

ANALISI

Descrizione

Assassination_Airport è un progetto che permette all'utente di inserire un certo numero di voli (non definiti a priori) e poterli ricercare per due valori chiave: il **codice volo** e la **destinazione**.

Per inserire un nuovo volo l'utente dovrà compilare un piccolo form formato dai seguenti campi:

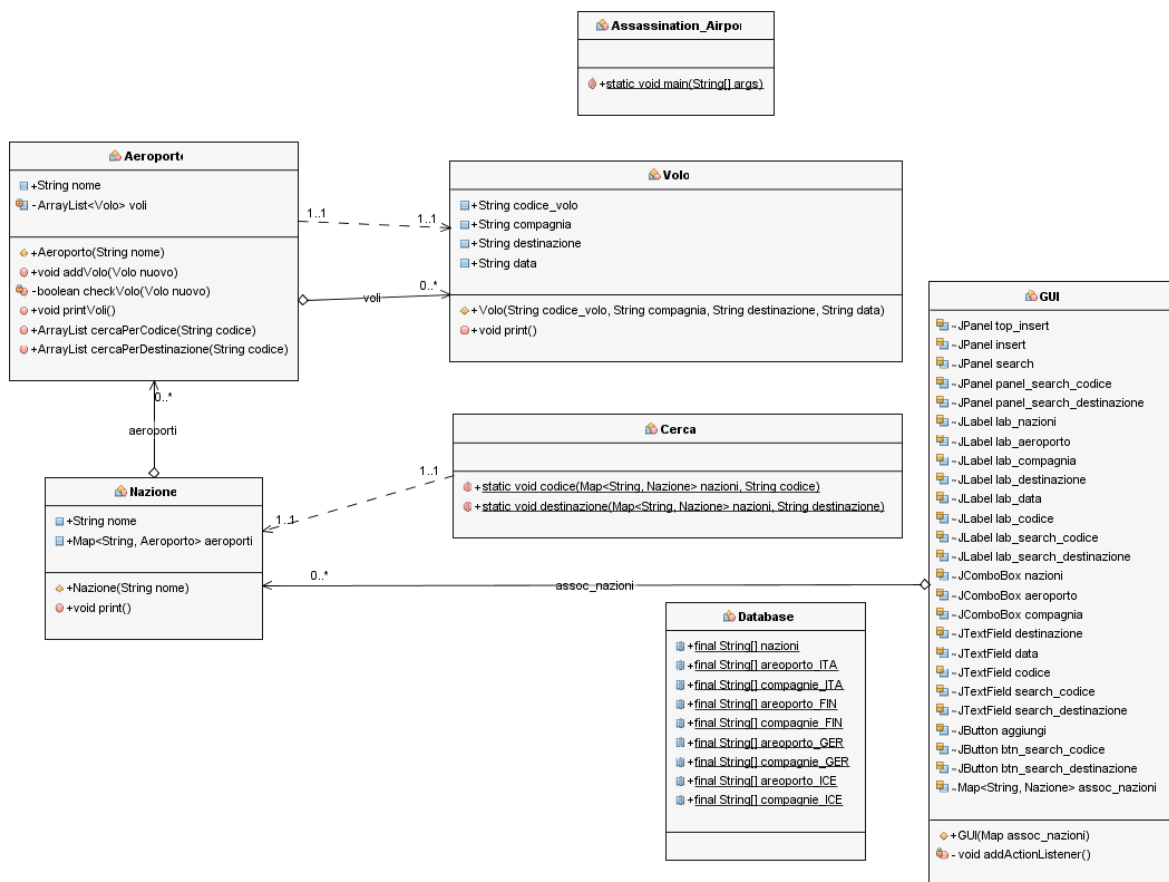
- La Nazione da dove il volo parte (selezionabile tramite un menu a tendina)
- L'Aeroporto da dove il volo parte (selezionabile tramite un menu a tendina)
- La Compagnia aerea (selezionabile tramite un menu a tendina)
- La Destinazione (l'utente deve inserirla nell'apposita casella)
- La Data (l'utente deve inserirla nell'apposita casella)
- Il Codice Volo (l'utente deve inserirlo nell'apposita casella)

Successivamente tramite la pressione dell'apposito bottone "Aggiungi !" i dati verranno controllati ed il volo sarà aggiunto.

Per la ricerca ci sono **due** caselle di testo, una per il **codice volo** e l'altra per la **destinazione**.

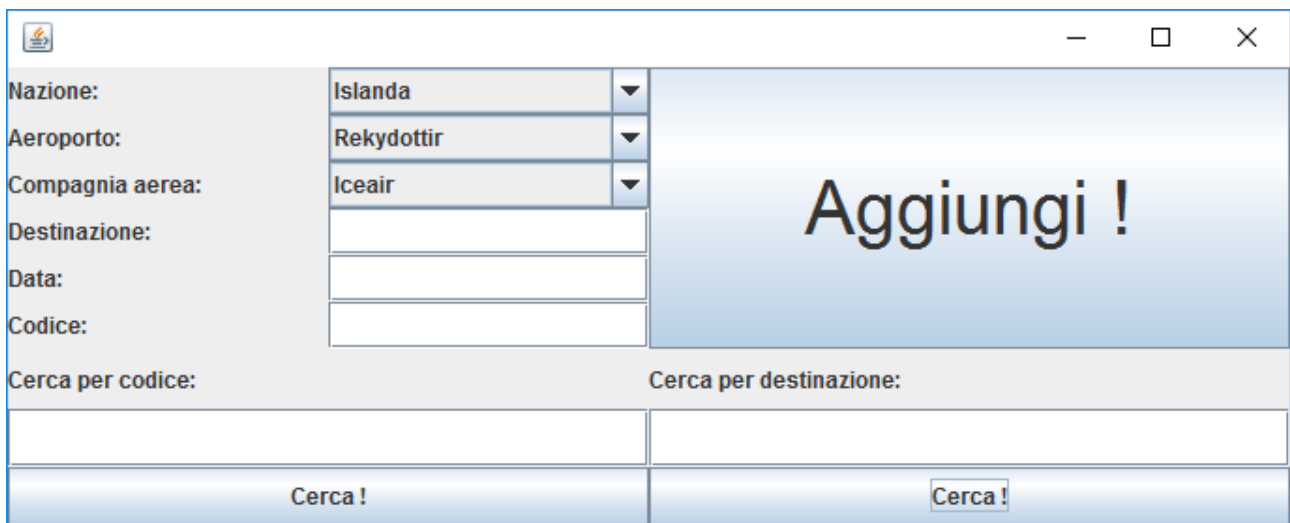
L'utente dovrà scrivere in una delle caselle di testo e premere il relativo bottone per effettuare la ricerca. I risultati verranno stampati a video.

UML



(consigliato zoom sullo schema per vedere meglio)

L'Interfaccia Grafica



(interfaccia del programma in esecuzione)

L'interfaccia grafica è composta da 2 parti principali:

- La parte superiore
- La parte inferiore

Ambedue sono formate da un JPanel gestito con il GridBoxLayout.

Analizziamo la parte superiore:

È composta da 2 colonne con 6 righe. Nella prima colonna si trovano i campi da completare per inserire un volo:

- 3 JComboBox che contengono dei dati prestabiliti
- 3 JTextField per l'inserimento dei dati

Nella seconda colonna c'è il bottone per aggiungere un nuovo volo.

La parte inferiore invece è adibita alla ricerca dei voli inseriti ed è sempre un JPanel con 2 righe e 2 colonne al cui interno ci sono:

- Le JLabel per far capire all'utente a cosa serve quello spazio
- Le JTextField per l'inserimento della chiave di ricerca
- I JButton per l'avvio della ricerca

Ogni bottone stampa a video un messaggio tramite un JOptionPane.

Le classi

È bene analizzare solo 2 classi di questo progetto, dato che le altre si spiegano da sole.

La classe **Database** è una classe statica che contiene una serie di **Array di Stringhe** per la creazione delle JComboBox e l'Array Associativo Nazioni.

La classe **Cerca**, anch'essa statica, contiene 2 metodi statici, **codice** e **destinazione**.

Questi metodi permettono la ricerca tramite codice e destinazione di uno o più voli e stampano a video i voli trovati.

In ingresso richiedono un **Array Associativo** di nazioni e una **Stringa** come chiave di ricerca.

Il codice

Per la realizzazione di questo esercizio ho utilizzato risorse che non sono state approfondite in classe ma che ho scoperto e studiato da solo, come gli **Array Associativi** e le **Lambda Expressions**.

```
Map<String, Nazione> nazioni = new HashMap<>();
{Germania=assassination_airport.Nazione@15db9742,
  Italia=assassination_airport.Nazione@6d06d69c,
  FinalIndia=assassination_airport.Nazione@7852e922,
  Islanda=assassination_airport.Nazione@4e25154f}
```

(dichiarazione e stampa a video di un Array Associativo)

Ho deciso di utilizzare un array associativo in quanto (lo dice il nome stesso “associativo”) permette di associare, di solito, una stringa ad un oggetto, in questo caso il nome della nazione al relativo oggetto Nazione e il nome dell’aeroporto al relativo oggetto Aeroporto. Questo mi ha permesso un notevole risparmio di tempo e una semplificazione del codice, in quanto per trovare un determinato oggetto serve solo una stringa che lo identifica. È per questo che ho optato per la scelta delle JComboBox con dei campi limitati (l’utente può scegliere solo alcuni valori predefiniti).

```
nazioni.put(nazione, new Nazione(nazione));
nazioni.get("Germania");
```

(aggiunta e prelevamento di un elemento in un array associativo)

```
aeroporti.forEach((String s, Aeroporto a) -> {
    System.out.println("\tNome: "+a.nome);
    a.printVoli();
});
```

(lambda expression per ciclare un array associativo)

Le Lambda Expression permettono di creare funzioni anonime (come quelle del c per intenderci) a cui possono essere passati dei parametri e che a loro volta possono essere passate a metodi, rendendo così il Java un linguaggio funzionale.

Prendiamo in considerazione l’immagine sopra:

aeroporti è un array associativo <Stringa, Aeroporto>, il metodo forEach() permette di ciclare questo array elemento per elemento (va ricordato che un elemento di questo array è formato da 2 elementi: una Stringa e un Aeroporto). Ogni elemento viene passato alla lambda expression che andrà a operare sui suoi attributi, in questo caso li stampa.

Gestione degli Eventi

Gli eventi in questo esercizio sono stati gestiti tramite le **Lambda Expressions** presenti all'interno del metodo `addActionListener` appositamente creato.

```
/*  
    ACTION LISTENER CERCA CODICE  
*/  
btn_search_codice.addActionListener((ActionEvent e) -> {  
    Cerca.codice(assoc_nazioni, search_codice.getText());  
});  
  
/*  
    ACTION LISTENER CERCA DESTINAZIONE  
*/  
btn_search_destinazione.addActionListener((ActionEvent e) -> {  
    Cerca.destinazione(assoc_nazioni, search_destinazione.getText());  
});
```

(due esempi di lambda expression per la gestione degli ActionListener)

In questo modo il codice risulta più chiaro e leggibile, in quanto non è necessario sovrascrivere un metodo generale che deve controllare ogni volta quale elemento ha generato l'evento, ma basterà creare una lambda expression per ogni elemento che genera un evento per poterlo gestire.

Nella gestione degli ActionListener manca un dettagliato controllo dei dati inseriti dall'utente, cioè l'unico controllo che viene fatto è che le caselle di testo non siano nulle e che non vengano inseriti 2 voli uguali nello stesso aeroporto. Non viene, per esempio, controllato se la data immessa sia effettivamente una data con un formato standard. Ho preso questa decisione in quanto non ritenevo necessario questo controllo e ho preferito concentrarmi su altre parti del codice più importanti.