**Elevator**

Για το task προσομοίωσης της λειτουργίας ενός ασανσέρ υλοποίησα τις ακόλουθες 3 κλάσεις: την κλάση ***Floor***, την κλάση ***Elevat or*** και την κλάση ***Building***.

**Floor**

Ένα αντικείμενο της κλάσης ***Floor*** αποτελείται από μια ουρά ατόμων *queue* και από ένα *ID* που υποδηλώνει τον αριθμό του ορόφου. Υπάρχουν 2 μέθοδοι: η *createFloorQueue(peopleArray)* όπου με βάση την είσοδο (*peopleArray)*, φτιάχνει για τον συγκεκριμένο όροφο μια ουρά με τα άτομα προς επιβίβαση στο ασανσέρ διατηρώντας την σειρά με την οποία αιτήθηκαν την είσοδο στο ασανσέρ. Επιπλέον, υπάρχει η μέθοδος *dequeue* που εκτελεί την ακόλουθη λειτουργία: για τον όροφο στον οποίο βρίσκεται το ασανσέρ, ελέγχει τα άτομα που περιμένουν στην ουρά. Εάν το ασανσέρ είναι γεμάτο, συνεχίζει προς την κατεύθυνση που πήγαινε, καθώς δεν μπορεί να δεχτεί άλλα άτομα. Διαφορετικά, ελέγχει την ουρά ώστε να βρει ποια άτομα θέλουν να πάνε προς την ίδια κατεύθυνση με το ασανσέρ και τα βάζει στο ασανσέρ μέχρις ότου γεμίσει ή αδειάσει η ουρά. Στο τέλος, κάνει sort τα άτομα ώστε να φαίνεται η έξοδος πιο σωστά με βάση την σειρά που θα κατέβουν. Για να διατηρηθεί η σειρά των ατόμων που περιμένουν στην ουρά, χρησιμοποιώ μια *temp\_queue.* Ο λόγος είναι πως πρέπει να ελέγξω αν το κάθε άτομο της ουράς θέλει να πάει σε όροφο με την ίδια κατεύθυνση που έχει το ασανσέρ. Για να γίνει ο έλεγχος, χρησιμοποιώ τη μέθοδο *get()*, πουκάνει ουσιαστικά *pop()*,με αποτέλεσμα τα άτομα που βγαίνουν από την ουρά για να ελεγχθούν, να χαλάνε την προτεραιότητα της ουράς, γιατί το άτομο μπορεί να μην πηγαίνει προς την ίδια κατεύθυνση ή το ασανσέρ να είναι γεμάτο. Τα άτομα αυτά προστίθενται με τη σωστή σειρά στην *temp\_queue* ενώ όσοι πληρούν τις προϋποθέσεις, ανεβαίνουν στο ασανσέρ. Στο τέλος θέτουμε την αρχική ουρά να είναι ίση με την *temp\_queue* για να επαναφέρουμε την σωστή σειρά.

**Elevator**

Ένα αντικείμενο της κλάσης ***Elevator*** αρχικοποιείται με τη μέγιστη χωρητικότητα (*maxCapacity*) και τον αριθμό των ορόφων που μπορεί να επισκεφθεί (*floors*). Στον κατασκευαστή ορίζεται η τρέχουσα χωρητικότητα (*currentCapacity)* ίση με την συνολική χωρητικότητα, το *currentFloor* (0 στην αρχή, δηλαδή ισόγειο), η κατεύθυνση (*direction*) που αρχικά είναι 1 και ένας άδειο array από άτομα (*people*). Κατασκεύασα την μέθοδο *addPersonToElevator*,η οποία παίρνει ως όρισμα ένα άτομο και το προσθεί στο people array του Elevator, μειώνοντας την τρέχουσα χωρητικότητα του, και την μέθοδο *removePeople* που αφαιρεί τα άτομα που θέλουνε να κατέβουνε στον *currentFloor*. Για να μην γίνει in-place η αφαίρεση από το array, δημιουργώ ένα shallow copy. Η διάσχιση των ατόμων γίνεται στα στοιχεία του copied array, αλλά οι αλλαγές γίνονται στο αρχικό array. Έτσι δεν δημιουργείται πρόβλημα με αλλαγές στα indexes του array και ελέγχονται όλα τα άτομα με σωστό τρόπο.

**Building**

Η κλάση ***Building*** αποτελείται από ένα array από *Floor queues* που αναπαριστά την αρχική κατάσταση στο κτίριο, και από ένα αντικείμενο της κλάσης *Elevator*. Η μέθοδος *checkAllFloors* εκτελείται συνεχώς, μέσα στην *basicElevatorFunction* που περιγράφει τη λειτουργία του ασανσέρ, και ελέγχει αν οι ουρές έχουν αδειάσει και αν το ασανσέρ είναι άδειο. Αν κάτι από τα παραπάνω δεν ισχύει, τότε επιστρέφει False, δηλαδή πως δεν έχει ολοκληρωθεί ακόμα η διαδικασία και το ασανσέρ συνεχίζει να εξυπηρετεί τα άτομα, ενώ αν ισχύουν και τα δύο, το ασανσέρ επιστρέφει στο ισόγειο και εμφανίζει το μήνυμα του τερματισμού. Η μέθοδος *basicElevatorFunction* όπως προείπα, με ένα while loop ελέγχει αν ισχύει η τερματική συνθήκη της *checkAllFloors*, διαφορετικά κάνει τα παρακάτω:

Αρχικά εμφανίζει τον όροφο στον οποίο βρίσκεται το ασανσέρ, την κατεύθυνση του ασανσέρ, τα άτομα μέσα στο ασανσέρ και την ουρά των ατόμων στον όροφο που βρίσκεται το ασανσέρ. Όταν ανεβαίνει, τα άτομα εμφανίζονται σε αύξουσα σειρά ενώ όταν κατεβαίνει σε φθίνουσα, όπως δηλαδή θα ήταν και σε ένα κανονικό ασανσέρ η σειρά που θα περιμένανε για να κατέβουν από αυτό. Έπειτα αποβιβάζει τα άτομα που θέλανε να πάνε στο *currentFloor*, καλώντας την *removePeople*, και στη συνέχεια επιβιβάζει τα άτομα που θέλουν να επιβιβαστούν και είναι «νόμιμο» να το κάνουν, μέσω της *floor.dequeue().* Έπειτα, ελέγχει την κατεύθυνση του ασανσέρ και τα άτομα που βρίσκοντα μέσα. Αν είχε κατεύθυνση προς τα πάνω και υπάρχει κάποιος που θέλει να πάει προς την ίδια κατεύθυνση, τότε αυξάνει το *elevator.currentFloor* και διατηρεί την κατεύθυνση. Για να βρει σε ποιον όροφο θα ανέβει, βρίσκει τον κοντινότερο υψηλότερο όροφο-στόχο από όσους είναι μέσα στο ασανσέρ και τον αποθηκεύει στη μεταβλητή *nextFloorFromInside.* Επίσης, βρίσκει από τους υψηλότερους ορόφους τον κοντινότερο υψηλότερο όροφο στον οποίον υπάρχει άτομο που θέλει να πάει προς την ίδια κατεύθυνση και αποθηκεύει το ID του ορόφου του ατόμου στη μεταβλητή *nextFloorFromOutside.* Στη συνέχεια πηγαίνει στον κοντινότερο από τα *nextFloorFromInside* και *nextFloorFromOutside*. Διαφορετικά, αν δεν υπάρχει κάποιος μέσα στο ασανσέρ που θέλει να πάει προς την ίδια κατεύθυνση, αλλά υπάρχουν άτομα σε πιο πάνω όροφο που έχουν καλέσει το ασανσέρ, τότε ανεβαίνει μέχρι τον υψηλότερο όροφο που υπάρχει κάποιο άτομο. Αν η κατεύθυνση ήταν προς τα κάτω, εκτελεί την αντίστοιχη λειτουργία. Στο τέλος, τυπώνει την κατάσταση του ασανσέρ αφού φύγει από τον όροφο και τον όροφο στον οποίο κατευθύνεται.

Η global μεταβλητή *PAUSE* χρησιμοποιείται για να δίνει μια πιο ρεαλιστική εμπειρία στον τρόπο λειτουργίας του ασανσέρ, αλλά για ταχύτερη εκτέλεση και testing μπορείτε να την θέσετε False.

**Main**

Για την είσοδο των δεδομένων έχω οργανώσει την main με τέτοιο τρόπο ώστε να μπορείτε να εισάγετε τα δεδομένα (capacity, floors, queues) ένα προς ένα μέσω terminal ή, για να κάνετε πιο γρήγορα τις δοκιμές, απευθείας μέσα από τη main και το *peopleArray*. Οι έλεγχοι για τα δεδομένα εισόδου πραγματοποιούνται μόνο με τον πρώτο τρόπο. Επιπλέον, δεν υλοποίησα την λειτουργία με τα κουμπιά, γιατί αν δεν κάνω λάθος, το γεγονός πως κάθε άνθρωπος αναπαρίσταται με τον αριθμό του ορόφου που θέλει να πάει, τότε κατά μία έννοια αποτελεί ο ίδιος το κουμπί που δείχνει την κατεύθυνση. Δηλαδή ο έλεγχος στην *dequeue: person – currentFloor* πρακτικά λειτουργεί ως κουμπί, δείχνοντας με το πρόσημο της αφαίρεσης αυτής αν θέλει να πάει προς τα πάνω ή προς τα κάτω.