται 210 αλλιώς γίνεται 2.

Στην εσωτερική κατάσταση 22 καλείται η resetReceivedList σε κάθε επωφελούμενο καθαρίζοντας την λίστα με όσα έχουν λάβει και ειδοποιεί κατάλληλα τον διαχειριστή. Η εσωτερική κατάσταση γίνεται 2.

Στην εσωτερική κατάσταση 100 προβάλλεται η ποσότητα του επιλεγμένου υλικού που προσφέρεται από τον οργανισμό και δίνεται η δυνατότητα στον διαχειριστή να πάει ένα βήμα πίσω στο μενού κάνοντας την εσωτερική κατάσταση 10.

Στην εσωτερική κατάσταση 110 προβάλλονται οι ώρες της επιλεγμένης υπηρεσίας που προσφέρονται από τον οργανισμό και δίνεται η δυνατότητα στον διαχειριστή να πάει ένα βήμα πίσω στο μενού κάνοντας την εσωτερική κατάσταση 11.

Στην εσωτερική κατάσταση 200 ο διαχειριστής επιλέγει αν θέλει να αδειάσει την λίστα με όσα έχει παραλάβει ο επωφελούμενος, αν θέλει να διαγράψει τον επωφελούμενο από τον οργανισμό ή αν θέλει να πάει ένα βήμα πίσω στο μενού. Στην πρώτη περίπτωση η εσωτερική κατάσταση γίνεται 200 ενώ στις άλλες 2 γίνεται 20.

Στην εσωτερική κατάσταση 210 ο διαχειριστής επιλέγει αν θέλει να διαγράψει τον δωρητή από τον οργανισμό ή αν θέλει να πάει ένα βήμα πίσω στο μενού. Η εσωτερική κατάσταση γίνεται 21.

Όταν ο χρήστης αποφασίσει να τερματίσει το πρόγραμμα τυπώνεται αποχαιρετιστήριο μήνυμα.

* Organization

→Έχει 7 πεδία.

Την στατική μεταβλητή org η οποία αποτελεί το αντικείμενο τύπου οργανισμού που δημιουργήθηκε τελευταίο ώστε για ευκολία να αποφεύγεται να περνιέται ως παράμετρος σε κάθε συνάρτηση που θέλει να χρησιμοποιήσει τον οργανισμό αφού στην περίπτωση μας υπάρχει μόνο ένας και δεν υπάρχει περίπτωση σύγχυσης.

Το όνομα του οργανισμού.

Τον διαχειριστή του οργανισμού.

Την λίστα με τα entities που υποστηρίζει ο οργανισμός.

Την λίστα με τους δωρητές.

Την λίστα με τους επωφελούμενους.

Την λίστα με τις υπάρχουσες δωρεές.

→Ο δημιουργός του οργανισμού αρχικοποιεί το όνομα, τον διαχειριστή και την λίστα με τις υπάρχουσες δωρεές. Επίσης αρχικοποιεί την μεταβλητή org αναθέτοντας της το αντικείμενο που μόλις δημιουργήθηκε. Επειδή δημιουργούμε μόνο ένα αντικείμενο οργανισμού δεν αντιμετωπίζουμε το πρόβλημα η org να αλλάξει τιμή λόγω δημιουργίας άλλου οργανισμού.

→Έχει 31 μεθόδους.

1) Η getStrMaterialList επιστρέφει μια λίστα με στοιχεία τύπου String που περιέχει τα ονόματα των υλικών που περιέχονται στην entityList.

2) Η getStrServiceList επιστρέφει μια λίστα με στοιχεία τύπου String που περιέχει τα ονόματα των υπηρεσιών που περιέχονται στην entityList.

3) Η getOrg είναι η getter της org.

4) ΗgetEntityList έιναι η getter της entityList.

5) Η getDonatorList είναι η getter της donatorList.

6) Η getBeneficiaryList είναι η getter της beneficiaryList.

7) Η getEntityFromName επιστρέφει ένα entity της entityList δωσμένου του ονόματος του και της κατηγορίας στην οποία ανήκει. Αν το entity δεν υπάρχει στην entityList επιστρέφεται ένα Material που υποδηλώνει ότι το ζητούμενο entity δεν βρέθηκε.

8) Η getCurrentDonations είναι η getter της currentDonations.

9) Η addOffer προσθέτει μια προσφορά στην λίστα των υπάρχουσων προσφορών και πετάει RequestDonationException αν η προσθήκη αποτύχει.

10) Η strMaterials επιστρέφει ένα String με τα ονόματα των υλικών που υπάρχουν στον οργανισμό με κατάλληλη μορφοποίηση.

11) Η strServices επιστρέφει ένα Stringμε τα ονόματα των υπηρεσιών που υπάρχουν στον οργανισμό με κατάλληλη μορφοποίηση.

12) Η getCategories επιστρέφει ένα String με τα ονόματα των κατηγοριών και το πλήθος των προσφερόμενων αγαθών ή υπηρεσιών της κάθε κατηγορίας.

13) Η getNoOfMaterials επιστρέφει τον αριθμό των υλικών που υπάρχουν στην entityList.

14) Η getNoOfServices επιστρέφει τον αριθμό των υπηρεσιών που υπάρχουν στην entityList.

15) Η getName είναι getter για το όνομα του οργανισμού.

16) Η setName είναι setter για το όνομα του οργανισμού.

17) Η setAdmin είναι setter για τον διαχειριστή του οργανισμού.

18) Η getAdmin είναι getter για τον διαχειριστή του οργανισμού.

19) Η addEntity προσθέτει ένα entity στο entityList του οργανισμού και αν αυτό υπάρχει ήδη πετάει AlreadyExistsException ειδοποιώντας τον χρήστη με την notifyMe.

20) Η removeEntity αφαιρεί ένα entity από την entityList του οργανισμού.

21) Η insertDonator προσθέτει έναν donator στο donatorList του οργανισμού και αν αυτός υπάρχει ήδη πετάει AlreadyExistsException ειδοποιώντας τον χρήστη με την notifyMe και επιστρέφοντας 0. Αν η προσθήκη είναι επιτυχής επιστρέφει 2.

22) Η removeDonator αφαιρεί έναν donator από την donatorList του οργανισμού.

23) Η insertBeneficiary προσθέτει έναν beneficiary στο beneficiaryList του οργανισμού και αν αυτός υπάρχει ήδη πετάει AlreadyExistsException ειδοποιώντας τον χρήστη με την notifyMe και επιστρέφοντας 0. Αν η προσθήκη είναι επιτυχής επιστρέφει 2.

24) Η removeBeneficiary αφαιρεί έναν beneficiary από την donatorList του οργανισμού.

25) Η listEntities επιστρέφει ένα String με όλα τα entities κατάλληλα μορφοποιημένα.

26) Η listDonators επιστρέφει ενα String με όλους τους donators και τα τηλέφωνα τους κατάλληλα μορφοποιημένα.

27) Η listBeneficiaries επιστρέφει ένα String με όλους τους beneficiaries και τα τηλέφωνα τους κατάλληλα μορφοποιημένα.

28) Η checkAccount παίρνει ως όρισμα ένα τηλέφωνο και αν υπάρχει χρήστης με αυτό το τηλέφωνο επιστρέφει 2 αλλιώς επιστρέφει 1.

29) Η whatIsThisPhone παίρνει ως όρισμα ένα τηλέφωνο και αν αυτό ανήκει σε έναν δωρητή επιστρέφει "donator", αν ανήκει σε έναν επωφελούμενο επιστρέφει "beneficiary", αν αυτό ανήκει στον διαχειριστή επιστρέφει "admin" αλλιώς επιστρέφει "error".

30) Η getDonatorFromPhone παίρνει ως όρισμα ένα τηλέφωνο και επιστρέφει τον δωρητή στον οποίο ανήκει και αν δεν ανήκει σε κανέναν δωρητή επιστρέφει έναν ‘ελαττωματικό’ δωρητή.

31) Η getBeneficiaryFromPhone παίρνει ως όρισμα ένα τηλέφωνο και επιστρέφει τον επωφελούμενο στον οποίο ανήκει και αν δεν ανήκει σε κανέναν επωφελούμενο επιστρέφει έναν ‘ελαττωματικό’ επωφελούμενο.

* Main

→Η κλάση αυτή είναι υπεύθυνη για την αρχικοποίηση του συστήματος.

→ Στην μοναδική της μέθοδο main() προστίθενται στο σύστημα 1 Admin, 2Donator, 3 Beneficiary, 3 Material και 3 Service. Ύστερα τίθεται σε εκκίνηση το menu.

* AlreadyExistsException

→Η κλάση αυτή αναπαριστά εξαίρεση που εγείρεται όταν υπάρχει ήδη κάτι που ζητείται να δημιουργηθεί/εισαχθεί.

→ Έχει 1 πεδίο String msg το οποίο αναπαριστά το μήνυμα που θα τυπώνει το exception. Έχει defaultτιμή "Already exists".

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msg και έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

* NotAcceptedValueException

→Η κλάση αυτή αναπαριστά εξαίρεση που εγείρεται όταν δίνεται μη επιτρεπτή τιμή.

→ Έχει 1 πεδίο String msg το οποίο αναπαριστά το μήνυμα που θα τυπώνει το exception. Έχει defaultτιμή "Not accepted value".

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msg και έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

* RequestDonationException

→Η κλάση αυτή αναπαριστά εξαίρεση που σχετίζεται με τον χειρισμό RequestDonation.

→ Έχει 1 πεδίο String msg το οποίο αναπαριστά το μήνυμα που θα τυπώνει το exception. Έχει default τιμή "We were unable to register your request.\nPlease try again.".

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msg και έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

* InsufficientQuantityException

→Υποκλάση της RequestDonationException. Η κλάση αυτή αναπαριστά εξαίρεση που εγείρεται όταν υπάρχει ήδη κάτι που ζητείται να δημιουργηθεί/εισαχθεί.

→ Έχει 1 πεδίο String msg το οποίο αναπαριστά το μήνυμα που θα τυπώνει το exception. Έχει default τιμή "You ask too much!"

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msgκαι έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

* RequestDonationNotFoundException

→Υποκλάση της RequestDonationException. Η κλάση αυτή αναπαριστά εξαίρεση που εγείρεται όταν δεν βρέθηκε ένα RequestDonation.

→ Δεν έχει κανένα πεδίο (πέρα από αυτά που κληρονομεί).

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msgκαι έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

* InvalidRequestException

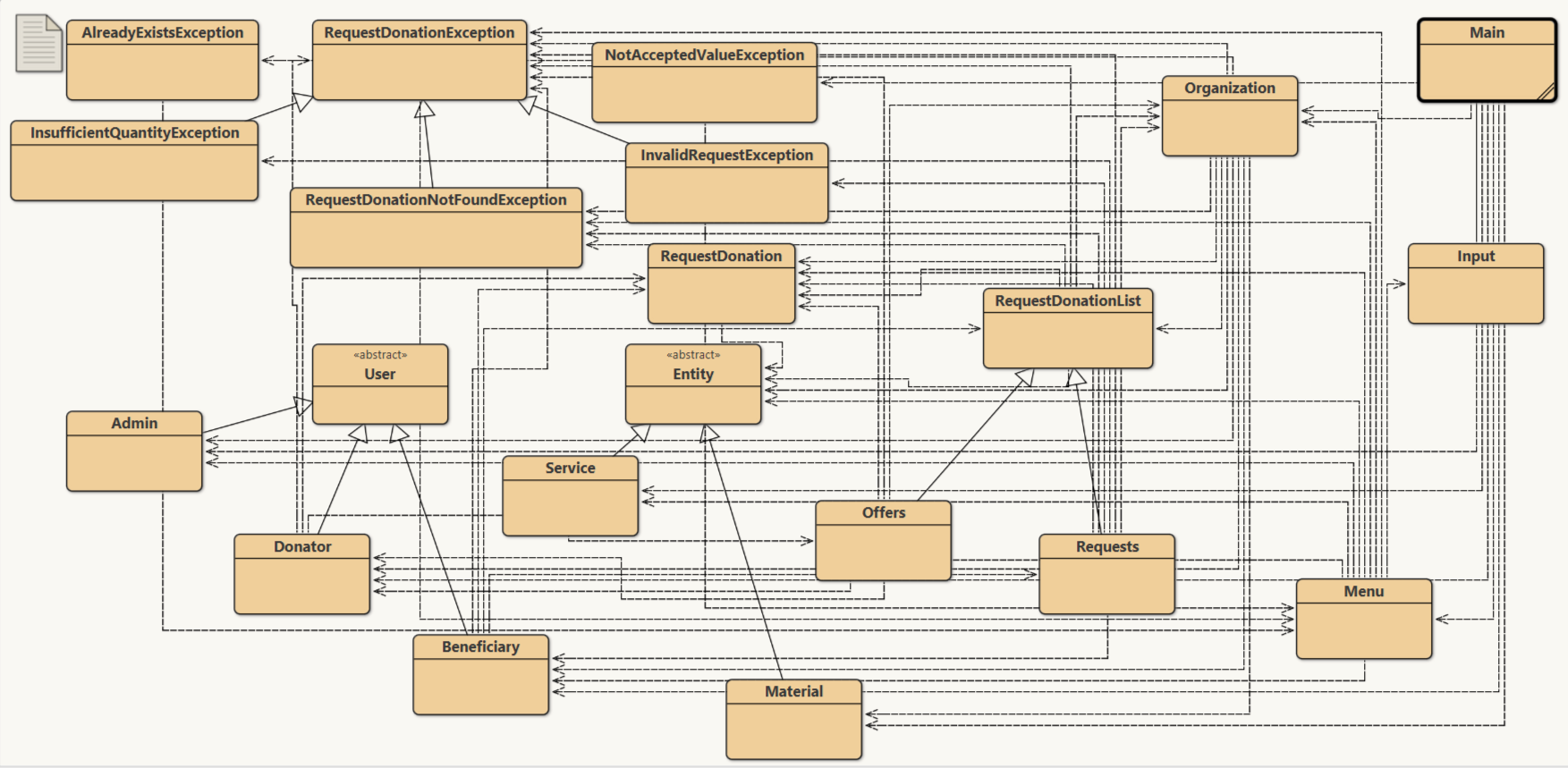
→Υποκλάση της RequestDonationException. Η κλάση αυτή αναπαριστά εξαίρεση που εγείρεται όταν δεν δικαιούται τα περιεχόμενα ενός αιτήματος o beneficiary που το υπέβαλε.

→ Έχει 1 πεδίο String msg το οποίο αναπαριστά το μήνυμα που θα τυπώνει το exception. Έχει default τιμή "You don’t deserve it!".

→Έχει 2 δημιουργούς, έναν χωρίς ορίσματα που αρχικοποιεί με την default τιμή το msgκαι έναν με όρισμα String msg για να αρχικοποιεί το msg με το όρισμα.

→ Έχει 1 μέθοδο, την notifyMe(), η οποία κάνει χρήση της Menu.notification() για να τυπώσει το msg.

**Διάγραμμα UML:**



**Κώδικας κλάσεων:**

* User

import java.util.\*;

abstract class User

{

private String name;

private String phone;

public User (String name, String phone)

{

this.name = name;

this.phone = phone;

}

public String getPhone()

{

return phone;

}

public String getName()

{

return name;

}

}

* Donator

public class Donator extends User

{

private Offers offersList = new Offers(this);

public Donator(String name, String phone)

{

super(name, phone);

}

public void addOffer(RequestDonation rd) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if "rd.entity" is not found in "Organization.entityList".

{

offersList.add(rd);

}

public void removeOffer(int id) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if no RequestDonation with the id is found in "offersList".

{

offersList.remove(id);

}

public void modifyOffer(int id, double quantity) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if no RequestDonation with the id is found in "offersList".

{

offersList.modify(id, quantity);

}

public String monitorOffers()

{

return "CURRENT OFFERS\n" + offersList.monitor();

}

public void commitOffers()

{

offersList.commit();

}

public void resetOffers()

{

offersList.reset();

}

public Offers getOffersList()

{

return offersList;

}

}

* Beneficiary

import java.util.ArrayList;

public class Beneficiary extends User

{

private int noPersons = 1;

private RequestDonationList receivedList = new RequestDonationList();

private Requests requestsList = new Requests(this);

public Beneficiary(String name, String phone)

{

super(name, phone);

}

public RequestDonationList getReceivedList()

{

return receivedList;

}

public void addReceived(RequestDonation rd) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if "rd.entity" is not found in "Organization.entityList".

{

receivedList.add(rd);

}

public void modifyReceived(int id, double quantity) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if no RequestDonation with the id is found in "receivedList".

{

receivedList.modify(id, quantity);

}

public void removeReceived(int id) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if no RequestDonation with the id is found in "receivedList".

{

receivedList.remove(id);

}

public String monitorReceived()

{

return "RECEIVED REQUESTS\n" + receivedList.monitor();

}

public void resetReceivedList()

{

receivedList.reset();

}

public Requests getRequestsList()

{

return requestsList;

}

public void addRequest(RequestDonation rd) throws RequestDonationException

//See add() in class Requests for exception handling.

{

requestsList.add(rd);

}

public void modifyRequest(int id, double quantity) throws RequestDonationException

//See modify() in class Requests for exception handling.

{

requestsList.modify(id, quantity);

}

public void removeRequest(int id) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if no RequestDonation with the id is found in "requestsList".

{

requestsList.remove(id);

}

public String monitorRequests()

{

return "CURRENT REQUESTS\n" + requestsList.monitor();

}

public void commitRequests()

{

requestsList.commit();

}

public void resetRequests()

{

requestsList.reset();

}

public int getNoPersons()

{

return noPersons;

}

public void setNoPersons(int noPersons)

{

this.noPersons = noPersons;

}

}

* Admin

public class Admin extends User

{

boolean isAdmin = true ;

public Admin(String name, String phone)

{

super(name, phone);

}

}

* Entity

abstract class Entity

{

private String name;

private String description;

private int id;

private int category; // 0 for material AND 1 for service

abstract String getDetails();

public Entity() {}

public Entity(String name, String description, int id, int category)

{

this.name = name;

this.description = description;

this.id = id;

this.category = category;

}

public int getCategory() {return category;}

public void setCategory(int category) {this.category = category;}

private String getEntityInfo()

{

return "Name: " + name + "\nDescription: " + description + "\nID: " + id;

}

public String toString()

{

return getEntityInfo() + "\n" + getDetails();

}

public String getName()

{

return name;

}

public String getDescription()

{

return description;

}

public int getId()

{

return id;

}

public void setName(String name)

{

this.name = name;

}

public void setDescription(String description)

{

this.description = description;

}

public void setId(int id)

{

this.id = id;

}

}

* Material

public class Material extends Entity

{

private final double level1;

private final double level2;

private final double level3;

public Material(String name, String description,int id, int category,

double level1, double level2, double level3)

{

super(name, description, id, category);

this.level1 = level1;

this.level2 = level2;

this.level3 = level3;

}

public double getLevel1()

{

return level1;

}

public double getLevel2()

{

return level2;

}

public double getLevel3()

{

return level3;

}

public String getDetails()

{

return "Type: Material\nLevel 1: " + level1 + "\nLevel 2: " +

level2 + "\nLevel 3: " + level3;

}

}

* Service

public class Service extends Entity

{

public Service() {}

public Service(String name, String description, int id, int category)

{

super(name, description, id, category);

}

public String getDetails()

{

return "Type: Service";

}

}

* RequestDonation

import java.util.\*;

public class RequestDonation

{

private Entity entity;

private double quantity = 0;

public RequestDonation(Entity entity, double quantity)

{

this.entity = entity;

this.quantity = quantity;

}

public Entity getEntity()

{

return entity;

}

public double getQuantity()

{

return quantity;

}

public void setQuantity(double quantity)

{

this.quantity = quantity;

}

}

* RequestDonationList

import java.util.\*;

public class RequestDonationList

{

private ArrayList<RequestDonation> rdEntities = new ArrayList<RequestDonation>();

public RequestDonation get(int id) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if "rd.entity" is not found in "rdEntities".

{

for(RequestDonation rd: rdEntities)

{

if(rd.getEntity().getId() == id) return rd;

}

throw new RequestDonationNotFoundException();

}

public void add(RequestDonation rd) throws RequestDonationException

//Throws RequestDonationNotFoundException

//if "rd.entity" is not found in "Organization.entityList".

{

int flag = 0;

for(Entity entity: Organization.getOrg().getEntityList())

{

if(rd.getEntity().getId() == entity.getId()) flag = 1;

}

if(flag == 0) throw new RequestDonationNotFoundException("" +

"This entity is not supported by our organization.");

for(RequestDonation rdon: rdEntities)

{

if(rdon.getEntity().getId() == (rd.getEntity()).getId())

{

rdon.setQuantity(rdon.getQuantity() + rd.getQuantity());

return;

}

}

rdEntities.add(rd);

}

public void remove(int id) throws RequestDonationException

//See get() for exception handling.

{

rdEntities.remove(this.get(id));

}

public void modify(int id, double quantity) throws RequestDonationException

//See get() for exception handling.

{

get(id).setQuantity(quantity);

}

public String monitor()

{

String rds = "";

int no = 1;

for(RequestDonation rdlist: rdEntities)

{

rds += "-" + no + "-\n" + "Name: " + rdlist.getEntity().getName() +

" ID: " + rdlist.getEntity().getId() +

" Quantity: " + (long)rdlist.getQuantity() + "\n";

no++;

}

return rds + "\n";

}

public void reset()

{

rdEntities.clear();

}

public ArrayList<RequestDonation> getRdEntities()

{

return rdEntities;

}

}

* Offers

public class Offers extends RequestDonationList

{

private Donator donor;

public Offers(Donator donor)

{

this.donor = donor;

}

public void commit()

{

for(RequestDonation rdlist: getRdEntities())

{

try

{

Organization.getOrg().getCurrentDonations().add(rdlist);

}

catch(RequestDonationException e)

{

try {

throw new RequestDonationException("This entity is" +

" no longer supported by the organization ("

+ rdlist.getEntity().getName() + ").");

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

}

this.reset();

}

}

* Requests

import java.util.\*;

public class Requests extends RequestDonationList

{

private Beneficiary benef;

public Requests(Beneficiary benef)

{

this.benef = benef;

}

public void add(RequestDonation rd) throws RequestDonationException

//InsufficientQuantityException, InvalidRequestException

//and RequestDonationNotFoundException.

{

boolean alreadyInRequests = true;

try {

get(rd.getEntity().getId());

} catch(RequestDonationException rde) {

alreadyInRequests = false;

}

try {

if (Organization.getOrg().getCurrentDonations().get(rd.getEntity().getId()).getQuantity()

< rd.getQuantity() + ( alreadyInRequests?

get(rd.getEntity().getId()).getQuantity():

0 ) )

{

throw new InsufficientQuantityException("Not enough stock.");

//Is thrown when the organization doesn't have enough stock.

}

} catch (RequestDonationNotFoundException rde) {

throw new InsufficientQuantityException("Not enough stock.");

//Is thrown when the organization doesn't have enough stock.

}

validRequestDonation(rd); //Throws InvalidRequestException.

super.add(rd); //Throws RequestDonationNotFoundException

//if "rd.entity" is not found in "Organization.entityList".

}

public void modify(int id, double quantity) throws RequestDonationException

//InsufficientQuantityException, InvalidRequestException

//and RequestDonationNotFoundException.

{

try{

if(Organization.getOrg().getCurrentDonations().get(id).getQuantity() < quantity)

{

throw new InsufficientQuantityException("Not enough stock.");

//Is thrown when the organization doesn't have enough stock.

}

} catch (RequestDonationNotFoundException rde) {

throw new InsufficientQuantityException("Not enough stock.");

//Is thrown when the organization doesn't have enough stock.

}

validRequestDonation(id, quantity);

//Throws InvalidRequestException (See validRequestDonation()).

super.modify(id, quantity);

//Throws RequestDonationNotFoundException if no RequestDonation

//with the id is found in the instance which runs the method.

}

public void validRequestDonation(RequestDonation rd) throws RequestDonationException

//Throws InvalidRequestException when the beneficiary

//should not be allowed the request due to quantity per beneficiary.

{

boolean alreadyInRequests = true;

try {

get(rd.getEntity().getId());

} catch(RequestDonationException rde) {

alreadyInRequests = false;

}

boolean alreadyInReceived = true;

try {

benef.getReceivedList().get(rd.getEntity().getId()).getQuantity();

} catch(RequestDonationException rde) {

alreadyInReceived = false;

}

double alreadyRequestedQuantity = ( alreadyInRequests?

get(rd.getEntity().getId()).getQuantity():

0 );

double alreadyReceivedQuantity = ( alreadyInReceived?

benef.getReceivedList().get(rd.getEntity().getId()).getQuantity():

0 );

if(rd.getEntity().getCategory() == 0)

{

if(benef.getNoPersons() < 2)

{

if(((Material)rd.getEntity()).getLevel1()

< rd.getQuantity() + alreadyRequestedQuantity +

alreadyReceivedQuantity)

throw new InvalidRequestException("You don't deserve it!");

}

else if(benef.getNoPersons() < 5)

{

if(((Material)rd.getEntity()).getLevel2()

< rd.getQuantity() + alreadyRequestedQuantity +

alreadyReceivedQuantity)

throw new InvalidRequestException("You don't deserve it!");

}

else

{

if(((Material)rd.getEntity()).getLevel3()

< rd.getQuantity() + alreadyRequestedQuantity +

alreadyReceivedQuantity)

throw new InvalidRequestException("You don't deserve it!");

}

}

}

public void validRequestDonation(int id, double quantity) throws RequestDonationException

//Same exception as above plus RequestDonationNotFoundException

//If no RequestDonation with the given id exists in the list.

{

boolean alreadyInReceived = true;

try {

benef.getReceivedList().get(id).getQuantity();

} catch(RequestDonationException rde) {

alreadyInReceived = false;

}

double alreadyReceivedQuantity = ( alreadyInReceived?

benef.getReceivedList().get(id).getQuantity():

0 );

if(get(id).getEntity().getCategory() == 0)

{

if(benef.getNoPersons() < 2)

{

if(((Material)get(id).getEntity()).getLevel1() < quantity +

alreadyReceivedQuantity) throw new InvalidRequestException("" +

"You don't deserve it!");

}

else if(benef.getNoPersons() < 5)

{

if(((Material)get(id).getEntity()).getLevel2() < quantity +

alreadyReceivedQuantity) throw new InvalidRequestException("" +

"You don't deserve it!");

}

else

{

if(((Material)get(id).getEntity()).getLevel3() < quantity +

alreadyReceivedQuantity) throw new InvalidRequestException("" +

"You don't deserve it!");

}

}

}

public void commit()

{

ArrayList<RequestDonation> rdentities = new ArrayList<RequestDonation>();

for(int i = 0; i < getRdEntities().size(); i++)

rdentities.add(getRdEntities().get(i));

double subtraction;

for(RequestDonation rd: rdentities)

{

try

{

try{

subtraction =

Organization.getOrg().getCurrentDonations().get(rd.getEntity().getId()).getQuantity()

- rd.getQuantity();

} catch(RequestDonationNotFoundException e) {

throw new InsufficientQuantityException();

}

if(subtraction < 0) throw new InsufficientQuantityException();

validRequestDonation(rd.getEntity().getId(), rd.getQuantity());

Organization.getOrg().getCurrentDonations().modify(rd.getEntity().getId(), subtraction);

benef.addReceived(rd);

remove(rd.getEntity().getId());

}

catch(InsufficientQuantityException e)

{

try {

throw new RequestDonationException("The request concerning " +

rd.getEntity().getName() +

" of quantity " + rd.getQuantity() +

" cannot be at the moment handled because of insufficient stock.");

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

catch(InvalidRequestException e)

{

try {

throw new RequestDonationException("You aren't presently eligible for " +

rd.getQuantity() + " of " + rd.getEntity().getName());

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

catch(RequestDonationNotFoundException e)

{

try {

throw new RequestDonationException("The request concerning " +

rd.getEntity().getName() +

" cannot be at the moment handled because " +

"it is no longer provided by the organization.");

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

catch(RequestDonationException e)

{

try {

throw new RequestDonationException("An unknown error has occured (" +

rd.getEntity().getName() + " & " +rd.getQuantity() + ").");

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

}}}

* Input

import java.util.\*;

import java.io.IOException;

public class Input

{

private ArrayList<String> list = new ArrayList<String>();

public static double pinf = Double.POSITIVE\_INFINITY;

public Input(String str)

{

list = new ArrayList(Arrays.asList(str.split(" ")));

}

public Input(ArrayList<String> list)

{

this.list = list;

}

public Input(String str, String splitter)

{

list = new ArrayList(Arrays.asList(str.split(splitter)));

}

public static void clearConsole()

{

try{

new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();

} catch (Exception e) {}

}

public static int intInRange(String msg, double min, double max)

{

while (true) {

try{

System.out.println(msg);

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

Input.clearConsole();

int selection = Integer.parseInt(ui);

if (selection < min || selection > max)

throw new NotAcceptedValueException();

return selection;

} catch (NotAcceptedValueException e) {

e.notifyMe();

} catch (NumberFormatException e){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

} catch (NoSuchElementException n){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

}

}

}

public static String strInput(String msg)

{

while (true) {

try{

System.out.println(msg);

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

Input.clearConsole();

return ui;

} catch (NoSuchElementException a){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException d) {

d.notifyMe();

}

}

}

}

public String checkStrInList(String msg)

{

while (true) {

try{

System.out.println(msg);

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

Input.clearConsole();

if (list.contains(ui) == false)

throw new NotAcceptedValueException();

return ui;

} catch (NotAcceptedValueException e) {

e.notifyMe();

} catch (NumberFormatException e){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

} catch (NoSuchElementException n){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

}

}

}

public int checkIntInList(String msg)

{

while (true) {

try{

System.out.println(msg);

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

Input.clearConsole();

if (list.contains(ui) == false)

throw new NotAcceptedValueException();

int selection = Integer.parseInt(ui);

return selection;

} catch (NotAcceptedValueException e) {

e.notifyMe();

} catch (NumberFormatException e){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

} catch (NoSuchElementException n){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException v) {

v.notifyMe();

}

}

}

}

public static String checkPhone(String msg)

{

while (true) {

try{

System.out.println(msg);

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

Input.clearConsole();

if (ui.equals("exit"))

return "exit";

else if (ui.length() != 10)

throw new NotAcceptedValueException();

Long.parseLong(ui);

return ui;

} catch (NotAcceptedValueException e) {

e.notifyMe();

} catch (NumberFormatException e){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException r) {

r.notifyMe();

}

} catch (NoSuchElementException a){

try{

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException d) {

d.notifyMe();

}

}

}

}

}

* Menu

import java.util.\*;

public class Menu

{

public static double seconds = 2.5;

public int showOffersOptions(Donator donator)

{

int selection = (new Input("1 2 3 4")).checkIntInList("" +

donator.monitorOffers() +

"--> To select an offer enter 1.\n" +

"--> To remove all your offers enter 2.\n" +

"--> To commit your offers enter 3.\n" +

"--> To go back enter 4.");

if (selection == 1) return 21;

else if (selection == 2) return 22;

else if (selection == 3) return 40;

else return 0;

}

public int showRequestsOptions(Beneficiary beneficiary)

{

int selection = (new Input("1 2 3 4")).checkIntInList("" +

beneficiary.monitorRequests() +

"--> To select a request enter 1.\n" +

"--> To remove all your requests enter 2.\n" +

"--> To commit your requests enter 3.\n" +

"--> To go back enter 4.");

if (selection == 1) return 21;

else if (selection == 2) return 22;

else if (selection == 3) return 40;

else return 0;

}

public String modifyOffersGetName(Donator donator)

{

String input = "back";

for(RequestDonation rd: donator.getOffersList().getRdEntities())

{

input += "\t" + rd.getEntity().getName();

}

String selection = new Input(input, "\t").checkStrInList("" +

donator.monitorOffers() +

"--> Enter the name of the offer you wish to select.\n" +

"--> To go back enter \"back\".");

return selection;

}

public String modifyRequestsGetName(Beneficiary beneficiary)

{

String input = "back";

for(RequestDonation rd: beneficiary.getRequestsList().getRdEntities())

{

input += "\t" + rd.getEntity().getName();

}

String selection = new Input(input, "\t").checkStrInList("" +

beneficiary.monitorRequests() +

"--> Enter the name of the request you wish to select.\n" +

"--> To go back enter \"back\".");

return selection;

}

public String getOfferEntityName(int category)

{

return Input.strInput(

(category == 0?

Organization.getOrg().strMaterials() +

"--> To donate a material enter its name.\n":

Organization.getOrg().strServices() +

"--> To provide a service enter its name.\n") +

"--> To go back enter \"back\".");

}

public String getRequestEntityName(int category)

{

return Input.strInput(

(category == 0?

Organization.getOrg().strMaterials() +

"--> To request a material enter its name.\n":

Organization.getOrg().strServices() +

"--> To request a service enter its name.\n") +

"--> To go back enter \"back\".");

}

public int confirm()

{

return (new Input("1 2")).checkIntInList("--> To confirm enter 1.\n" +

"--> To cancel enter 2.\n");

}

public void welcome()

{

Input.clearConsole();

System.out.println("===============================\n"+

"Welcome to our donation system!\n"+

"===============================");

}

public void bye()

{

Input.clearConsole();

System.out.println("===========================================\n"+

"Thank you for trusting us! Have a nice day!\n"+

"===========================================");

}

public String getPhone()

{

String phone = (Input.checkPhone("Enter your phone number or enter 'exit' if you want to exit."));

return phone;

}

public int register(Organization org, String phone)

{

String name = Input.strInput("Enter your name.");

int selection = (new Input("1 2")).checkIntInList("--> To become a donator enter 1.\n"+

"--> To become a beneficiary enter 2.");

int no = 0;

if(selection==2)

no = Input.intInRange("" +

"Enter the number of individuals" +

" that your family consists of.", 1, 100);

int confirm = (new Input("1 2")).checkIntInList("--> To confirm your registration enter 1.\n"+

"--> To cancel your registration enter 2.");

if (confirm == 1 && selection == 1) {

return org.insertDonator(new Donator(name, phone));

} else if (confirm == 1 && selection == 2) {

Beneficiary benef = new Beneficiary(name, phone);

benef.setNoPersons(no);

return org.insertBeneficiary(benef);

} else return 0;

}

public int login(String phone, Organization org)

{

String account = org.whatIsThisPhone(phone);

if (account.equals("donator")) return 3;

else if (account.equals("beneficiary")) return 4;

else if (account.equals("admin")) return 5;

else {System.out.println("BUG!!!"); return -1;}

}

public int donatorStartMenu()

{

int selection = (new Input("1 2 3 4 5")).checkIntInList(""+

"-->To add an offer enter 1.\n"+

"-->To view your offers enter 2.\n"+

"-->To commit your offers enter 3.\n"+

"-->To logout enter 4.\n"+

"-->To exit enter 5.\n");

if (selection == 1) {

return 1;

} else if (selection == 2) {

return 20;

} else if (selection == 3) {

return 40;

} else if (selection == 4) {

return -1;

} else {

return -2;

}

}

public int beneficiaryStartMenu()

{

int selection = (new Input("1 2 3 4 5")).checkIntInList(""+

"-->To request a donation enter 1.\n"+

"-->To view your requests enter 2.\n"+

"-->To commit your requests enter 3.\n"+

"-->To logout enter 4.\n"+

"-->To exit enter 5.\n");

if (selection == 1) {

return 1;

} else if (selection == 2) {

return 20;

} else if (selection == 3) {

return 40;

} else if (selection == 4) {

return -1;

} else {

return -2;

}

}

public int donatorAddOfferChooseCategory(String categories)

{

int selection = (new Input("1 2 3")).checkIntInList(categories +

"--> To offer materials enter 1.\n" +

"--> To offer services enter 2.\n" +

"--> To go back enter 3.");

if (selection == 1) return 2;

else if (selection == 2) return 10;

else return 0;

}

public int beneficiaryRequestDonationChooseCategory(String categories)

{

int selection = (new Input("1 2 3")).checkIntInList(categories +

"--> To request materials enter 1.\n" +

"--> To request services enter 2.\n" +

"--> To go back enter 3.");

if (selection == 1) return 2;

else if (selection == 2) return 10;

else return 0;

}

public int adminStartMenu()

{

int selection = (new Input("1 2 3 4")).checkIntInList(""+

"-->To view the current available donations enter 1.\n"+

"-->To monitor the organization enter 2.\n"+

"-->To logout enter 3.\n"+

"-->To exit enter 4.\n");

if (selection == 1) {

return 1;

} else if (selection == 2) {

return 2;

} else if (selection == 3) {

return -1;

} else {

return -2;

}

}

public int adminChooseCategory(String categories)

{

int selection = new Input("1 2 3").checkIntInList(categories +

"-->To view materials enter 1.\n"+

"-->To view services enter 2.\n"+

"-->To go back enter 3.\n");

if (selection == 1) return 10;

else if (selection == 2) return 11;

else return 0;

}

public String adminViewMaterials()

{

String input = "back";

for(Entity entity: Organization.getOrg().getEntityList())

{

if (entity.getCategory() == 0) input += "\t" + entity.getName();

}

String selection = new Input(input, "\t").checkStrInList(Organization.getOrg().strMaterials()+

"-->To view an item enter its name.\n"+

"-->To go back enter \"back\".\n");

return selection;

}

public String adminViewServices()

{

String input = "back";

for(Entity entity: Organization.getOrg().getEntityList())

{

if(entity.getCategory() == 1) input += "\t" + entity.getName();

}

String selection = new Input(input, "\t").checkStrInList(Organization.getOrg().strServices()+

"-->To view an item enter its name.\n"+

"-->To go back enter \"back\".\n");

return selection;

}

public int adminMonitorOrganization()

{

int selection = new Input("1 2 3 4").checkIntInList(""+

"-->To list all beneficiaries enter 1.\n"+

"-->To list all donators enter 2.\n"+

"-->To reset all beneficiaries' received items enter 3.\n"+

"-->To go back enter 4.\n");

if (selection == 1) return 20;

else if (selection == 2) return 21;

else if (selection == 3) return 22;

else return 0;

}

public String adminMonitorBeneficiaries()

{

String input = "back";

for(Beneficiary benef: Organization.getOrg().getBeneficiaryList())

{

input += " " + benef.getPhone();

}

String selection = new Input(input).checkStrInList(Organization.getOrg().listBeneficiaries()+

"\n-->To select a beneficiary enter their phone number.\n"+

"-->To go back enter \"back\".\n");

return selection;

}

public int adminManageBeneficiary(Beneficiary benef)

{

String s = "'s ";

if(benef.getName().charAt(benef.getName().length()-1) == 's')

{

s = "' ";

}

int selection = new Input("1 2 3").checkIntInList(benef.monitorReceived()+

"-->To reset " + benef.getName() + s +

"received items enter 1.\n"+

"-->To delete " + benef.getName() +

" from the system enter 2.\n"+

"-->To go back enter 3.\n");

if(selection == 1)

{

benef.resetReceivedList();

return 200;

}

else if(selection == 2) Organization.getOrg().removeBeneficiary(benef);

return 20;

}

public String adminMonitorDonators()

{

String input = "back";

for(Donator donor: Organization.getOrg().getDonatorList())

{

input += " " + donor.getPhone();

}

String selection = new Input(input).checkStrInList(Organization.getOrg().listDonators()+

"\n-->To select a donator enter their phone number.\n"+

"-->To go back enter \"back\".\n");

return selection;

}

public int adminManageDonator(Donator donor)

{

String s = "'s ";

if(donor.getName().charAt(donor.getName().length()-1) == 's')

{

s = "' ";

}

int selection = new Input("1 2").checkIntInList(donor.monitorOffers()+

"-->To delete " + donor.getName() +

" from the system enter 1.\n"+

"-->To go back enter 2.\n");

if(selection == 1) Organization.getOrg().removeDonator(donor);

return 21;

}

public int handleUnregistered()

{

int state;

int selection = new Input("1 2 3").checkIntInList("\nYou are not registered!\n\n" +

"--> To register press 1\n" +

"--> To enter another phone number press 2\n" +

"--> To exit press 3\n");

if (selection == 1) state = 1;

else if (selection == 2) state = 0;

else state = -1;

return state;

}

public static void notification(String msg, double seconds)

{

Input.clearConsole();

System.out.println(msg);

try {

Thread.sleep((long)seconds\*1000);

} catch (InterruptedException ie) {}

Input.clearConsole();

}

public void startMenu()

{

this.welcome();

int state = 0;

String phone = "phone";

while(state != -1)

{

switch (state)

{

case 0:

phone = this.getPhone();

if (phone.equals("exit")) {state = -1; break;}

state = Organization.getOrg().checkAccount(phone);

if (state==2) break; //registered

state = this.handleUnregistered(); //unregistered

break;

case 1:

state = this.register(Organization.getOrg(), phone);

break;

case 2:

state = this.login(phone, Organization.getOrg());

break;

case 3:

//donator

{

int innerstate = 0;

Donator donator = Organization.getOrg().getDonatorFromPhone(phone);

Entity entity;

double quantity = 0;

System.out.println("Welcome " + donator.getName() + "!!!\n" +"Phone: " +

phone + "\nOrganization: "+Organization.getOrg().getName());

while (innerstate >= 0){

switch (innerstate){

case 0:

//donator.printDetails();

innerstate = this.donatorStartMenu();

break;

case 1: //add offer

innerstate = this.donatorAddOfferChooseCategory(Organization.getOrg().getCategories());

break;

case 2: //materials

{

String name = this.getOfferEntityName(0);

if (name.equals("back"))

{innerstate = 1; break;}

entity = Organization.getOrg().getEntityFromName(name, 0);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try

{

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->Would you like to donate this item? (y/n)").equals("n")) break;

} catch (RequestDonationException e) {

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: 0" +

"\n-->Would you like to donate this item? (y/n)").equals("n")) break;

}

quantity = Input.intInRange("Enter the quantity you would like to donate.",

1 , Input.pinf);

try {

donator.addOffer(new RequestDonation(entity, quantity));

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

innerstate = 0;

break;

case 10: //services

{

String name = this.getOfferEntityName(1);

if (name.equals("back"))

{innerstate = 1; break;}

entity = Organization.getOrg().getEntityFromName(name, 1);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try

{

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->Would you like to donate this service? (y/n)").equals("n")) break;

} catch (RequestDonationException e) {

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: 0" +

"\n-->Would you like to donate this service? (y/n)").equals("n")) break;

}

quantity = Input.intInRange("Enter the amount of hours you would like to provide.",

1 , Input.pinf);

try {

donator.addOffer(new RequestDonation(entity, quantity));

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

innerstate = 0;

break;

case 20: //show offers

if (donator.getOffersList().getRdEntities().size() == 0) {

this.notification("There are no offers.", Menu.seconds);

innerstate = 0;

} else {

innerstate = this.showOffersOptions(donator);

}

break;

case 21: //modify offers

{

String name = this.modifyOffersGetName(donator);

if (name.equals("back"))

{innerstate = 20; break;}

entity = Organization.getOrg().getEntityFromName(name, -1);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try{

quantity = donator.getOffersList().get(entity.getId()).getQuantity();

} catch (RequestDonationException e){

quantity = 0;

}

switch(new Input("1 2 3").checkIntInList(entity.toString()+

"\nQuantity: " + quantity +

"\n-->To modify this offer enter 1." +

"\n-->To remove this offer enter 2." +

"\n-->To go back enter 3.")){

case 1:

quantity = Input.intInRange("Enter the new quantity you would like to donate.",

1 , Input.pinf);

try {

donator.modifyOffer(entity.getId(), quantity);

} catch (RequestDonationException rde) {

rde.notifyMe();

}

break;

case 2:

try{

donator.removeOffer(entity.getId());

} catch (RequestDonationException e) {}

break;

}

}

innerstate = 20;

break;

case 22: //remove offers

donator.resetOffers();

this.notification("Offers removed successfully", Menu.seconds);

innerstate = 0;

break;

case 40: //commit

if (donator.getOffersList().getRdEntities().size() == 0) {

this.notification("There are no offers.", Menu.seconds);

} else {

this.notification("Your offers were commited successfully.", Menu.seconds);

donator.commitOffers();

}

innerstate = 0;

break;

case 41: //logout

innerstate = -1;

break;

case 42: //exit

innerstate = -2;

break;

}

}

if (innerstate == -2) state = -1;

else if (innerstate == -1) state = 0;

break;

}

case 4:

//beneficiary

{

int innerstate = 0;

Beneficiary beneficiary = Organization.getOrg().getBeneficiaryFromPhone(phone);

Entity entity;

double quantity = 0;

System.out.println("Welcome " + beneficiary.getName() + "!!!\n" +"Phone: " +

phone + "\nOrganization: "+Organization.getOrg().getName());

while (innerstate >= 0){

switch (innerstate){

case 0:

innerstate = this.beneficiaryStartMenu();

break;

case 1: //add request

innerstate = this.beneficiaryRequestDonationChooseCategory(

Organization.getOrg().getCategories());

break;

case 2: //materials

{

String name = this.getRequestEntityName(0);

if (name.equals("back"))

{innerstate = 1; break;}

entity = Organization.getOrg().getEntityFromName(name, 0);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try

{

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->Would you like to request this item? (y/n)").equals("n")) break;

} catch (RequestDonationException e) {

if(new Input("y n").checkStrInList(entity.toString()+ "\nQuantity: 0" +

"\n-->Would you like to request this item? (y/n)").equals("n")) break;

}

quantity = Input.intInRange("Enter the quantity you would like to request.",

1 , Input.pinf);

try {

beneficiary.addRequest(new RequestDonation(entity, quantity));

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

innerstate = 0;

break;

case 10: //services

{

String name = this.getRequestEntityName(1);

if (name.equals("back"))

{innerstate = 1; break;}

entity = Organization.getOrg().getEntityFromName(name, 1);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try

{

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->Would you like to request this service? (y/n)").equals("n")) break;

} catch (RequestDonationException e) {

if(new Input("y n").checkStrInList(entity.toString() + "\nQuantity: 0" +

"\n-->Would you like to request this service? (y/n)").equals("n")) break;

}

quantity = Input.intInRange("Enter the amount of hours you would like to request.",

1 , Double.POSITIVE\_INFINITY);

try {

beneficiary.addRequest(new RequestDonation(entity, quantity));

} catch (RequestDonationException rde) {

rde.notifyMe();

}

}

innerstate = 0;

break;

case 20: //show requests

if (beneficiary.getRequestsList().getRdEntities().size() == 0) {

this.notification("There are no requests.", Menu.seconds);

innerstate = 0;

} else {

innerstate = this.showRequestsOptions(beneficiary);

}

break;

case 21: //modify requests

{

String name = this.modifyRequestsGetName(beneficiary);

if (name.equals("back"))

{innerstate = 20; break;}

entity = Organization.getOrg().getEntityFromName(name, -1);

try{

if(entity.getName().equals("error"))

throw new NotAcceptedValueException();

} catch (NotAcceptedValueException nave) {

nave.notifyMe();

break;

}

try{

quantity = beneficiary.getRequestsList().get(entity.getId()).getQuantity();

} catch (RequestDonationException e){

quantity = 0;

}

switch(new Input("1 2 3").checkIntInList(entity.toString()+

"\nQuantity: " + quantity +

"\n-->To modify this request enter 1." +

"\n-->To remove this request enter 2." +

"\n-->To go back enter 3.")){

case 1:

quantity = Input.intInRange("Enter the new quantity you would like to request.",

1 , Input.pinf);

try {

beneficiary.modifyRequest(entity.getId(), quantity);

} catch (RequestDonationException rde) {

rde.notifyMe();

}

break;

case 2:

try{

beneficiary.removeRequest(entity.getId());

} catch (RequestDonationException e) {}

break;

}

}

innerstate = 20;

break;

case 22: //remove offers

beneficiary.resetRequests();

this.notification("Requests removed successfully", Menu.seconds);

innerstate = 0;

break;

case 40: //commit

if (beneficiary.getRequestsList().getRdEntities().size() == 0) {

this.notification("There are no requests.", Menu.seconds);

} else {

this.notification("Your requests were commited successfully.", Menu.seconds);

beneficiary.commitRequests();

}

innerstate = 0;

break;

case 41: //logout

innerstate = -1;

break;

case 42: //exit

innerstate = -2;

break;

}

}

if (innerstate == -2) state = -1;

else if (innerstate == -1) state = 0;

break;

}

case 5:

//admin

{

int innerstate = 0;

Admin admin = Organization.getOrg().getAdmin();

Beneficiary beneficiary = new Beneficiary("Dummy","404");

Donator donator = new Donator("Dummy","404");

Entity entity = new Service();

double quantity = 0;

System.out.println("Welcome (Administrator) " + admin.getName() + "!!!\n" +"Phone: " +

phone + "\nOrganization: "+Organization.getOrg().getName());

while (innerstate >= 0){

switch (innerstate){

case 0:

innerstate = this.adminStartMenu();

break;

case 1: //view

innerstate = this.adminChooseCategory(Organization.getOrg().getCategories());

break;

case 2: //monitor organization

innerstate = this.adminMonitorOrganization();

break;

case 10: //view materials

entity = Organization.getOrg().getEntityFromName(this.adminViewMaterials(), 0);

innerstate = 100;

if(entity.getId() == -1) innerstate = 1;

break;

case 11: //view services

entity = Organization.getOrg().getEntityFromName(this.adminViewServices(), 1);

innerstate = 110;

if(entity.getId() == -1) innerstate = 1;

break;

case 20: //list beneficiaries

beneficiary = Organization.getOrg().getBeneficiaryFromPhone(

this.adminMonitorBeneficiaries());

innerstate = 200;

if(beneficiary.getName().equals("error")) innerstate = 2;

break;

case 21: //list donators

donator = Organization.getOrg().getDonatorFromPhone(this.adminMonitorDonators());

innerstate = 210;

if(donator.getName().equals("error")) innerstate = 2;

break;

case 22: //reset beneficiaries

for(Beneficiary benef: Organization.getOrg().getBeneficiaryList())

{

benef.resetReceivedList();

}

this.notification("All beneficiaries' received items have been reset!", Menu.seconds);

innerstate = 2;

break;

case 100:

try{

innerstate = new Input("1").checkIntInList(entity.toString() + "\nQuantity: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->To go back enter 1.\n") + 9;

} catch (RequestDonationException e) {

innerstate = new Input("1").checkIntInList(entity.toString() +

"\nQuantity: 0\n-->To go back enter 1.\n") + 9;

}

break;

case 110:

try{

innerstate = new Input("1").checkIntInList(entity.toString() + "\nHours: " +

Organization.getOrg().getCurrentDonations().get(entity.getId()).getQuantity() +

"\n-->To go back enter 1.\n") + 10;

} catch (RequestDonationException e) {

innerstate = new Input("1").checkIntInList(entity.toString()+

"\nHours: 0\n-->To go back enter 1.\n") + 10;

}

break;

case 200: //beneficiary-specific submenu

innerstate = this.adminManageBeneficiary(beneficiary);

break;

case 210: //donator-specific submenu

innerstate = this.adminManageDonator(donator);

break;

}

}

if (innerstate == -2) state = -1;

else if (innerstate == -1) state = 0;

break;

}

}

}

this.bye();

}

}

* Organization

import java.util.\*;

public class Organization

{

private static Organization org;

private String name;

private Admin admin;

private ArrayList<Entity> entityList = new ArrayList<Entity>();

private ArrayList<Donator> donatorList = new ArrayList<Donator>();

private ArrayList<Beneficiary> beneficiaryList = new ArrayList<Beneficiary>();

private RequestDonationList currentDonations;

public Organization(String name, Admin admin)

{

this.name = name;

this.admin = admin;

this.org=this;

currentDonations = new RequestDonationList();

}

public ArrayList<String> getStrMaterialList()

{

ArrayList<String> names = new ArrayList<String>();

for (Entity entity: entityList)

{

if(entity.getCategory() == 0)

names.add(entity.getName());

}

return names;

}

public ArrayList<String> getStrServiceList()

{

ArrayList<String> names = new ArrayList<String>();

for (Entity entity: entityList)

{

if(entity.getCategory() == 1)

names.add(entity.getName());

}

return names;

}

public static Organization getOrg()

{

return org;

}

public ArrayList<Entity> getEntityList()

{

return entityList;

}

public ArrayList<Donator> getDonatorList()

{

return donatorList;

}

public ArrayList<Beneficiary> getBeneficiaryList()

{

return beneficiaryList;

}

public Entity getEntityFromName(String name, int category)

{

for (Entity entity: entityList)

{

if (entity.getName().equals(name) && (entity.getCategory() == category || category == -1))

return entity;

}

return new Material("error", "bug", -1, 0, 1, 2, 3);

}

public RequestDonationList getCurrentDonations()

{

return currentDonations;

}

public void addOffer(RequestDonation rd) throws RequestDonationException

{

try

{

currentDonations.add(rd);

}

catch (RequestDonationNotFoundException rdnfe)

{

rdnfe.notifyMe();

}

}

public String strMaterials()

{

String names = "";

names += "Materials:\n";

for (Entity entity: entityList)

{

if (entity.getCategory() == 0)

{

names += "\t-" + entity.getName() + "\n";

}

}

return names;

}

public String strServices()

{

String names = "";

names += "Services:\n";

for (Entity entity: entityList)

{

if (entity.getCategory() == 1)

{

names += "\t-" + entity.getName() + "\n";

}

}

return names;

}

public String getCategories()

{

return "Materials: (" + getNoOfMaterials() + ").\n" +

"Services: (" + getNoOfServices() + ").\n";

}

public int getNoOfMaterials()

{

int no = 0;

for (Entity entity: entityList)

{

if (entity.getCategory() == 0) no++;

}

return no;

}

public int getNoOfServices()

{

int no = 0;

for (Entity entity: entityList)

{

if (entity.getCategory() == 1) no++;

}

return no;

}

public String getName() {return name;}

public void setName(String name) {this.name = name;}

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

public Admin getAdmin() {return admin;}

public void addEntity(Entity entity)

{

try{

if(entityList.contains(entity))throw new AlreadyExistsException();

entityList.add(entity);

} catch (AlreadyExistsException e) {

e.notifyMe();

}

}

public void removeEntity(Entity entity)

{

entityList.remove(entity);

}

public int insertDonator(Donator possibleDonator)

{

try{

for(Donator donator : donatorList)

{

if (donator.getPhone().equals(possibleDonator.getPhone()))

throw new AlreadyExistsException();

}

donatorList.add(possibleDonator);

return 2;

} catch (AlreadyExistsException e) {

e.notifyMe();

return 0;

}

}

public void removeDonator(Donator donator)

{

donatorList.remove(donator);

}

public int insertBeneficiary(Beneficiary possibleBeneficiary)

{

try{

for(Beneficiary beneficiary : beneficiaryList)

{

if (beneficiary.getPhone().equals(possibleBeneficiary.getPhone()))

throw new AlreadyExistsException();

}

beneficiaryList.add(possibleBeneficiary);

return 2;

} catch (AlreadyExistsException e) {

e.notifyMe();

return 0;

}

}

public void removeBeneficiary(Beneficiary beneficiary)

{

beneficiaryList.remove(beneficiary);

}

public String listEntities()

{

String matstr = "-Materials-";

String serstr = "\n-Services-";

for (Entity entity: entityList)

{

if(entity.getCategory()==0) matstr += "\n" + entity.getName();

else serstr += "\n" + entity.getName();

}

return matstr + serstr;

}

public String listDonators()

{

String out = "-Donator (Phone)-";

int no = 1;

for (Donator donor : donatorList)

{

out += "\n" + no + ". " + donor.getName() + " (" + donor.getPhone() + ")";

no++;

}

return out;

}

public String listBeneficiaries()

{

String out = "-Beneficiary (Phone)-";

int no = 1;

for (Beneficiary benef : beneficiaryList)

{

out += "\n" + no + ". " + benef.getName() + " (" + benef.getPhone() + ")";

no++;

}

return out;

}

public int checkAccount(String phone)

{

ArrayList<String> phoneList = new ArrayList<String>();

for(Donator donator : donatorList)

{phoneList.add(donator.getPhone());}

for(Beneficiary beneficiary : beneficiaryList)

{phoneList.add(beneficiary.getPhone());}

phoneList.add(admin.getPhone());

if (phoneList.contains(phone)) return 2;

else return 1;

}

public String whatIsThisPhone(String phone)

{

for(Donator donator : donatorList)

{

if (phone.equals(donator.getPhone())) return "donator";

}

for(Beneficiary beneficiary : beneficiaryList)

{

if (phone.equals(beneficiary.getPhone())) return "beneficiary";

}

if (phone.equals(admin.getPhone())) return "admin";

return "error";

}

public Donator getDonatorFromPhone(String phone)

{

for(Donator donator : donatorList)

{

if (phone.equals(donator.getPhone())) return donator;

}

return new Donator("error", "bug");

}

public Beneficiary getBeneficiaryFromPhone(String phone)

{

for(Beneficiary beneficiary : beneficiaryList)

{

if (phone.equals(beneficiary.getPhone())) return beneficiary;

}

return new Beneficiary("error", "bug");

}

}

* Main

import java.util.\*;

public class Main

{

public static void main(String[] args)

{

Admin theAdmin = new Admin("Tim", "1234567890");

Organization org = new Organization("GoodOrg", theAdmin);

org.insertDonator(new Donator("Park", "3333333333"));

org.insertDonator(new Donator("Bark", "1231231231"));

org.insertBeneficiary(new Beneficiary("Mark", "1111111111"));

org.insertBeneficiary(new Beneficiary("Clark", "2222222222"));

org.insertBeneficiary(new Beneficiary("Dark", "3213213211"));

org.addEntity(new Material("Rice", "One package containing 1kg of rice.", 16, 0, 5, 20, 50));

org.addEntity(new Material("Milk", "A single 1L bottle of milk.", 14, 0, 3, 10, 20));

org.addEntity(new Material("Sugar", "One pack containing 1kg of sugar.", 12, 0, 2, 4, 10));

org.addEntity(new Service("Medical Support", "One 1-hour visit to healthcare facilities.", 10, 1));

org.addEntity(new Service("Nursery Support", "Provision of nursing services for 1 hour per request.", 8, 1));

org.addEntity(new Service("Babysitting", "One hour babysitting per request.", 6, 1));

Menu menu = new Menu();

menu.startMenu();

}

}

* AlreadyExistsException

public class AlreadyExistsException extends Exception

{

private String msg = "Already exists";

public AlreadyExistsException()

{

}

public AlreadyExistsException(String msg)

{

this.msg = msg;

}

public void notifyMe()

{

Menu.notification(msg, Menu.seconds);

}

}

* NotAcceptedValueException

public class NotAcceptedValueException extends Exception

{

private String msg = "Not accepted value";

public NotAcceptedValueException()

{

}

public NotAcceptedValueException(String msg)

{

this.msg = msg;

}

public void notifyMe()

{

Menu.notification(msg, Menu.seconds);

}

}

* RequestDonationException

public class RequestDonationException extends Exception

{

private String msg = "We were unable to register your request.\nPlease try again.";

public RequestDonationException()

{

}

public RequestDonationException(String msg)

{

this.msg = msg;

}

public void notifyMe()

{

Menu.notification(msg, Menu.seconds);

}

}

* InsufficientQuantityException

public class InsufficientQuantityException extends RequestDonationException

{

private String msg = "You ask too much!";

public InsufficientQuantityException()

{

super();

}

public InsufficientQuantityException(String msg)

{

super(msg);

}

}

* RequestDonationNotFoundException

public class RequestDonationNotFoundException extends RequestDonationException

{

public RequestDonationNotFoundException()

{

super();

}

public RequestDonationNotFoundException(String msg)

{

super(msg);

}

}

* InvalidRequestException

public class InvalidRequestException extends RequestDonationException

{

private String msg = "You dont deserve it!";

public InvalidRequestException()

{

super();

}

public InvalidRequestException(String msg)

{

super(msg);

}

}