

## Fondamenti di Informatica T2

### **Lab13 - Flights UI**

*Corso di Laurea in Ingegneria Informatica*

Anno accademico 2021/2022

Prof. ROBERTA CALEGARI

Prof. AMBRA MOLESINI

*Dipartimento di Informatica – Scienza e Ingegneria (DISI)*



# Graphic User Interface (GUI)

---

- Realizzare *bene* interfacce utente non è semplice!
- L'interfaccia utente deve essere:
  - **intuitiva e facilmente usabile** (sembra facile...)
  - **ben progettata** dal punto di vista delle **responsabilità** (leggi: ingegnerizzazione)
- Quindi:
  - cambiare la GUI non deve significare cambiare il resto della applicazione
  - **i dati e la logica applicativa che lavora su di essi NON devono essere intimamente collegati alla GUI**



# Graphic User Interface (GUI)

---

- Per ottenere ciò, l'applicazione deve essere organizzata opportunamente
- Alcuni tipici design pattern:
  - **Model View Controller (MVC)**
  - Model View Presenter (MVP)
  - Model View ViewModel (MVVM)
  - Command/Action
  - Delegate
  - ...



# MVC @ Flights

---

- **Lab13 – Flights**

- GUI originariamente realizzata col framework Swing
- Poi, aggiunta ulteriore UI a Console
- Poi, rifatta la GUI col framework JavaFX
- .. ma il Controller è sempre rimasto **lo stesso!**



# Mr. Controller, I suppose

- Affinché la view sia veramente sostituibile, è necessario che *la GUI non contenga alcuna logica riconducibile ai dati*
- Conseguenza: **a fronte di eventi grafici, la view *non deve eseguire direttamente operazioni sui dati*** ma bensì ***redirigerle al controller*** che deciderà come reagire
- Il controller può ad esempio:
  - compiere azioni sul *Model*
  - dare un feedback alla view
  - attivare un'altra view
  - ...



# JavaFX: ripasso

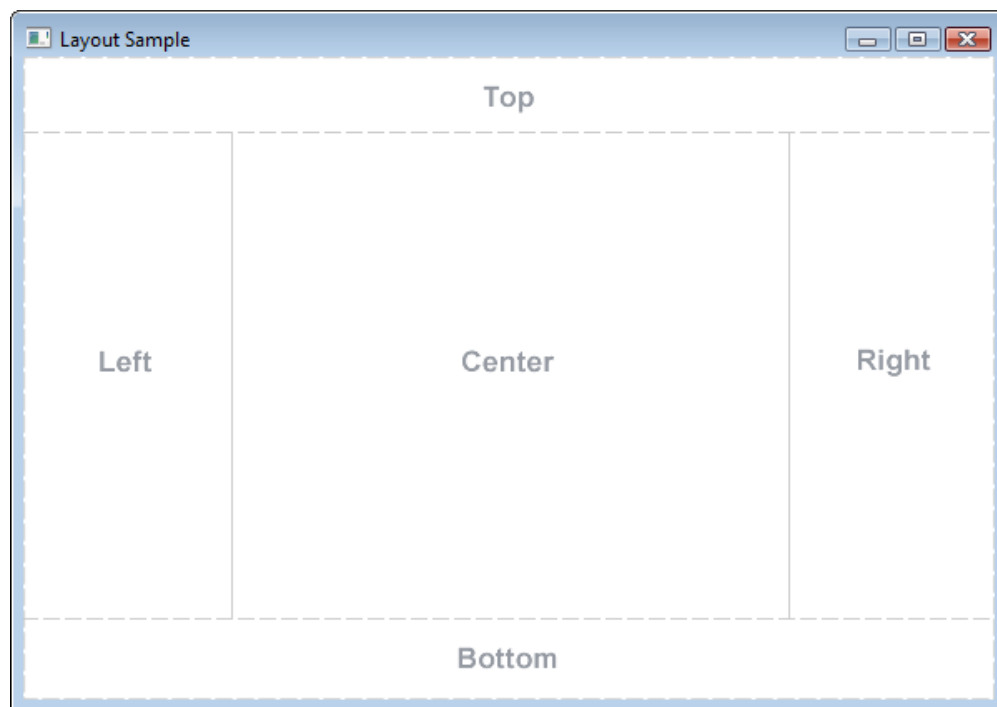
---

- Cosa dobbiamo sapere:
  - Come vengono posizionati i widget sui pannelli?
  - Come si agganciano ai widget gli ascoltatori degli eventi?
  - Nuovo concetto: liste osservabili
  - Architettura generale di una *JavaFX Application*

# Disegno del Pannello

Tipi di pannelli e rispettivi layout

- FlowPane, BorderPane, TilePane, GridPane
- Hbox, VBox



# Agganciare ascoltatori di eventi

- I **widget** emettono **eventi** in risposta alle azioni fisiche dell'utente
- Affinché ci sia una reazione, a ogni evento utile dev'essere agganciato un opportuno **ascoltatore (listener)**
  - ad esempio, i **Button** emettono un **evento di azione** quando cliccati
- L'aggancio si effettua invocando sul widget il giusto metodo (tipicamente: **setOnAction** o analoghi) con argomento una **lambda expression** che specifichi il **comportamento desiderato**
  - nei casi più semplici, nella forma di *method reference*

```
Button searchButton = new Button("Find");  
searchButton.setOnAction(this::myHandle);  
leftPane.getChildren().add(searchButton);
```



```
private void myHandle(ActionEvent event)
```



# Liste osservabili

- I widget che lavorano su elenchi di elementi usano tutti una versione particolare di lista, la *lista osservabile*
  - una lista in grado di *avvisare delle eventuali modifiche* (aggiunta, rimozione, spostamento di elementi, ecc.) tramite *eventi*
  - un meccanismo semplice ed efficace per *propagare informazioni*
- Una lista osservabile si ottiene *incapsulando una normale collezione* tramite la factory **`javafx.collections.FXCollections`**

Crea la collezione osservabile

```
List<Airport> airports = controller.getSortedAirports();  
ObservableList<Airport> observableAirports = FXCollections.observableArrayList airports);  
  
ComboBox<Airport> departureAirportComboBox = new ComboBox<Airport>(observableAirports);  
departureAirportComboBox.setEditable(false);  
departureAirportComboBox.setValue(observableAirports.get(0));
```

Imposta come selezionato il primo elemento.

# Anatomia di Application

```
public class FlightsApplication extends Application {
```

```
    public void init(){  
        // do nothing  
    }
```

Applicazione non ancora attivata: si possono compiere solo azioni che non coinvolgano la grafica

```
    public void start(Stage stage) {  
        DataManager dataManager = createDataManager();  
        if (!readData(dataManager))  
            return;
```

Creazione dei reader e del DataManager

Se la lettura dei dati non va a buon fine, si chiude qui

```
        MyController controller = new MyController(dataManager);  
        stage.setTitle("Flights Scheduler");
```

Creazione del controller

```
        BorderPane root = new MainPane(controller);  
        Scene scene = new Scene(root, 980, 480);  
        stage.setScene(scene);  
        stage.show();  
    }
```

Creazione dell'elemento root della grafica: il pane che contiene tutto.

Per verificare il funzionamento dell'applicazione *in modo indipendente dalla persistenza* (magari non ancora realizzata/debuggata) basta sostituire l'implementazione del metodo **createDataManager** 😊



# Anatomia di Application

- Versione completa con persistenza operativa e funzionante

```
protected DataManager createDataManager() {  
    MyCitiesReader citiesReader = new MyCitiesReader();  
    MyAircraftsReader aircraftsReader = new MyAircraftsReader();  
    MyFlightScheduleReader flightScheduleReader = new MyFlightScheduleReader();  
  
    DataManager dataManager = new DataManager(citiesReader,  
        aircraftsReader, flightScheduleReader);  
    return dataManager;  
}
```

- Versione "mock" con persistenza simulata

```
public class FlightsApplicationMock extends FlightsApplication {  
  
    @Override  
    protected DataManager createDataManager() {  
        DataManager dataManager = new DataManagerMock();  
        return dataManager;  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

# Flights

- Testo del compito



# Flights

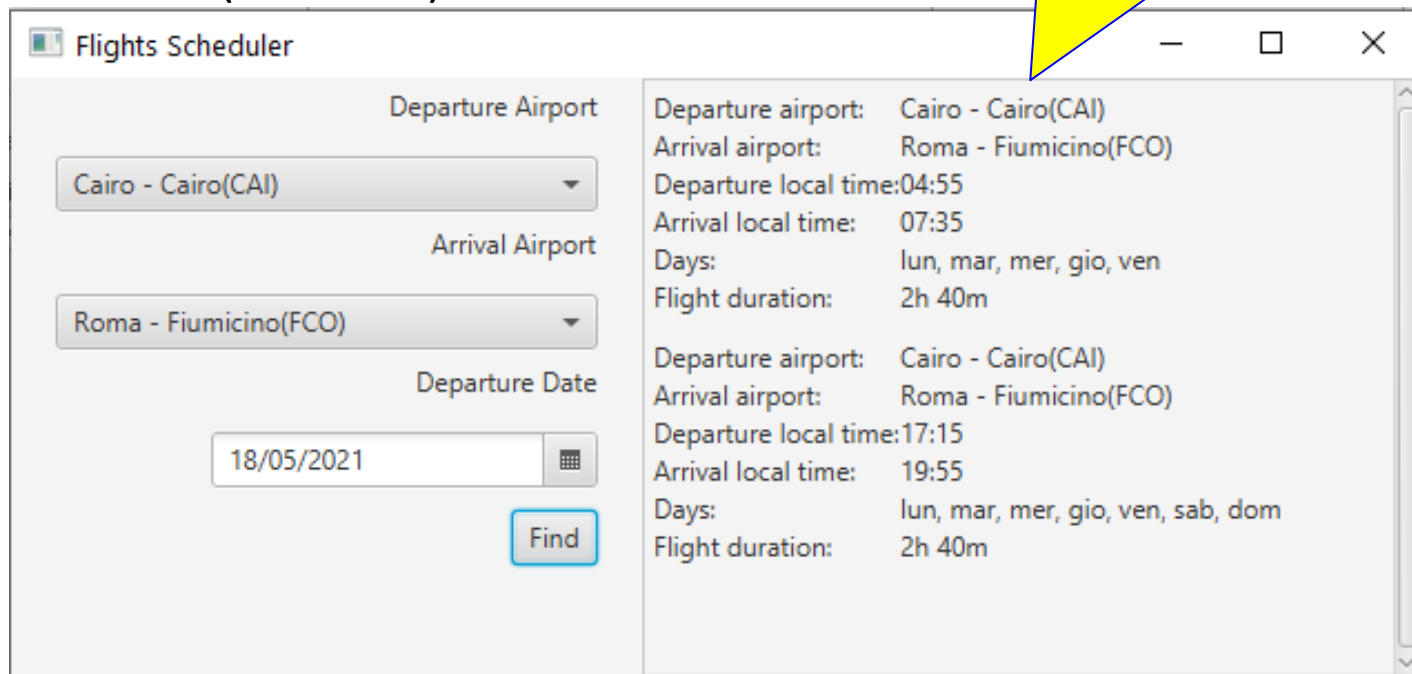
- **Classico compito** con
  - PARTE 1: modello, persistenza
  - PARTE 2: GUI
- Alcune classi sono già fornite nello start kit
  - **FlightScheduleListPanel**
  - **FlightSchedulePanel**
- Da realizzare il frame principale (**MainPane**)
- Adottare il seguente ordine di realizzazione:
  1. Modello
  2. Persistenza
  3. GUI: UI & Controller

In caso di problemi nella persistenza, nello start kit c'è un'Application apposita in cui la persistenza è realizzata mediante mock (**FlightsApplicationMock**)

# MainPane

- Deve consentire all'utente di:
  - inserire l'aeroporto di partenza (**ComboBox**)
  - inserire l'aeroporto di arrivo (**ComboBox**)
  - inserire la data di partenza (**DatePicker**)
  - cercare i voli (**Button**)

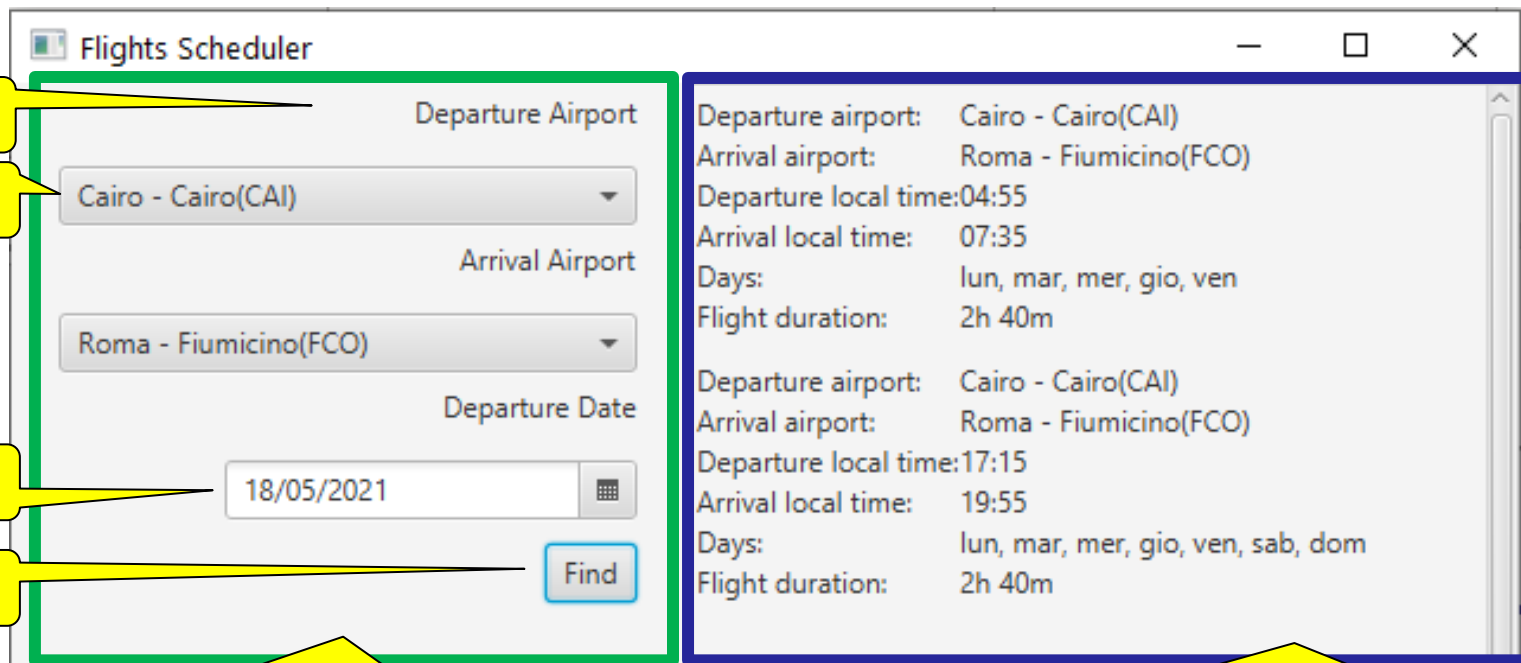
OUTPUT: vedere i voli disponibili nella data selezionata



The screenshot shows a window titled "Flights Scheduler" with a light gray background. On the left side, there are three input fields: "Departure Airport" with a dropdown menu showing "Cairo - Cairo(CAI)", "Arrival Airport" with a dropdown menu showing "Roma - Fiumicino(FCO)", and "Departure Date" with a text box containing "18/05/2021" and a calendar icon. Below these fields is a blue "Find" button. On the right side, there is a list of flight details. The first entry shows: "Departure airport: Cairo - Cairo(CAI)", "Arrival airport: Roma - Fiumicino(FCO)", "Departure local time: 04:55", "Arrival local time: 07:35", "Days: lun, mar, mer, gio, ven", and "Flight duration: 2h 40m". The second entry shows: "Departure airport: Cairo - Cairo(CAI)", "Arrival airport: Roma - Fiumicino(FCO)", "Departure local time: 17:15", "Arrival local time: 19:55", "Days: lun, mar, mer, gio, ven, sab, dom", and "Flight duration: 2h 40m". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

# MainPane

- **MainPane** deriva da **BorderPane**



**Flights Scheduler**

Departure Airport: Cairo - Cairo(CAI)

Arrival Airport: Roma - Fiumicino(FCO)

Departure Date: 18/05/2021

Find

Flight Details 1:

- Departure airport: Cairo - Cairo(CAI)
- Arrival airport: Roma - Fiumicino(FCO)
- Departure local time: 04:55
- Arrival local time: 07:35
- Days: lun, mar, mer, gio, ven
- Flight duration: 2h 40m

Flight Details 2:

- Departure airport: Cairo - Cairo(CAI)
- Arrival airport: Roma - Fiumicino(FCO)
- Departure local time: 17:15
- Arrival local time: 19:55
- Days: lun, mar, mer, gio, ven, sab, dom
- Flight duration: 2h 40m

## VBox nel MainPane

- allineamento **BASELINE\_RIGHT**
- impostato a sinistra (**setLeft**)

## FlightScheduleListPanel nel MainPane

- è uno **ScrollPane** che contiene un **VBox** che contiene vari **FlightSchedulePanel**
- impostato al centro (**setCenter**)

# UI da console

```
16 - LAS: Las Vegas - Las Vegas
17 - LGW: London - Gatwick
18 - LHR: London - Heathrow
19 - STN: London - Stansted
20 - LIN: Milano - Linate
21 - MXP: Milano - Malpensa
22 - JFK: New York - John F. Kennedy
23 - LGA: New York - La Guardia
24 - EWR: New York - Newark
25 - NCE: Nice - Nice
26 - OLB: Olbia - Olbia
27 - PMO: Palermo - Palermo
28 - BVA: Paris - Beauvais
29 - CDG: Paris - De Gaulle
30 - ORY: Paris - Orly
31 - PRG: Prague - Prague
32 - CIA: Roma - Ciampino
33 - FCO: Roma - Fiumicino
34 - SFO: San Francisco - San Francisco
35 - TYO: Tokyo - Tokyo
36 - VRN: Verona - Verona
0 - Termina
Scegli...
```

11

Immettere Data: :

14/05/19

Partenza: 14:30 - Arrivo: 16:15 - Durata: PT1H45Mmin

Partenza: 07:30 - Arrivo: 09:00 - Durata: PT1H30Mmin

Partenza: 18:35 - Arrivo: 20:05 - Durata: PT1H30Mmin

Immettere 'c' per continuare, 'q' per uscire. :

Per completezza dell'esercitazione troverete anche una UI da console

## INPUT

- scelta aeroporto di partenza e di arrivo da una lista
- data

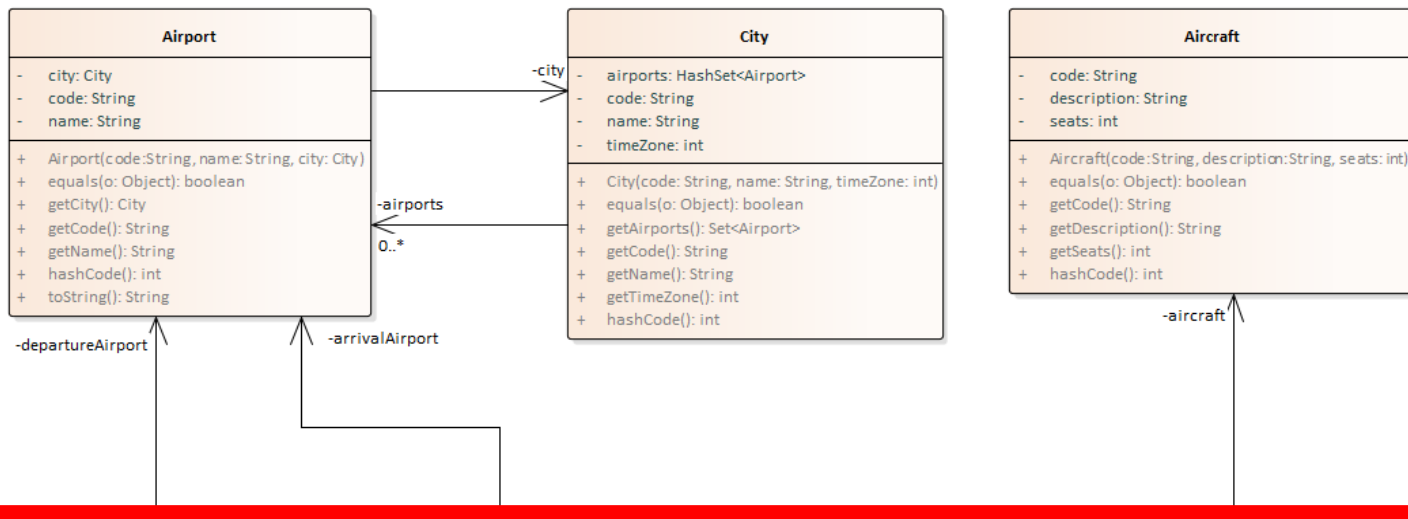
## OUTPUT

- lista dei voli

Immissione della data nel formato  
**FormatStyle.SHORT, Locale.ITALY**



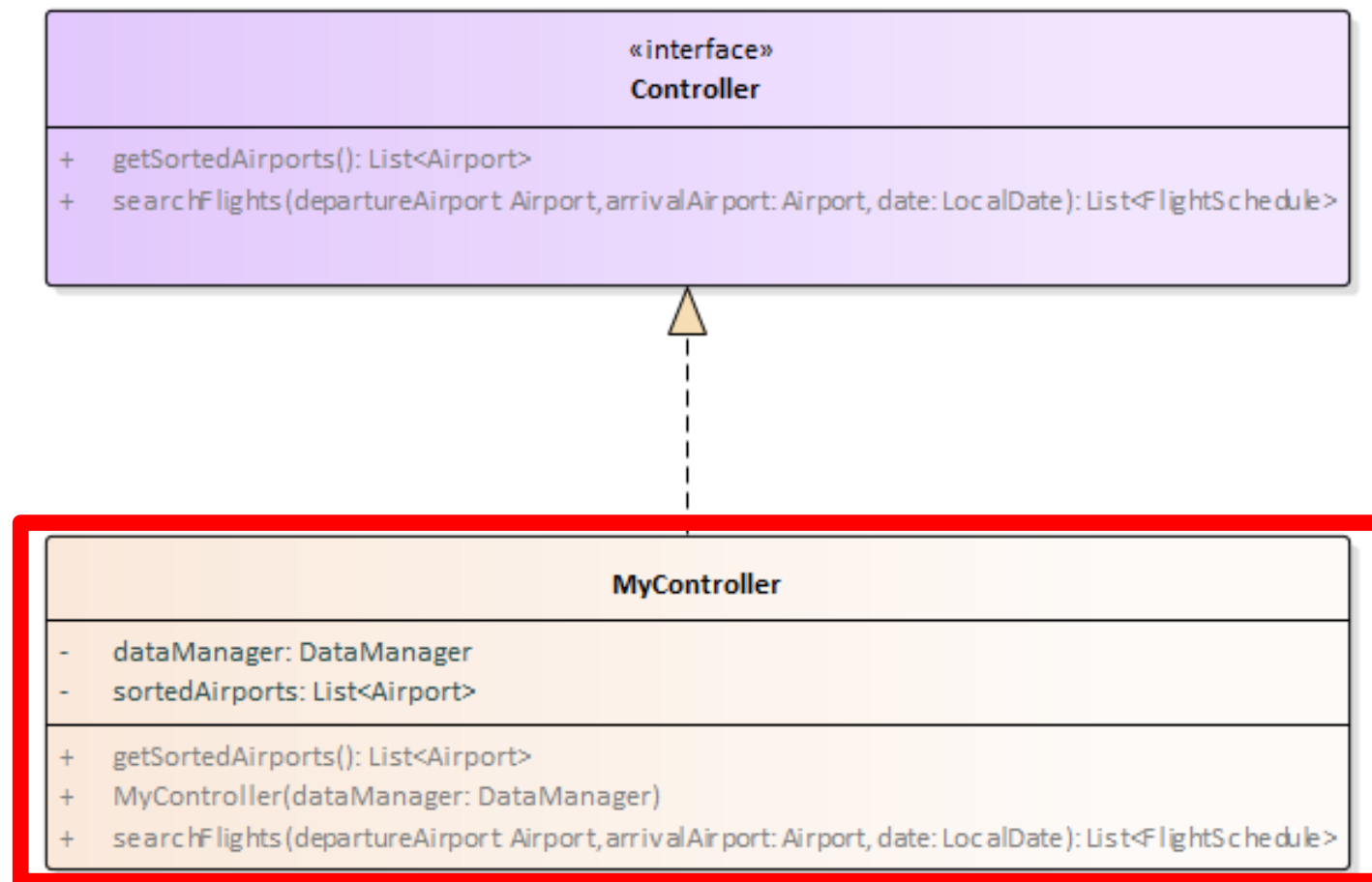
# Flights: modello



Da  
Fare

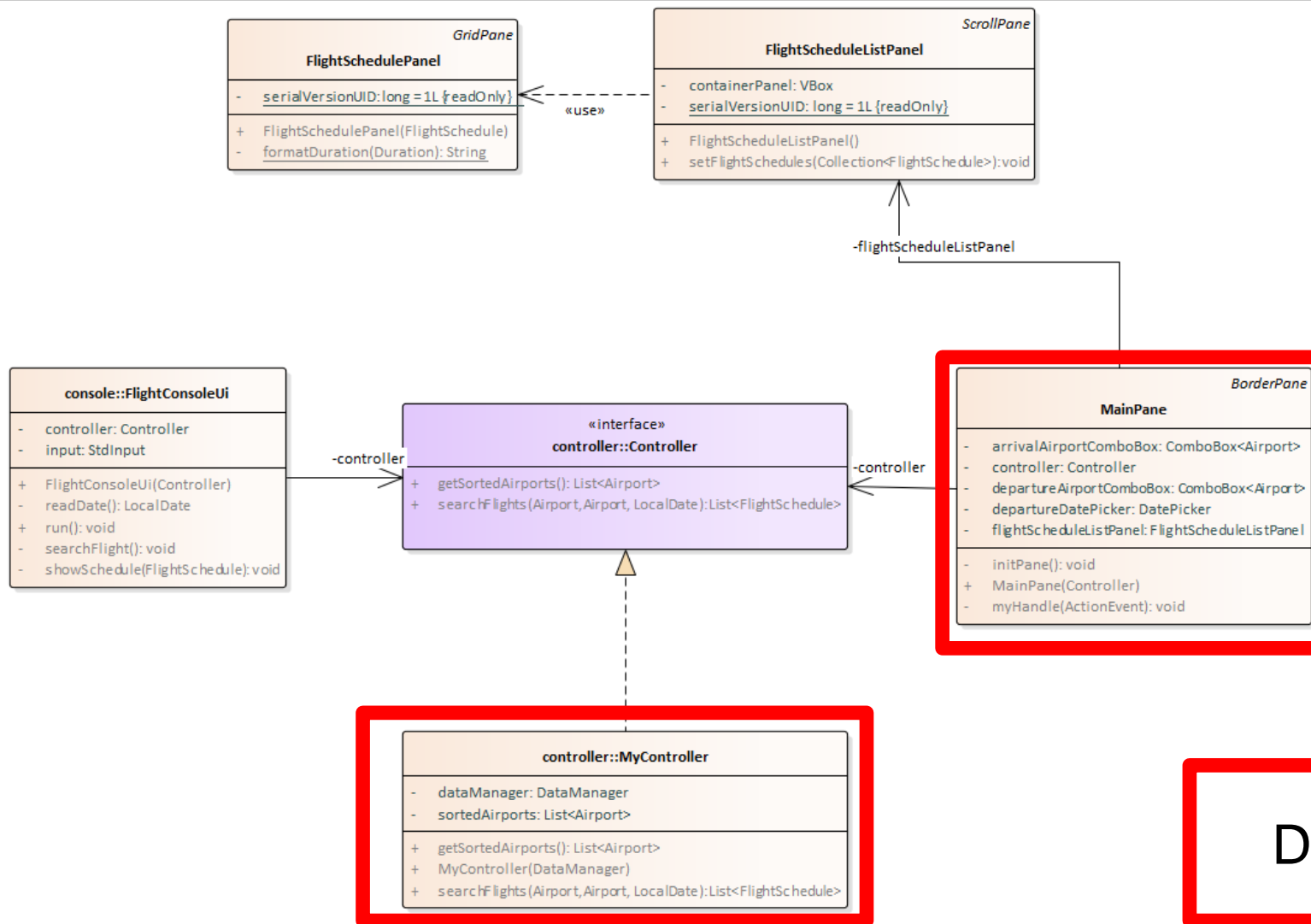


# Flights: controller



Da  
Fare

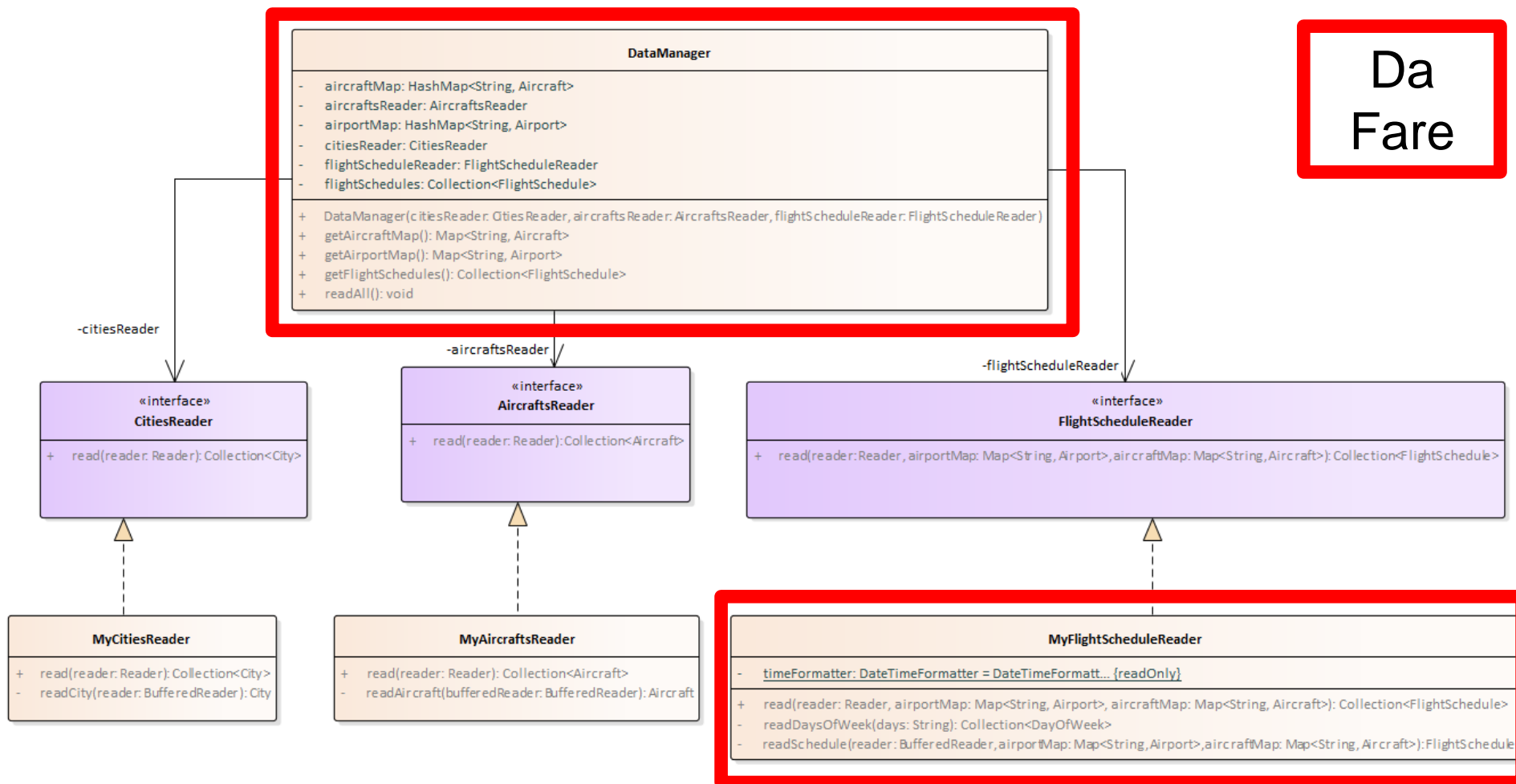
# Flights: controller-view



Da Fare

# Flights: persistence

Da  
Fare



# Organizzazione Package

## Lab13-Flights

### src

#### flights

- > FlightsApplication.java
- > FlightsApplicationMock.java
- > MainConsoleUI.java

Main delle applicazioni

- > flights.controller
- > flights.model
- > flights.persistence
- > flights.ui
- > flights.ui.console
- > fondt2.ioutils

#### test

Test e relativi mock

- > JRE System Library [jdk-17.0.2]
- > JUnit 5
- > JavaFx17
- Aircrafts.txt
- Cities.txt
- FlightSchedule.txt

# Hey!

---



**KEEP  
CALM  
AND  
HAPPY  
CODING**