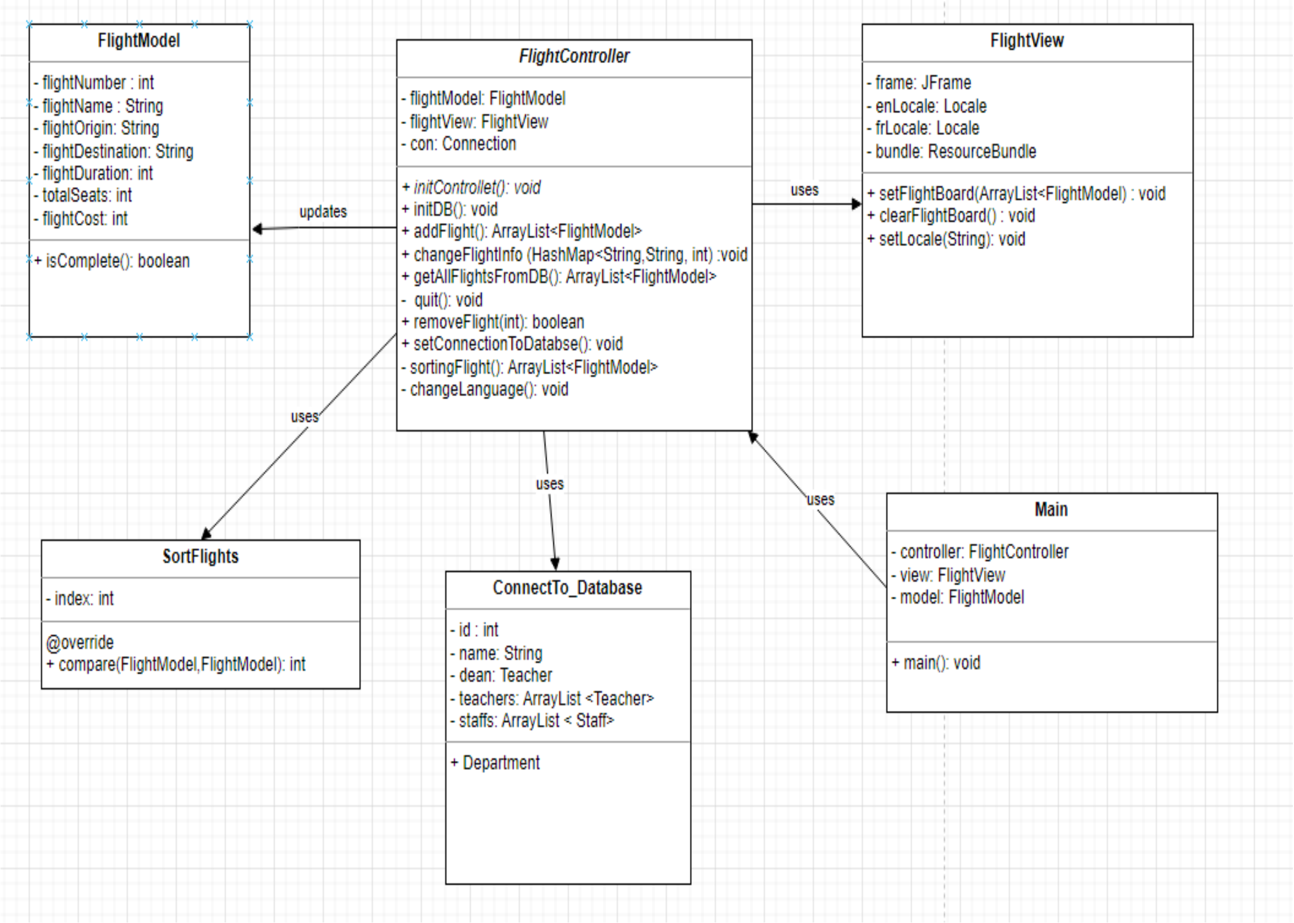
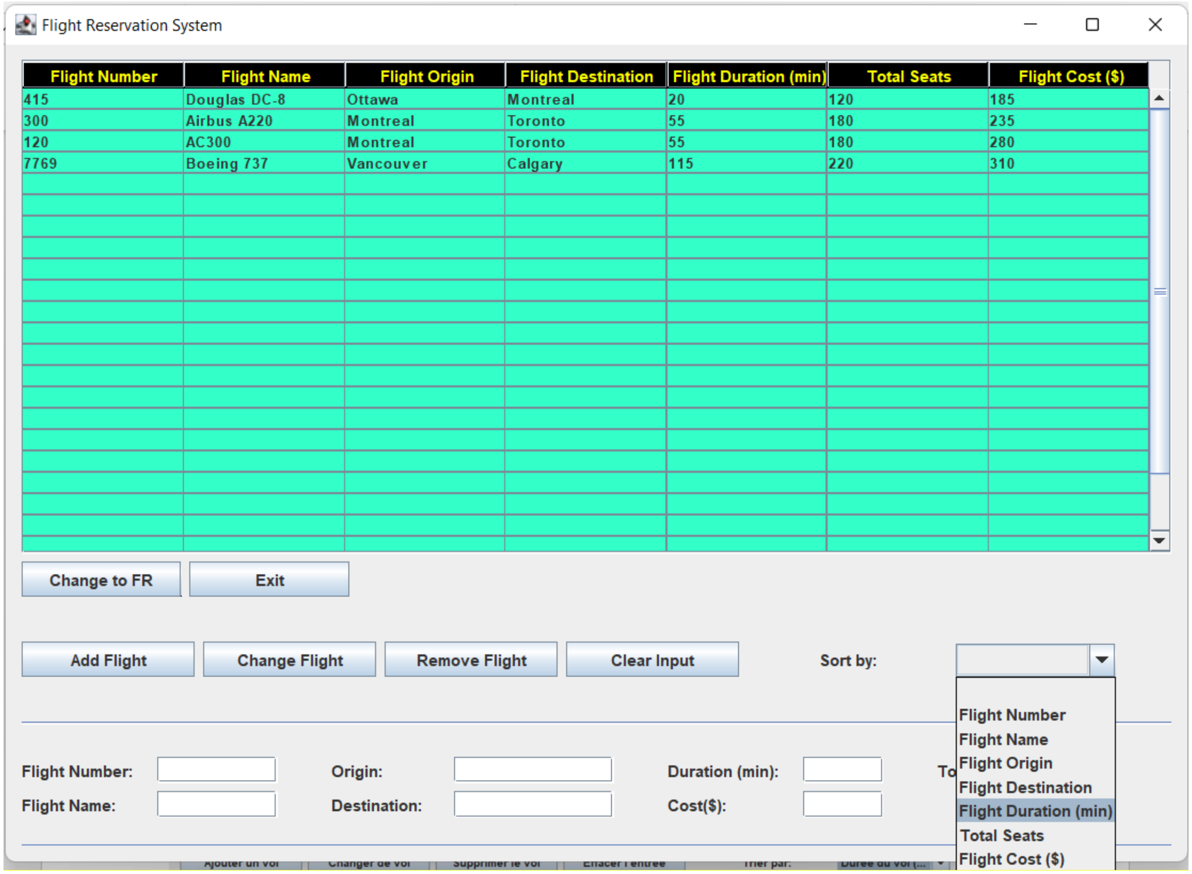
This document provides, a view of certain important design and implementation of the project. SQLite was used as the database (sqlite-jdbc-3.39.3.0.jar).

In this project, the MVC architecture is used as seen in the UML

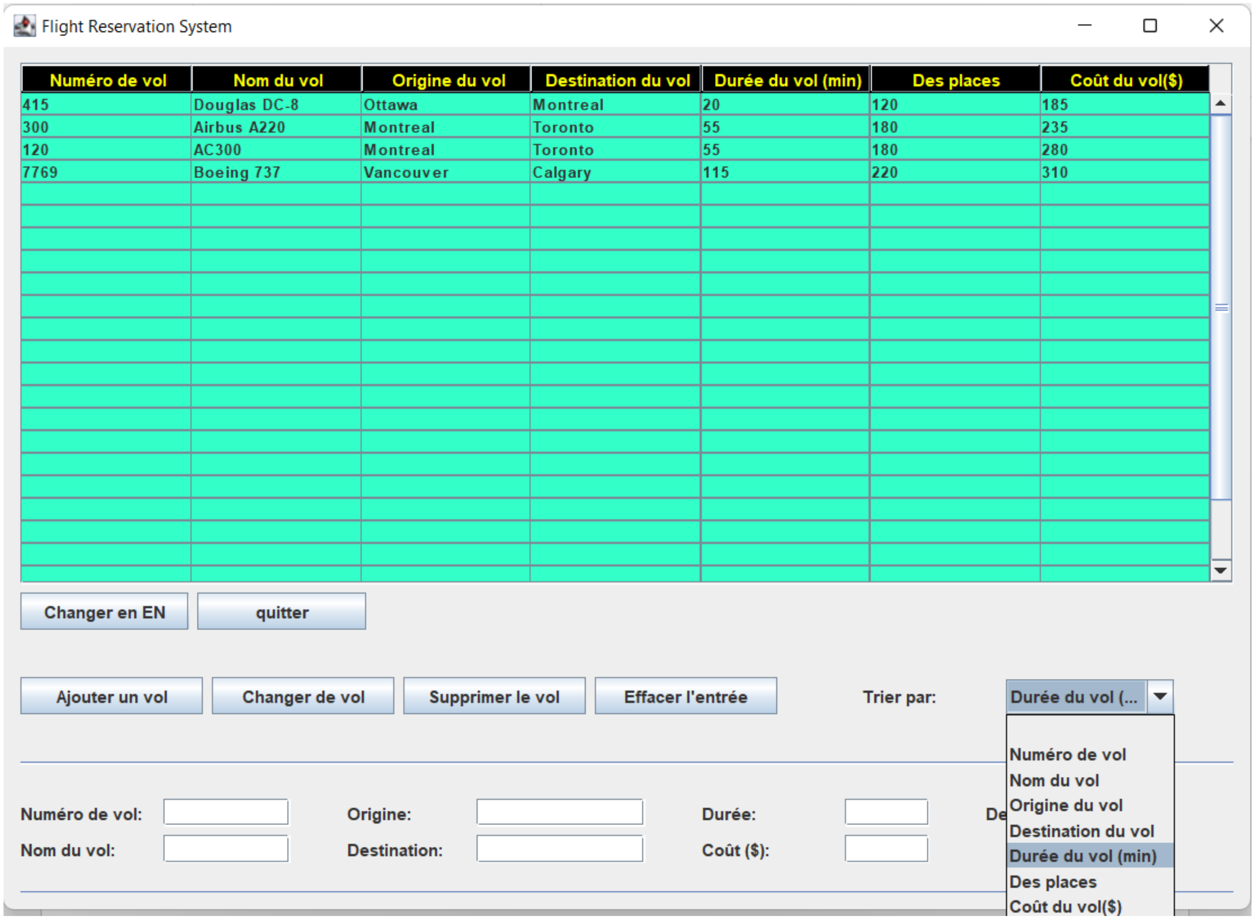


Screenshot of GUI when set to en_US Locale as it’s shown you can add, remove and change flights. You can also clear the input form (information for the flights). The table with green background is only used to display flight information and cannot be edited. Any change or update should be done in the input form by using any of the mentioned buttons. Everytime there is a change in the flight information, the flight board will be updated with the new data, so no need for additional action to display information.



Flight Number	Flight Name	Flight Origin	Flight Destination	Flight Duration (min)	Total Seats	Flight Cost ($)
415	Douglas DC-8	Ottawa	Montreal	20	120	185
300	Airbus A220	Montreal	Toronto	55	180	235
120	AC300	Montreal	Toronto	55	180	280
7769	Boeing 737	Vancouver	Calgary	115	220	310

Screenshot of GUI when set to fr_FR



Numéro de vol	Nom du vol	Origine du vol	Destination du vol	Durée du vol (min)	Des places	Coût du vol($)
415	Douglas DC-8	Ottawa	Montreal	20	120	185
300	Airbus A220	Montreal	Toronto	55	180	235
120	AC300	Montreal	Toronto	55	180	280
7769	Boeing 737	Vancouver	Calgary	115	220	310

Trier par: Durée du vol (...

Numéro de vol:

Nom du vol:

Origine:

Destination:

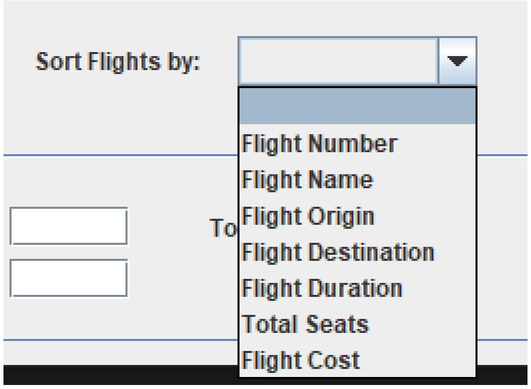
Durée:

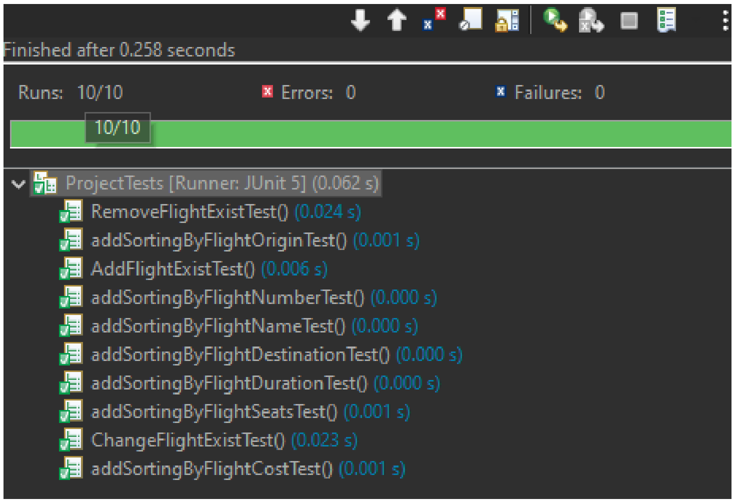
Coût ($):

De

Screenshot of Junit Test class passing

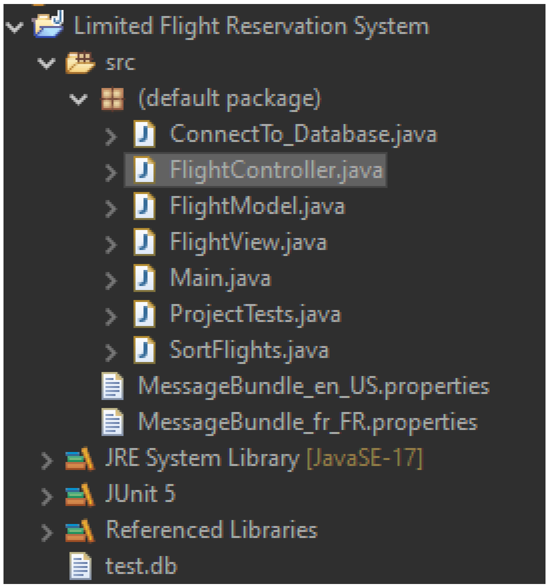
Flight board can be sorted per user selection





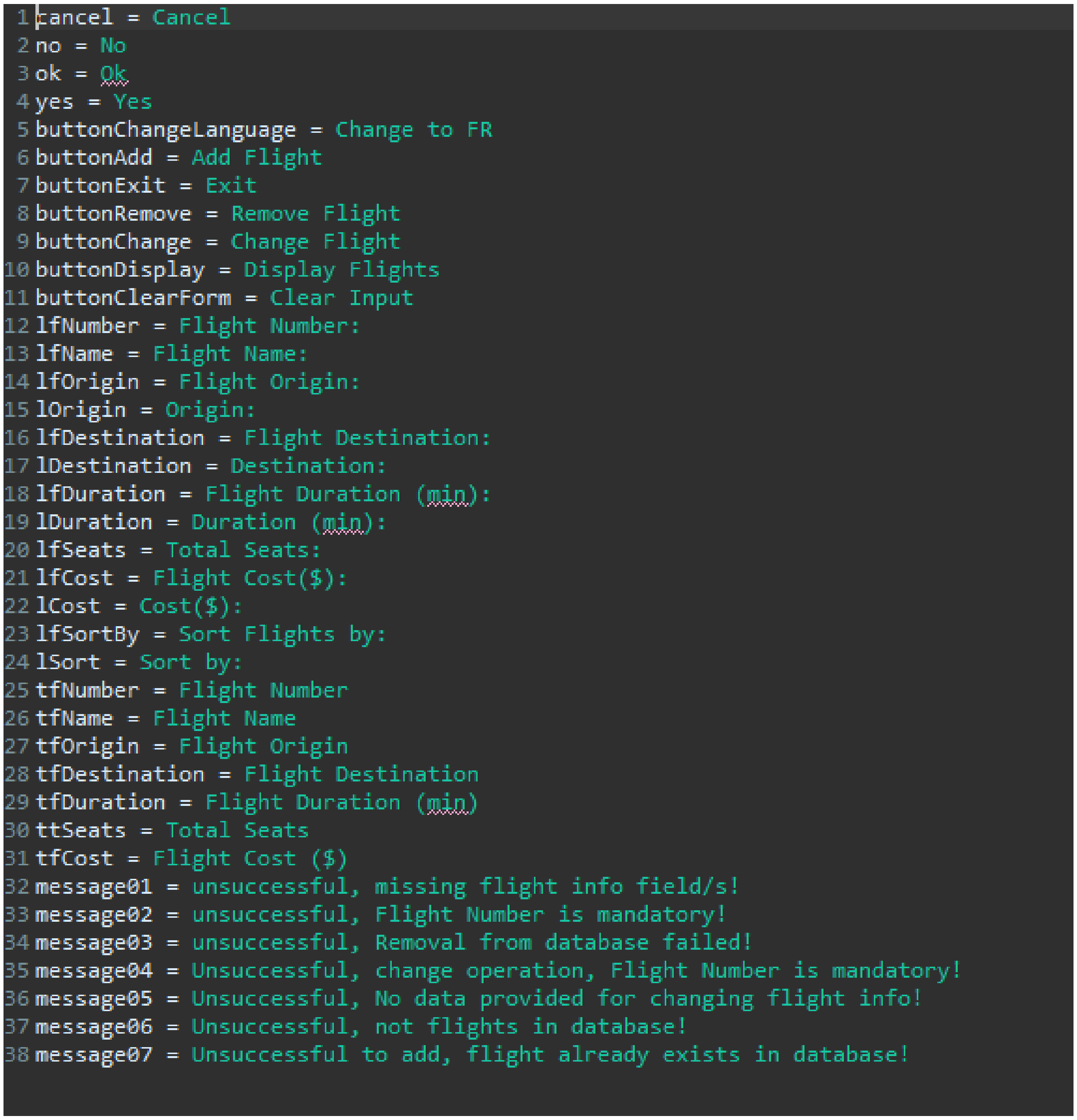
```
Finished after 0.258 seconds

Runs:  10/10          Errors:  0          Failures:  0
       10/10

ProjectTests [Runner: JUnit 5] (0.062 s)
    RemoveFlightExistTest() (0.024 s)
    addSortingByFlightOriginTest() (0.001 s)
    AddFlightExistTest() (0.006 s)
    addSortingByFlightNumberTest() (0.000 s)
    addSortingByFlightNameTest() (0.000 s)
    addSortingByFlightDestinationTest() (0.000 s)
    addSortingByFlightDurationTest() (0.000 s)
    addSortingByFlightSeatsTest() (0.001 s)
    ChangeFlightExistTest() (0.023 s)
    addSortingByFlightCostTest() (0.001 s)
```

Project code structure shows all the classes created, database file(test.db), and the 2 properties files used for Locales.



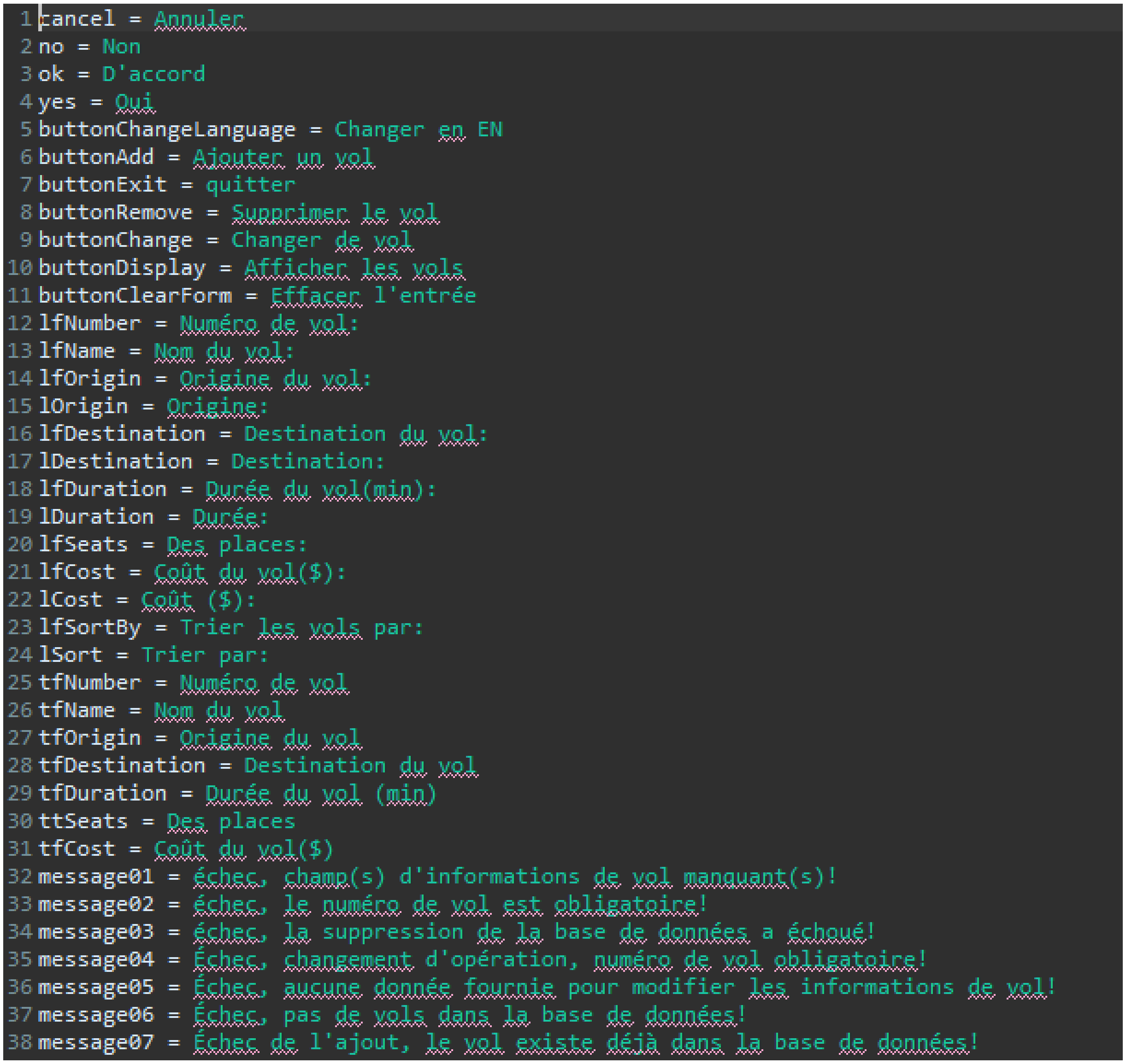
```
Limited Flight Reservation System
  src
    (default package)
      ConnectTo_Database.java
      FlightController.java
      FlightModel.java
      FlightView.java
      Main.java
      ProjectTests.java
      SortFlights.java
    MessageBundle_en_US.properties
    MessageBundle_fr_FR.properties
  JRE System Library [JavaSE-17]
  JUnit 5
  Referenced Libraries
  test.db
```

Properties files for US English Locale(MessageBundle_en_US.properties)



```
cancel = Cancel
no = No
ok = Ok
yes = Yes
buttonChangeLanguage = Change to FR
buttonAdd = Add Flight
buttonExit = Exit
buttonRemove = Remove Flight
buttonChange = Change Flight
buttonDisplay = Display Flights
buttonClearForm = Clear Input
lfNumber = Flight Number:
lfName = Flight Name:
lfOrigin = Flight Origin:
lOrigin = Origin:
lfDestination = Flight Destination:
lDestination = Destination:
lfDuration = Flight Duration (min):
lDuration = Duration (min):
lfSeats = Total Seats:
lfCost = Flight Cost($):
lCost = Cost($):
lfSortBy = Sort Flights by:
lSort = Sort by:
tfNumber = Flight Number
tfName = Flight Name
tfOrigin = Flight Origin
tfDestination = Flight Destination
tfDuration = Flight Duration (min)
ttSeats = Total Seats
tfCost = Flight Cost ($)
message01 = unsuccessful, missing flight info field/s!
message02 = unsuccessful, Flight Number is mandatory!
message03 = unsuccessful, Removal from database failed!
message04 = Unsuccessful, change operation, Flight Number is mandatory!
message05 = Unsuccessful, No data provided for changing flight info!
message06 = Unsuccessful, not flights in database!
message07 = Unsuccessful to add, flight already exists in database!
```

Properties files for FR French Locale(MessageBundle_fr_FR.properties)



```
cancel = Annuler
no = Non
ok = D'accord
yes = Oui
buttonChangeLanguage = Changer en EN
buttonAdd = Ajouter un vol
buttonExit = quitter
buttonRemove = Supprimer le vol
buttonChange = Changer de vol
buttonDisplay = Afficher les vols
buttonClearForm = Effacer l'entrée
lfNumber = Numéro de vol:
lfName = Nom du vol:
lfOrigin = Origine du vol:
lOrigin = Origine:
lfDestination = Destination du vol:
lDestination = Destination:
lfDuration = Durée du vol(min):
lDuration = Durée:
lfSeats = Des places:
lfCost = Coût du vol($):
lCost = Coût ($):
lfSortBy = Trier les vols par:
lSort = Trier par:
tfNumber = Numéro de vol
tfName = Nom du vol
tfOrigin = Origine du vol
tfDestination = Destination du vol
tfDuration = Durée du vol (min)
ttSeats = Des places
tfCost = Coût du vol($)
message01 = échec, champ(s) d'informations de vol manquant(s)!
message02 = échec, le numéro de vol est obligatoire!
message03 = échec, la suppression de la base de données a échoué!
message04 = Échec, changement d'opération, numéro de vol obligatoire!
message05 = Échec, aucune donnée fournie pour modifier les informations de vol!
message06 = Échec, pas de vols dans la base de données!
message07 = Échec de l'ajout, le vol existe déjà dans la base de données!
```