Implementation Description

Περιγραφή υλοποίησης

Η εφαρμογή χρησιμοποιεί δύο νήματα, ένα για την εκτέλεση της προσομοίωσης, το οποίο χρησιμοποιεί σε την συνάρτηση Stopwatch εντός ενός while loop στη main, προκειμένου να υλοποιούνται οι απαιτούμενες εργασίες ανά 5 δευτερόλεπτα, και ένα δεύτερο νήμα για τον έλεγχο της γραφικής διεπαφής του χρήστη.

Για την προσομοίωση έχουν δημιουργηθεί οι παρακάτω κλάσεις με βάση τις προδιαγραφές που δόθηκαν στην εκφώνηση.

- Για την υλοποίηση των χώρων στάθμευσης δημιουργήθηκε η κλάση parking.
- Για την υλοποίηση των πτήσεων δημιουργήθηκε η κλάση Flight.
- Για την υλοποίηση των αεροδρομίων δημιουργήθηκε η κλάση Airport.
- Για την υλοποίηση των κατηγόριων των χώρων στάθμευσης, δημιουργήθηκε η κλάση Category, ενώ με χρήση κληρονομικότητας δημιουργήθηκαν κλάσεις για τη κάθε διαφορετική κατηγορία.
- Για τα είδη Πτήσεων, δημιουργήθηκε το Enumeration flight_type,το οποίο περιέχει τις τιμές COMMERCIAL, PASSENGER και PRIVATE για πτήσεις εμπορευματικές, επιβατικές και ιδιωτικές αντίστοιχα.
- Με όμοιο τρόπο δημιουργήθηκαν αντίστοιχα Enumerations για τον τύπο αεροσκάφους (airship_type), για τις διάφορες υπηρεσίες (special_services).
- Για τον έλεγχο της κατάστασης της κάθε πτήσης δημιουργήθηκε ένα ακόμη Enumeration, το flight_state.
- Επιπλέον δημιουργήθηκε ένα Enumeration για τις καταστάσεις της κάθε θέσης στάθμευσης, parking_state.
- Η λογική της προσομοίωσης έχει συγκεντρωθεί στην κλάση simulation, η οποία χρησιμοποίει οντότητες από τις προαναφερθέντες κλάσεις. Για να διασφαλιστεί η μοναδικότητα της οντότητας του simulation, καθώς και να γίνεται προσβάσιμο με ευκολία από τις κλάσεις και τα τμήματα της γραφικής διεπαφής χρήστη, χρησιμοποιήθηκε Singleton Pattern για την δόμηση του Constructor αυτής της κλάσης.

Για την γραφική διεπάφη χρήστη (GUI) έχουν δημιουργηθεί τα παρακάτω τμήματα.

- Για τον έλεγχο του κεντρικού παραθύρου και των επιμέρους τμημάτων του έχει δημιουργηθεί η κλάση GuiController, με την οποία γίνεται ο έλεγχος του GUI.fxml (αρχείο περιγραφής του κεντρικού παραθύρου της εφαρμογής).
- Για τον έλεγχο του αναδυόμενου παραθύρου για την φόρτωση διαφορετικού σεναρίου, χρησιμοποιείτε η κλάση LoadPopUpController, με την οποία γίνεται ο έλεγχος του Load_scenario.fxml.
- Για τα διάφορα αναδυόμενα παράθυρα που παρέχουν πληροφορίες για τις πτήσεις(Gates, Flights, Delayed, Holding, Next Departures) γίνεται χρήση της κλάσης GatesController, με την οποία ελέγχεται το αρχείο Gates.fxml

Απαιτήσεις

- Για το χρονόμετρο της προσομοίωσης και την τήρηση της σύμβασης ως προς αποψή του χρόνου, έχει δημιουργήθει η συνάρτηση stopwatch, ως public Boolean συνάρτηση της κλάσης simulation. Η συνάρτηση αυτή καλείτε εντός μιας δομής while, από την Main Class μέχρι να τερματιστεί το πρόγραμμα.
- Αρχικοποίηση: Αρχικά μέσω της Main Class γίνεται η δημιουργία του simulation instance μέσω του public constructor του simulation. Μέσα στον Public Constructor του simulation, αρχικοποιούνται οι τιμές των βασικών μεταβλητών της προσομοίωσης . Επίσης οι λίστες για την περιγραφή των πτήσεων και των χώρων στάθμευσης, αρχικοποιούνται με βάση τα αρχεία της τρέχουσας προσομοίωσης. Δημιουργείτε μια οντότητα για το τρέχων αεροδρόμιο και μέσω χρήσης των συναρτήσεων **setup(int)** εισάγονται σε κατάσταση Hold όλες οι πτήσεις που είναι απαραίτητες σε χρόνο 00:00. Στη συνέχεια καλείτε η συνάρτηση **initialiseParking()** η οποία αναθέτει σε κάθε πτήση από κάποια θέση στάθμευσης εφόσον γίνεται(όσες πτήσεις δεν ανατεθούν σε χώρο στάθμευσης, απλά μένουν στην λίστα με τις πτήσεις που βρίσκονται σε αναμονή). Στη συνέχεια καλείτε η συνάρτηση **setInitialState()** η οποία για όσες πτήσεις έχουν ανατεθεί σε κάποια θέση στάθμευσης, τις αφαιρεί από την λίστα πτήσεων σε αναμονή, και την προσθέτει στη λίστα με τις σταθμευμένες πτήσεις, επίσης θέτει την κατάσταση των πτήσεων σε PARKED.
- Αίτημα προσγείωσης και εξυπηρέτησης: Κάθε πτήση που προστίθεται από την γραφική διεπάφη, προστίθεται στη ουρά fOnHold. Κάθε λεπτό εντός των πλαισίων της προσομοίωσης, ελέγχονται όλες οι πτήσης σε αναμονή με σειρά προτεραιότητας First come First Served, αν είναι δυνατόν να εξυπηρετηθούν σε κάποια θέση στάθμευσης μέσω της συνάρτησης checkForLanding(). Στις θέσεις στάθμευσης αλλάζει η κατάσταση σε OCCUPIED ενώ οι πτήσεις μεταφέρονται σε κατάσταση Landing.
  Οι πτήσεις που είναι σε κατάσταση Landing είναι αποθηκευμένες σε μία λίστα fLanding. Όσο βρίσκεται σε αυτή τη κατάσταση, προσμετράτε ο χρόνος που βρίσκεται η κάθε πτήση σε αυτήν. Όταν για το δεδομένο είδος πτήσης έχει περάσει ο κατάλληλος χρόνος χρησιμοποιείτε η διαδικασία parkAfterLand() για την αλλαγή της πτήσης σε κατάσταση PARKED και την προσθήκη της στη λίστα fParked, μαζί με όλες τις άλλες σταθμευμένες πτήσεις. Αυτός ο ελέγχος γίνεται με την συνάρτηση **landingTransition().**
- Επιλογή χώρου στάθμευσης και εξυπηρέτησης: Όταν πρέπει να γίνει έλεγχος για την επιλογή χώρου στάθμευσης για μια πτήση που βρίσκεται σε αναμονή, χρησιμοποιείτε η **public Boolean checkForFlight(Flight)** η οποία παίρνοντας ως είσοδο μία πτήση, ελέγχει αν για τα δεδομένα της οντότητας της κλάσης είναι δυνατό να προσγειωθεί εκεί η πτήση. Κάθε πτήση ελέγχεται με σειρά προτεραιότητας στους χώρους στάθμευσης, αρχίζοντας από τους χώρους σταθμεύσεις οι οποίοι έχουν αρχικοποιηθεί πρώτοι, κοινώς από αυτούς που έχουν μικρότερο χρονικό όριο. Η κάθε πτήση ανατίθεται στην πρώτη θέση που είναι δυνατό να γίνει στάθμευση.
- Παραμονή και αναχώρηση: Όλες οι πτήσεις που είναι σε κατάσταση PARKED είναι ταυτόχρονα και στη λίστα fParked. Με κάθε τικ της προσομοίωσης, ελέγχουμε αν κάποια από τις πτήσεις αυτές αναχωρούν με τη συνάρτηση **updateDeparts().**
    - Το αν μια πτήση σε κατάσταση PARKED αναχωρεί ελέγχεται τυχαία με αυξανόμενη πιθανότητα αναχωρήσεις όσο πλησιάζει στον προγραμματισμένο χρόνο αναχώρησης. Η πιθανότητα αυτή αυξάνεται όσο καθυστερεί

περισσότερο πέρα από αυτόν το χρόνο. Ο έλεγχος γίνεται μέσω της συνάρτησης **calculateDepart(int)** και τη συνάρτηση **determineSuccess(int)**
Με την **determineSuccess(int)** δίνεται ένας ακέραιος ενώ το πρόγραμμα επιλέγει έναν αριθμό τυχαία στο σύνολο 0 έως 10000. Αν ο ακέραιος που δοθεί είναι μεγαλύτερος από τον τυχαίο αριθμό, τότε επιστρέφεται true, αλλιώς false.
Με την **calculateDeparts** δίνεται ως είσοδος η διάφορα του προγραμματισμένου χρόνου με τον σύγχρονο χρόνο της προσομοίωσης, ανάλογα με αυτόν, καλείτε η συνάρτηση **determineSuccess(int)** με διαφορετική είσοδο, η οποία αυξάνεται όσο η χρονική διάφορα μειώνεται.
    - Εφόσον μια πτήση κριθεί ότι πρέπει να αναχωρήσει γίνονται οι κατάλληλοι υπολογισμοί για την εύρεση του κόστους παραμονής και στο τέλος διαγράφεται η πτήση από το σύστημα.
- Η σειρά τον δραστηριοτήτων της προσομοίωσης δίνεται από την συνάρτηση **private void update()**, η οποία καλείτε κάθε 1 λεπτό (εντος των πλαισίων της προσομοίωσης)
    - Πρώτα αυξάνεται ο χρόνος.
    - Έπειτα ελέγχεται αν κάποια πτήση αλλάζει από κατάσταση Landing σε Parked.
    - Έπειτα για κάθε πτήση στο σύστημα μειώνεται ο σχετικός χρόνος προς τον προγραμματισμένο χρόνο αναχώρησης.
    - Στη συνέχεια γίνεται έλεγχος για αναχώρηση πτήσεων.
    - Στη συνέχεια γίνεται έλεγχος για νέες αναθέσεις πτήσεων σε θέσεις στάθμευσης.

Δημιουργία γραφικής διεπάφη

Πραγματοποιήθηκε κάθε τμήμα που αναφερόταν στην εκφώνηση, ενώ ταυτόχρονα έγινε προσθήκη της δυνατότητας αλλαγής της ταχύτητας της προσομοίωσης, μέσω ενός menu “Settings”.

Λοιπές απαιτήσεις

Οι αρχές σχεδίασης του αντικειμενοστραφούς προγραμματισμού, έχουν τηρηθεί.

Έχει γίνει τεκμηρίωση των μεθόδων των κλάσεων simulation, Airport, Flight με τις προδιαγραφές Javadoc .