

# ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &  
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ



Τεχνολογία πολυμέσων Project

Ον. Επ. : Δημήτριος Αμπελικιώτης  
Α.Μ. : 03114198

## 1) Εισαγωγή

Η εφαρμογή περιγράφει την λειτουργία ενός χειριστή πτήσεων σε ένα αεροδρόμιο. Αποτελείται από δύο βασικά μέρη, το backend και το frontend. Το πρώτο είναι υπεύθυνο για όλο το λειτουργικό κομμάτι της εφαρμογής και το δεύτερο είναι κατα κύριο αρμόδιο για το interface και την αλληλεπίδρασή της με τον χρήστη.

## 2) Frontend

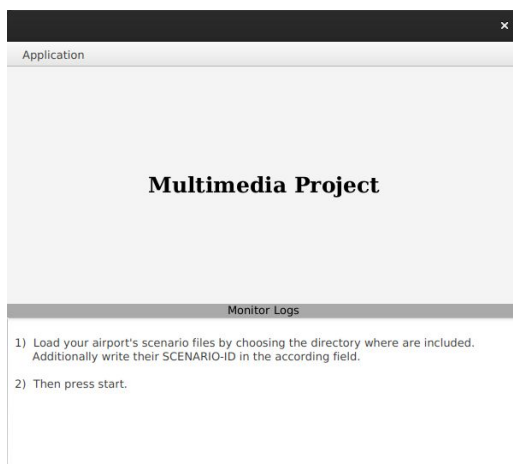
Ξεκινάμε με την περιγραφή του frontend αφού είναι το πρώτο που ξεκινάει και στην συνέχεια φορτώνει το backend για την εκκίνηση της εφαρμογής.

Με την εκκίνηση της εφαρμογής καλείται η συνάρτηση main() (στην κλάση Main) και στην συνέχεια η συνάρτηση start(Stage stage) και τελικά η beforeStart(). Η τελευταία είναι υπεύθυνη για να δημιουργήσει την αρχική μας σκηνή η οποία δίνει τις δυνατότητες στον χρήστη να φορτώσει τα αρχεία σεναρίων που θέλει να χρησιμοποιήσει, να αρχίσει την εφαρμογή και να την κλείσει.

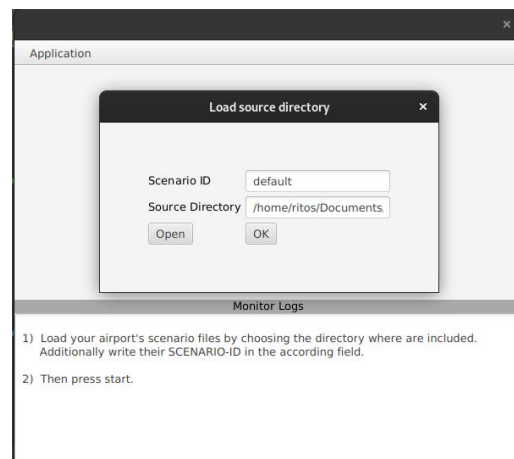
Εκτός από το μενού επιλογών, στο κάτω μέρος της οθόνης παρατηρούμε και ένα output console που είναι υπεύθυνο για την ενημέρωση του χρήστη για την πρόοδο του συστήματος σε ανταπόκριση με τις επιλογές του.

Παρακάτω βλέπουμε μία μικρή παρουσία του αρχικού παραθύρου:

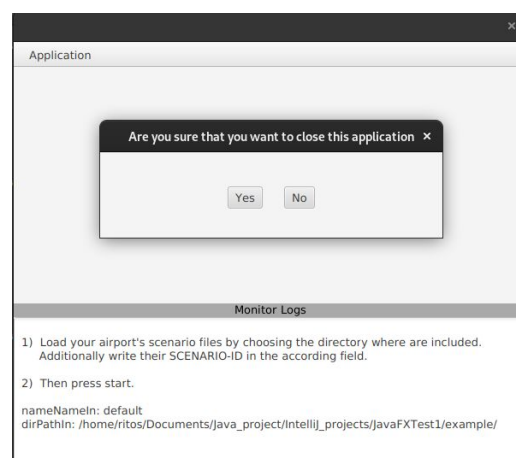
### 1) Αρχικό παράθυρο



### 2) Επιλογή Φόρτωσης αρχείου



### 3) Επιλογή εξόδου



Μόλις πατήσουμε το start τότε το frontend ελέγχει αν έχει προηγηθεί το στάδιο της φόρτωσης των αρχείων και αν εκείνα δεν είναι κενά πεδία. Αν οι παραπάνω έλεγχοι γίνουν με επιτυχία τότε καλείται ο constructor της κλάσης Airport με παραμέτρους το filepath και το scenario name. Η κατασκευή του παραπάνω αντικειμένου είναι μέσα σε try catch. Αυτό έχει νόημα, αφού μας ελαφρύνει από την διεξαγωγή διεξοδικών ελέγχων για την ύπαρξη των αρχείων ή την ορθότητα τους. Έτσι, αν το backend μας επιστρέψει exception μπορούμε να ενημερώσουμε τον χρήστη για τα παραπάνω προβλήματα αντίστοιχα και τον αφήνουμε να φορτώσει κάποιο άλλο σενάριο.

Αν η κλήση της κατασκευής του αεροδρομίου επιστρέψει χωρίς να προκληθεί κάποιο exception τότε καλείται η δεύτερη σκηνή της εφαρμογής, η οποία δίνει την δυνατότητα στον χρήστη εισαγωγής μίας αίτησης πτήσης και μία συνολική εποπτεία για την λειτουργία.

Παρακάτω βλέπουμε μία μικρή παρουσία της νέας σκηνής:

The screenshot shows a window titled 'Application' with a 'Details' tab. It contains a table with 'Question' and 'Info' columns. Below the table is a form with fields for 'Flight ID', 'Flight Type', 'Plane Type', 'Parking Time', 'City', and 'Required services'. There are 'Submit' and 'Clear' buttons. At the bottom, there is a 'Monitor Logs' section with a list of log messages.

Question	Info
No. of arrived flights:	0
No. of Available Parking Spaces:	34
No. of flights will be departed into the next 10 min:	0
Total amount of money earned:	6800.0
Time since Application has started:	0 days, 2 hours, 40 minutes, 0 seconds

Flight ID: dc14c173  
Flight Type: Cargo  
Plane Type: SingleMotor  
Parking Time: 120  
City: Chicago  
Required services: [dropdown]  
[Submit] [Clear]

Monitor Logs  
Set parameters for time interrupt  
Activate time interrupt  
Your request for flight 14 is being handled  
The flight 14 is parked  
The flight b2 is parked  
The flight 14 is leaving  
The flight b2 is leaving

The screenshot shows a window titled 'Application' with a 'Details' tab. It contains a table with 'Question' and 'Info' columns. Below the table is a form with fields for 'Flight ID', 'Flight Type', 'Plane Type', 'Parking Time', 'City', and 'Required services'. There are 'Submit' and 'Clear' buttons. At the bottom, there is a 'Monitor Logs' section with a list of log messages. A 'Parking Spaces' table is overlaid on the form.

ID	Category	Status	Flight ID	Leaving Time
G0	Gate	Available		
G1	Gate	Available		
G2	Gate	Available		
G3	Gate	Available		
G4	Gate	Available		
G5	Gate	Available		
G6	Gate	Available		
G7	Gate	Available		
G8	Gate	Available		
G9	Gate	Available		
C0	CargoGate	Available		
C1	CargoGate	Available		
C2	CargoGate	Available		
C3	CargoGate	Available		
K0	ZoneA	Available		

Πριν μπούμε σε έναν πιο λεπτομερειακό τρόπο περιγραφής του interface και το πως το κάθε module ανανεώνεται ας κάνουμε μία διακοπή να περιγράψουμε τον βασικό τρόπο λειτουργίας του backend.

### 3) Backend

Η λειτουργία του back αρχίζει με την κλήση του constructor της κλάσης Airport. Τότε ο παραπάνω constructor διαβάζει το αρχείο για την κατασκευή των χώρων στάθμευσης και δημιουργεί τα αντίστοιχα αντικείμενα ParkingSpace και τα προσθέτει στην λίστα availableSpaces.

Στην συνέχεια, καλεί τον constructor του scheduler, οποίος είναι υπεύθυνος για την διαχείριση του αεροδρομίου και γενικότερα της λειτουργικότητας ολοκληρης της εφαρμογής. Δύο βασικά βήματα που εκτελεί ο scheduler εκτός από διάφορες αρχικοποιήσεις, είναι η ανάγνωση του αρχείου πτήσεων, η κατασκευή των αντίστοιχων αντικειμένων και η προσάρτησή τους στην λίστα flights και η αρχικοποίηση του timer. Ο τελευταίος είναι κύριος παράγοντας στην αυτόματη λειτουργία της εφαρμογής, αφού μετά την ρύθμιση του προκαλεί interrupt ανά τακτά χρονικά

διαστήματα και ο έλεγχος περνάει στον handler “timeInterruptHandler()” του scheduler που δρομολογεί του ελέγχους και την εξυπηρέτηση πτήσεων.

Αξίζει να σχολιάσουμε πως μόνο αν όλα τα παραπάνω πετύχουν χωρίς κάποιο λάθος θα δημιουργηθεί η τελική σκηνή. Στην ουσία, όποιο error – exception προκύψει από την κατασκευή τον παραπάνω προωθείται στο frontend και τυπώνεται το ανάλογο μήνυμα για λαθοι στα αρχεία σεναρίων. Με τον παραπάνω τρόπο εκτελείται ο έλεγχος δεδομένων.

Αναλυτικότερα, ο timeInterruptHandler() (καλείται ανά 1 sec) καλεί με την σειρά του τρεις συναρτήσεις. Η πρώτη, αποτελεί η updateTime() η αυξάνει την μεταβλητή Timestamp app\_time κατά 5 δευτερόλεπτα (όπως μας ζητείται από την εκφώνηση) και εκείνη αποτελεί το “ρολόι” της εφαρμογής. Η δεύτερη και καρδιά όλων είναι η checkFlights(), η οποία ελέγχει αν μία εισερχόμενη πτήση μπορεί να εξυπηρετηθεί από το αεροδρόμιο, προσθέτει αποδεκτές πτήσης στην ουρά αναμονής, της εναποθέτει σε χώρους στάθμευσης και αλλάζει κατάσταση στους δύο ανάλογα με τον χρόνο και της προδιαγραφές τους.

Η βασική λειτουργικότητα του backend διαδραματίζεται στα παραπάνω, στην συνέχεια θα δούμε αναφορικά τις υπόλοιπες κλάσεις που χρησιμοποιούνται ως βασικά συστατικά στοιχεία.

Η κλάση ParkingSpace είναι εκείνη που αντιπροσωπεύει τους χώρους στάθμευσης. Μέσα της κρύβει όλα τα βασικά operation μέσω των οποίων μπορούμε να αλληλεπιδράσουμε με τα αντικείμενα που γεννιούνται από εκείνη. Στην πραγματικότητα, συγκεντρώνει όλες τις μεθόδους που πρέπει να έχει ένας χώρος στάθμευσης και κατα κύριο λόγο χρησιμοποιείται με την κληρονομία της από πιο εξειδικευμένους χώρους στάθμευσης όπως Gate, ZoneA κλπ. όπου γίνεται και η παραμετροποίηση του καθενός.

Με την παραπάνω προσέγγιση έχουμε δύο πολύ σημαντικά οφέλη. Αποφεύγουμε την επανάληψη μεγάλων μπλοκ κώδικα για κάθε διαφορετικά κλάση και κάνουμε το πρόγραμμά μας επεκτάσιμο από την στιγμή που κάθε κλάση χώρου στάθμευσης εκτός από τις μεθόδους που κληρονομεί μπορεί να επεκταθεί με τις δικές τις μοναδικές λειτουργίες. Τέλος, η πρόσθεση ενός νέου χώρου στάθμευσης μίας νέας κατηγορίας γίνεται πολύ εύκολα με την δημιουργία νέας κλάσεις με το αντίστοιχο template και η προσθήκη του ονόματός της στο dictionary του scheduler, το οποίο μας αφήνει να κάνουμε δομικές αλλαγές στο πρόγραμμά μας με την μικρότερη δυνατή γνώση της λειτουργικότητας του project.

Άλλες κλάσεις αποτελούν οι Flight και Logs. Η πρώτη αποτελεί την υλοποίηση μίας πτήσης στην εφαρμογή μας με της ζητούμενες λειτουργίες και μεθόδους. Η δεύτερη, υλοποιεί τα διαφορετικά μηνύματα που εμφανίζονται στο Πίνακα ενημερώσεων και ενημερώνεται τακτά από τον scheduler.

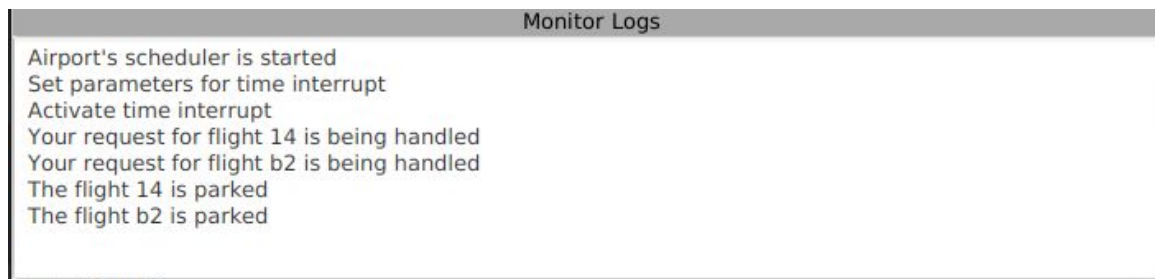
Τέλος, πρέπει να αναφερθεί πως οι παραπάνω κλάσεις ParkingSpace και Scheduler κληρονομούν την κλάση Observable. Με αυτό τον τρόπο έχουμε την δυνατότητα να παρατηρούμε αλλαγές και να ανανεώνουμε τα δεδομένα μας στο interface. Περαιτέρω ανάλυση του backend θα γίνεται σποραδικά παρακάτω μέσα στην ανάλυση των δομικών στοιχείων του frontend όπου κρίνεται κρίσιμο.

#### 4) Συστατικά στοιχεία του Frontend

Παρακάτω περιγράφεται αναφορικά ο τρόπος λειτουργία των τεσσάρων βασικών συστατικών του interface της εφαρμογής.

##### I) MonitorLogs

Στην παραπάνω κλάση δημιουργείται ένα TextArea αντικείμενο που με την βοήθεια που περνάει ως παράμετρος σε ένα αντικείμενο console (της κλάσης Console) και αυτό με την σειρά του σε ένα αντικείμενο ps (της κλάσης PrintStream) που μας δίνει την δυνατότητα να τυπώνουμε το stdout και stderr μέσα στην console log area που βλέπουμε παρακάτω.



##### II) ParkingSpaceOut

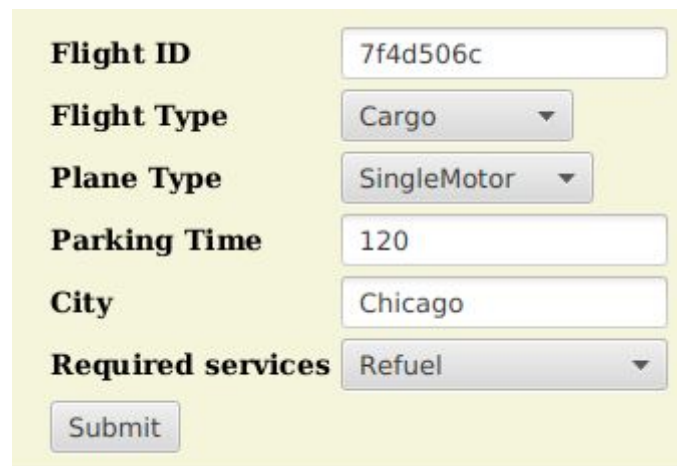
Το αντικείμενο αυτής της κλάσης είναι στην ουσία ένα GripPane αντικείμενο της JavaFX που μας δίνει την δυνατότητα να οπτικοποιήσουμε την κατάσταση των χώρων στάθμευσης. Αξίζει να επισημανθεί πως με την κληρονόμηση του interface observer μπορούμε να παρακολουθούμε τα αντικείμενα της κλάσης Parking Space και να ενημερώνουμε των πίνακα με νέα δεδομένα όπως με κόκκινο χρώμα αν είναι κατειλημμένα και των αριθμό πτήσης των οποίων φιλοξενούν. Ακόμα, μας δίνεται η δυνατότητα μεγέθυνσης σκρολάροντας μέσα στην περιοχή και μετακίνησης με σύρσιμο της εικόνα. Παρακάτω παρουσιάζονται με εικόνες τα προαναφερθέντα.



### III) AddFlightWin

Όπως και προηγουμένως, το αντικείμενο της κλάσης AddFlightWin είναι ένα GridPane που μας δίνει την δυνατότητα να προσθέσουμε μία πτήση. Όλες οι επιλογές είναι σε Check boxes ή multiple choice ώστε να αποφύγουμε την περίπτωση εισαγωγής λανθασμένων δεδομένων και τον εκτενή έλεγχο τους στο backend αλλά και ταυτόχρονα την γρήγορη και εύκολη χρήση της πλατφόρμας μας χωρίς να απαιτείται η γνώση ακριβείς σύνταξης των ειδών πτήσης και αεροπλάνου. Συμπληρωματικά αξίζει να αναφερθεί πως αν ο συνδυασμών που θα επιλέξει ο χρήστης δεν μπορεί να υποστηριχθεί από το αεροδρόμιό μας (π.χ. πολύ μεγάλος χρόνος παραμονής) σε αυτή την περίπτωση το backend ενημερώνει μέσω του console log παραθύρου πως αυτή η αίτηση πτήσης είναι αδύνατων να εκπληρωθεί. Ως σύμβαση έχει γίνει πως η πτήση προς καταχώρηση πρέπει να έχει οπωσδήποτε μία αιτούμενη υπηρεσία. Τέλος, τα πεδία έχουν ήδη προεπιλεγμένες τιμές και το id της πτήσης ανανεώνεται τυχαία μετά από κάθε καταχώρηση, ο λόγος αυτού είναι περισσότερο για ευκολία στο testing κομμάτι της εφαρμογής αφού μπορούμε να κάνουμε μαζικά πολλές αιτήσεις για να δούμε την απόκριση του συστήματος.

Παρουσίαση των παραπάνω:



The screenshot shows a form titled 'AddFlightWin' with the following fields and values:

- Flight ID: 7f4d506c
- Flight Type: Cargo
- Plane Type: SingleMotor
- Parking Time: 120
- City: Chicago
- Required services: Refuel

A 'Submit' button is located at the bottom left of the form.

### IV) ImportantInfoWin

Το παράθυρο στο πάνω μέρος της οθόνης εμφανίζει συγκεντρωτικές πληροφορίες για την κατάσταση της εφαρμογής που αφορούν την ώρα, τις πτήσεις και της θέσεις στάθμευσης. Ουσιαστικά σε κάθε αλλαγή του ρολογιού ο scheduler όπως περιγράφηκε παραπάνω ανανεώνει το ρολόι τα logs και την εσωτερική κατάσταση της εφαρμογής. Έτσι, με την αλλαγή του ρολογιού το αντικείμενο της παρούσας κλάσης ως observer του χρόνου ανανεώνεται με τα νέα logs και τα αναπαριστά σε έναν πίνακα της κλάσης TableView.

Παρακάτω βλέπουμε την γραφική απεικόνιση:

Question	Info
No. of arrived flights:	1
No. of Available Parking Spaces:	33
No. of flights will be departed into the next 10 min:	0
Total amount of money earned:	5000.0
Time since Application has started:	0 days, 1 hours, 35 minutes, 0 seconds



## 5) Details Menu

Στο Detail Menu υλοποιούνται η επιλογές της οπτικής απεικόνισης πληροφοριών σχετικά με τους χώρους στάθμευσης, των πτήσεων σε μορφή πίνακα. Τα δεδομένα όπως και η προηγούμενη κλάση ενημερώνονται σε κάθε χτύπο του ρολογιού, όλες οι κλάσεις υλοποιούν το interface του observer, καθώς παρακολουθούν τις αλλαγές της μεταβλητής του ρολογιού του scheduler. Παρακάτω επισυνάπτονται τα κατά την λειτουργία του προγράμματος.

The screenshot displays the IntelliJ IDEA interface with four data tables open:

- Parking Spaces' Table**
- Delayed Flights' Table**
- Next Departures' Table**
- Flights' Table**

ID	Category	Status	Flight ID	Leaving Time
K4	ZoneA	Available		
L0	ZoneB	Available		
L1	ZoneB	Available		
L2	ZoneB	Available		
L3	ZoneB	Available		
L4	ZoneB	Available		
M0	ZoneC	Unavailable	9257c576	2020-03-14 23:06:58.331
M1	ZoneC	Unavailable	85642e62	2020-03-14 23:35:58.331

Parking S...	ID	Flight Type	Plane Type	Leaving Time
N0	d952e60f	Cargo	SingleMotor	2020-03-14 22:45:58.331
N1	9b92fe02	Cargo	SingleMotor	2020-03-14 22:53:58.331
Z3	ce177051	Cargo	SingleMotor	2020-03-14 22:22:58.331

ID	Flight Type	Plane Type
ce177051	Cargo	SingleMotor

ID	City	Flight Type	Plane Type	Status	Parking Space ID	Leaving Time
d952e60f	Chicago	Cargo	SingleMotor	Parked	N0	2020-03-14 22:45:58.331
9b92fe02	Chicago	Cargo	SingleMotor	Parked	N1	2020-03-14 22:53:58.331
ce177051	Chicago	Cargo	SingleMotor	Parked	Z3	2020-03-14 22:22:58.331
28838309	Chicago	Cargo	SingleMotor	Parked	Z1	2020-03-15 00:00:58.331
ab0129a4	Chicago	Cargo	SingleMotor	Parked	Z2	2020-03-15 02:10:58.331
02d1f80f	Chicago	Cargo	SingleMotor	Landing	Z0	2020-03-15 02:13:58.331
45873976	Chicago	Cargo	SingleMotor	Holding		
85642e62	Chicago	Cargo	SingleMotor	Parked	M1	2020-03-14 23:35:58.331
076d2bd6	Chicago	Cargo	SingleMotor	Holding		
9257c576	Chicago	Cargo	SingleMotor	Parked	M0	2020-03-14 23:06:58.331
9362af18	Chicago	Cargo	SingleMotor	Parked	M2	2020-03-14 22:42:58.331
d5a636be	Chicago	Cargo	SingleMotor	Parked	M3	2020-03-15 00:22:58.331

ID	Flight Type	Plane Type	Time of Landing Request
45873976	Cargo	SingleMotor	2020-03-14 19:09:58.331
076d2bd6	Cargo	SingleMotor	2020-03-14 19:14:58.331

## 6) Γενικές πληροφορίες

Δίνεται η δυνατότητα στον χρήστη κατά την λειτουργία της εφαρμογής να φορτώσει ένα νέο αρχείο και πατώντας την επιλογή start να την επανακινήσει με νέο σενάριο ή και με το ίδιο.

Ακόμα στις κλάσεις Airport και Scheduler έχει προστεθεί η περιγραφή των constructors και των public μεθόδων με σχολιασμό που υποδεικνύεται κατά το JavaDoc πρότυπο.

Περαιτέρω σχολιασμός και ανάλυση της λειτουργίας μπορεί να εντοπιστεί στα σχόλια και στο source code του προγράμματος.