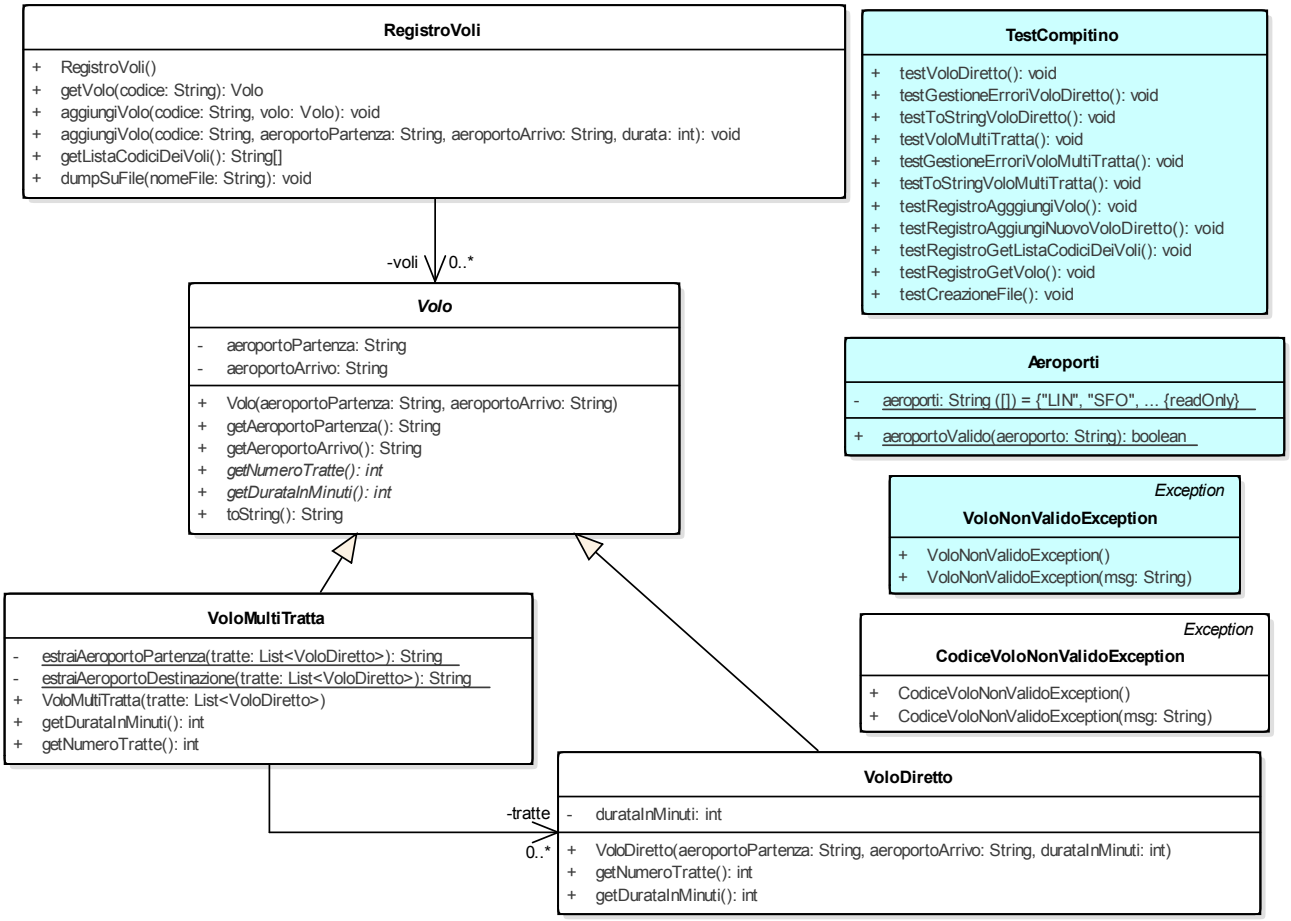
Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

Esempio Secondo Compitino

Testo parte di pratica

Si consideri un'applicazione per gestire voli aerei che tiene conto del caso in cui il collegamento fra un aeroporto di partenza e di destinazione avvenga in una singola tratta (voli diretti) o prevedendo scali intermedi (voli multi tratta). Il sistema da implementare è rappresentato in maggior dettaglio dal seguente diagramma UML:



Le classi in azzurro sono già implementate (`Aeroporti`, classe `VoloNonValidoException` e classe `TestCompitino`). Le rimanenti sono da implementare nella loro interezza secondo le seguenti specifiche.

Classi `Volo`, `VoloDiretto`, `VoloMultiTratta`

- ✓ Rappresentano i tipi di voli aerei gestiti dal sistema, ovvero voli diretti (senza scalo) e voli multi tratta (che prevedono uno o più scali intermedi), organizzati secondo una gerarchia di generalizzazione-specializzazione in cui la classe `Volo` rappresenta genericamente i voli di entrambi i tipi. Lo studente implementi le tre classi come opportuno per realizzare le loro relazioni, ottimizzare la gestione di caratteristiche comuni a diversi tipi di voli e favorire l'uso polimorfo degli oggetti di tipo `Volo`, tenendo opportunamente conto di quanto segue:
  - Tutti i voli sono caratterizzati da: un aeroporto di partenza e uno di destinazione, il numero di tratte previste e la durata complessiva del volo. Queste 4 caratteristiche sono immutabili e accessibili con metodi `get` (vedi diagramma).
  - Un volo diretto viene istanziato con il costruttore della classe `VoloDiretto`, fornendo come parametri gli aeroporti, di partenza e destinazione, e la durata del volo in minuti. Per definizione,

questo tipo di volo consiste sempre di una sola tratta. Perché il volo possa essere costruito in modo valido, la durata del volo deve essere un numero positivo e gli aeroporti di partenza e destinazione devono essere verificati secondo il metodo `Aeroporti.aeroportoValido(String)`.

- Un volo multi tratta viene istanziato con il costruttore della classe `VoloMultiTratta`, fornendo come parametro una lista (`List`) di oggetti `VoloDiretto`, che rappresenta l'insieme delle singole tratte (intesa ciascuna come volo diretto) da percorrere. La durata di un `VoloMultiTratta` è ottenuta come somma delle durate di tutte le tratte previste. Perché il volo possa essere costruito in modo valido, devono esserci 2 o più tratte e deve esserci corrispondenza fra aeroporto di arrivo e di partenza delle tratte che si succedono nella lista di input.
- Se i parametri passati a uno dei costruttori non corrispondono a un volo valido secondo quanto previsto sopra per ogni tipo di volo, il costruttore in questione solleva un'eccezione di tipo `VoloNonValidoException`.
- Per ogni volo, il metodo `toString` restituisce una stringa che descrive il volo. La stringa restituita deve includere: aeroporto di partenza, aeroporto di destinazione, il numero di tratte previste e la durata complessiva del volo.

Classe `RegistroVoli`:

- ✓ Rappresenta un registro dei voli disponibili. Si usi una hash-map (`java.util.HashMap`) per rappresentare l'associazione fra il registro e i voli: ogni volo viene identificato con un codice di tipo `String` che ne rappresenta la chiave di accesso nella hash-map.
- ✓ Il metodo `aggiungiVolo(codice, volo)` aggiunge un volo al registro usando il codice passato come parametro come chiave di accesso nella hash-map. Il codice deve essere una stringa lunga esattamente 5 caratteri, altrimenti il volo ritorna una eccezione di tipo `CodiceVoloNonValidoException`. Questa eccezione deve essere implementata.
- ✓ Il metodo `aggiungiVolo(codice, aeroportoPartenza, aeroportoDestinazione, durata)` è un *overload* del metodo sopradescritto per aggiungere un nuovo volo diretto al registro. Il volo viene creato in base ai parametri passati. Il codice deve soddisfare lo stesso vincolo di cui sopra.
- ✓ Il metodo `getVolo` restituisce un volo dato il suo codice, o `null` se non esiste nessun volo con il codice dato.
- ✓ Il metodo `getListaCodiciDeiVoli` restituisce un array di stringhe che contiene **tutti e soli** i codici dei voli presenti nel registro.
- ✓ Il metodo `dumpSuFile(String fileName)` scrive sul file fileName tutte gli elementi di registro volo. Ogni elemento deve occupare una riga differente. Il codice deve essere separato dal volo da uno spazio. Il metodo può produrre eccezioni di tipo `IOException`.

Usare i test nella classe `TestCompitino` per validare il codice implementato.