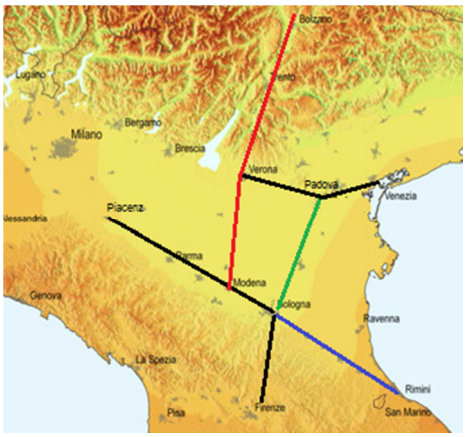
ESAME DI FONDAMENTI DI INFORMATICA T-2 del 12/09/2012
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)

Le *Autostrade di Zannonia* hanno commissionato un'applicazione per il calcolo del costo delle soluzioni di viaggio sulla propria rete autostradale, rappresentata in figura.



L'applicazione deve proporre le soluzioni di viaggio fra i caselli di entrata e uscita richieste, ordinandoli – laddove ce ne siano più d'una – o per lunghezza complessiva o per costo, secondo le preferenze indicate dall'utente nella GUI.

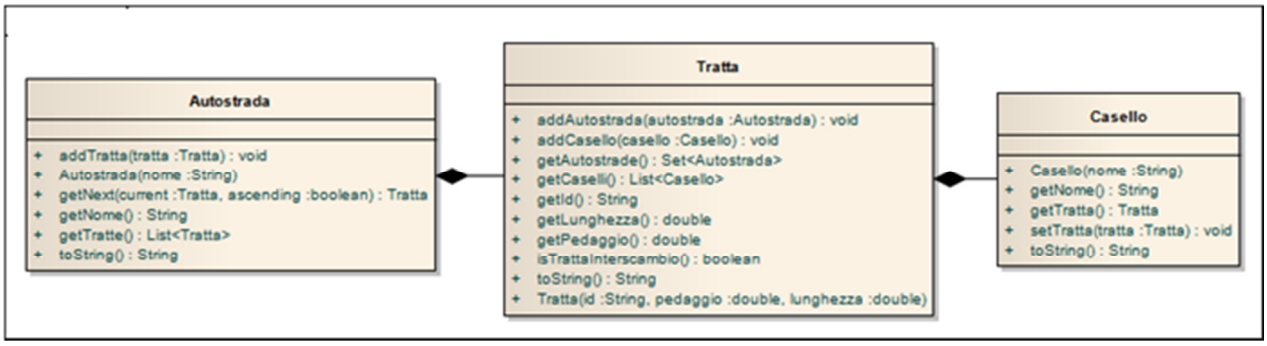
I file di testo autostrade.txt e tratte.txt contengono le descrizioni rispettivamente delle autostrade e delle tratte che le compongono, nei formati più oltre specificati.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA.
La società adotta il modello di tariffazione "a sistema aperto", in cui i caselli sono ad accesso libero, senza barriere. Per questo, una *autostrada* è vista come una sequenza di *tratte,* ognuna caratterizzata da una lunghezza, un insieme di caselli e un pedaggio: il pedaggio *complessivo* da pagare per un dato percorso fra due caselli qualsiasi A,B è quindi determinato semplicemente sommando i pedaggi delle tratte attraversate per andare da A a B.
Alcune tratte (*tratte di interscambio*) appartengono contemporaneamente a più autostrade e permettono quindi di cambiare autostrada se il percorso lo richiede.

Parte 1 (punti: 20)

Dati (namespace zannonia.model) *(punti: 10)*

I dati relativi alle autostrade e alle tratte sono essere organizzati in accordo al diagramma UML in figura.



SEMANTICA:

a) La classe ***Casello*** (fornita nello start kit) rappresenta un casello autostradale col proprio nome univoco; il metodo ***setTratta*** permette di associare a un casello la tratta di appartenenza.

b) La classe ***Tratta*** **(da realizzare)** rappresenta una tratta dell'autostrada: è caratterizzata da un identificatore univoco, un pedaggio e un elenco (eventualmente vuoto) di caselli, forniti in fase di costruzione e mai più modificati successivamente. Essa offre i seguenti metodi:

- ***getId*** per ottenere l'identificatore univoco della tratta;
- ***getPedaggio*** per ottenere il pedaggio relativo all'attraversamento della tratta;
- ***getLunghezza*** per ottenere la lunghezza della tratta (in km);
- ***getCaselli*** per ottenere l'elenco dei caselli appartenenti alla tratta;

- ***aggiungiAutostrada*** per aggiungere un'autostrada alla tratta (ossia, affermare che la tratta in questione appartiene anche a quell'autostrada; una tratta può appartenere a più autostrade);
- ***getAutostrade*** per ottenere l'insieme delle autostrade a cui la tratta appartiene;
- ***isTrattaInterscambio*** per sapere se è una tratta di interscambio, ossia una di quelle tratte che appartengono a due o più autostrade;
- ***addCasello*** per aggiungere un casello alla tratta (avendo cura di verificare che tale casello non sia già contenuto nella tratta stessa).

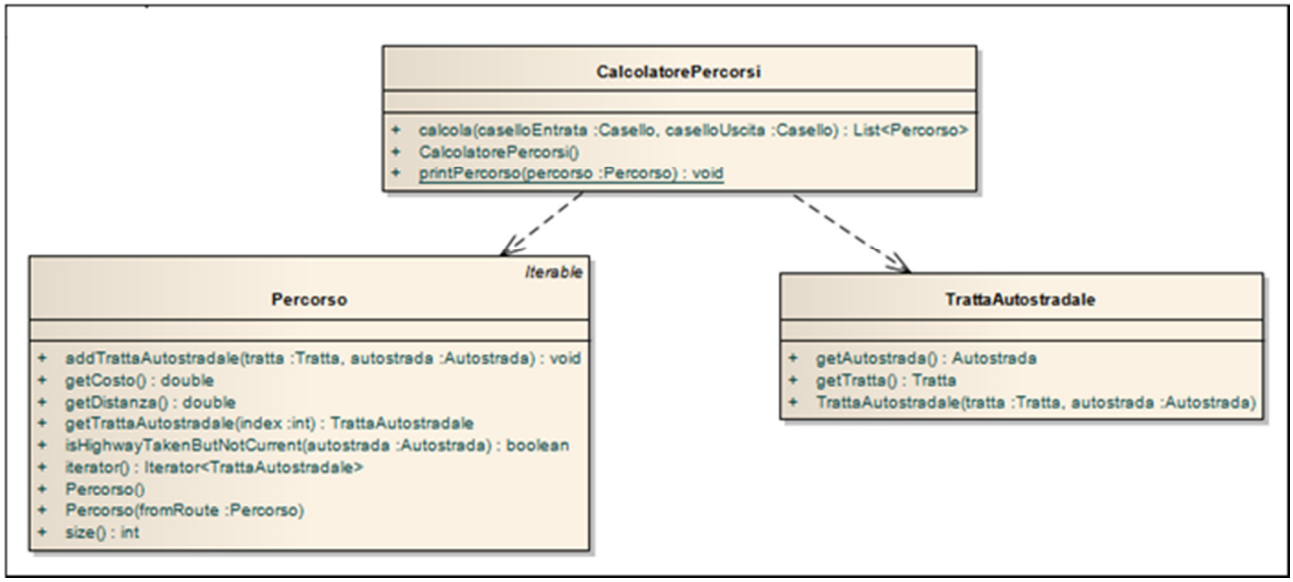
Il metodo ***toString*** deve essere ridefinito in modo da restituire l'elenco dei caselli separati da opportune virgole.

c) La classe ***Autostrada*** **(da realizzare)** rappresenta un'autostrada con la propria sigla univoca, fornita in fase di costruzione e mai più modificata, ed è caratterizzata da una sequenza di tratte. Offre i seguenti metodi:

- ***getNome*** per recuperare il nome dell'autostrada;
- ***addTratta*** per aggiungere una tratta all'autostrada (se non è già presente);
- ***getNext*** per ottenere la tratta successiva alla tratta data nella direzione indicata dal booleano ***ascending*** (se **true** restituisce la successiva, se **false** restituisce la precedente);
- ***getTratte*** restituisce l'elenco delle tratte.

Il metodo ***toString*** deve essere ridefinito in modo da restituire il nome dell'autostrada.

d) Le classi del package ***zannonia.model.routing*** (tutte fornite nello start kit) svolgono importanti servizi di utilità: in particolare, il metodo ***cercaPercorsi*** di ***CalcolatorePercorsi*** cerca tutti i percorsi fra due caselli della rete autostradale e restituisce un elenco ordinato di istanze di ***Percorso*** (anch'essa fornita nello start kit).



Persistenza (namespace zannonia.persistence) *(punti: 10)*

Come già anticipato, il file di testo Autostrade.txt contiene la descrizione delle diverse autostrade, una per riga: ogni riga contiene l'elenco (ordinato) delle tratte che la compongono (identificatori privi di spazi), separate da spazi.

A loro volta, le tratte sono descritte nel file di testo Tratte.txt, una per riga: i vari elementi sono separati uno dall'altro da spazi e/o tabulazioni, *fatta eccezione per l'elenco dei caselli,* i cui nomi possono contenere spazi e sono quindi separati fra loro da virgole. Nel dettaglio, ogni riga contiene nell'ordine:

- l'identificativo univoco della tratta
- la lunghezza in km della tratta
- il pedaggio in Euro della tratta
- l'elenco dei caselli appartenenti alla tratta, separati da virgole (può mancare se non ci sono caselli)

ESEMPIO DEL FILE autostrade.txt

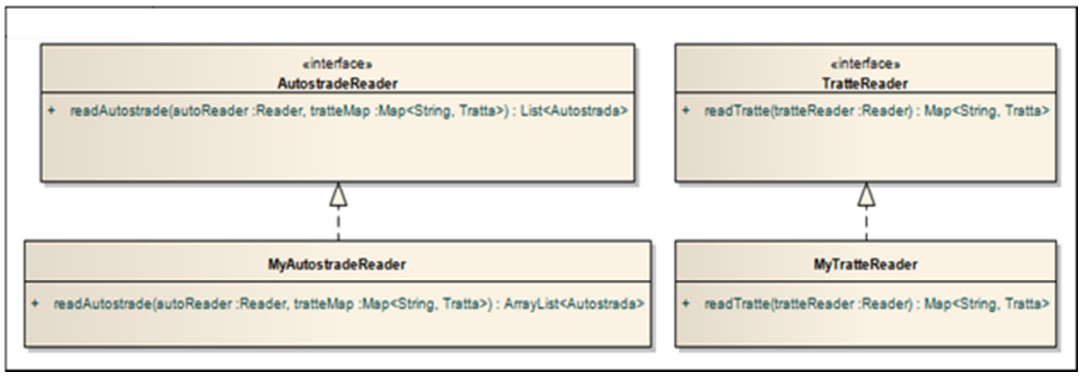
```
Z1  pc pc-pr pr-mo mo mo-bo bo bo-fi fi
Z14 bo bo-rn
Z22 mo mo-vr vr vr-tn tn-bz- bz-bre
Z4E vr vr-pd pd pd-ve
Z13 pd pd-ro ro-bo bo
...
```

ESEMPIO DEL FILE tratte.txt

```
pc      11  0.60   Piacenza nord, Piacenza sud
pc-pr   53  3.50   Fiorenzuola, Fidenza, Parma
pr-mo   48  3.10   Canossa, Reggio Emilia, Modena Nord
mo       2  0.00
mo-bo   31  2.40   Modena Sud, Crespellano, Bologna Ovest
bo       5  0.40   Bologna Ovest, Bologna Sud, Bologna Est, Bologna Nord
bo-fi   97  7.20   Sasso Marconi, Rioveggio, Roncobilaccio, Barberino
fi      28  1.80   Firenze Nord, Firenze Scandicci, Firenze Certosa, Firenze Sud
bo-rn   104 6.70   Imola, Faenza, Forlì, Cesena, Rimini
mo-vr   97  5.80   Campogalliano, Carpi, Reggiolo, Mantova, Nogarole Rocca
vr      6   0.70   Verona Sud, Verona Est, Verona Nord
vr-tn   87  6.00   Affi, Rovereto Sud, Rovereto Nord, Trento Sud, Trento Centro
tn-bz   65  4.60   Trento Nord, San Michele, Egna-Ora, Bolzano Sud
bz-bre  78  5.60   Bolzano Nord, Chiusa, Bressanone, Vipiteno, Confine di Stato
vr-pd   75  4.60   Soave, Montebello, Montecchio, Vicenza, Padova Ovest
pd      12  0.80   Padova Sud, Padova ZI, Padova est
pd-ve   42  3.70   Dolo, Mira, Mestre, Venezia
pd-ro   36  2.30   Terme Euganee, Monselice, Boara, Rovigo
ro-bo   77  4.90   Rovigo Sud, Occhiobello, Ferrara, Ferrara Sud, Altedo
```

a) L'interfaccia **TratteReader** (fornita nello start kit) dichiara il metodo **readTratte** che, dato un **Reader**, legge e restituisce una mappa la cui chiave è il nome trasformato in maiuscolo di una tratta e il cui valore è la tratta stessa (NB: la trasformazione in maiuscolo serve ovviamente a rendere l'accesso alla mappa insensibile all'uso di maiuscole o minuscole). La classe ***MyTratteReader*** (**da realizzare**) implementa l'interfaccia **TratteReader** in modo da leggere il formato sopra specificato: in caso di errore nel formato del file, il metodo di lettura deve lanciare una ***MalformedFileException*** come dichiarato nell'interfaccia.

b) L'interfaccia **AutostradeReader** (fornita) dichiara il metodo **readAutostrade** che, dato un **Reader** e la mappa delle tratte spiegata sopra, legge e restituisce una lista di autostrade. La classe ***MyAutostradeReader*** (**da realizzare**) implementa l'interfaccia **AutostradeReader** in modo da leggere il formato sopra specificato (durante la ricerca delle tratte nella mappa, avere cura di trasformare in maiuscolo la chiave di ricerca – v. sopra). In caso di errore nel formato del file, il metodo di lettura deve lanciare una opportuna ***MalformedFileException***.



Parte 2 (punti: 10)

Controller (namespace zannonia.controller)

La classe ***MainController*** è già fornita nello start kit: **non vi è quindi nulla da realizzare**. A titolo informativo:

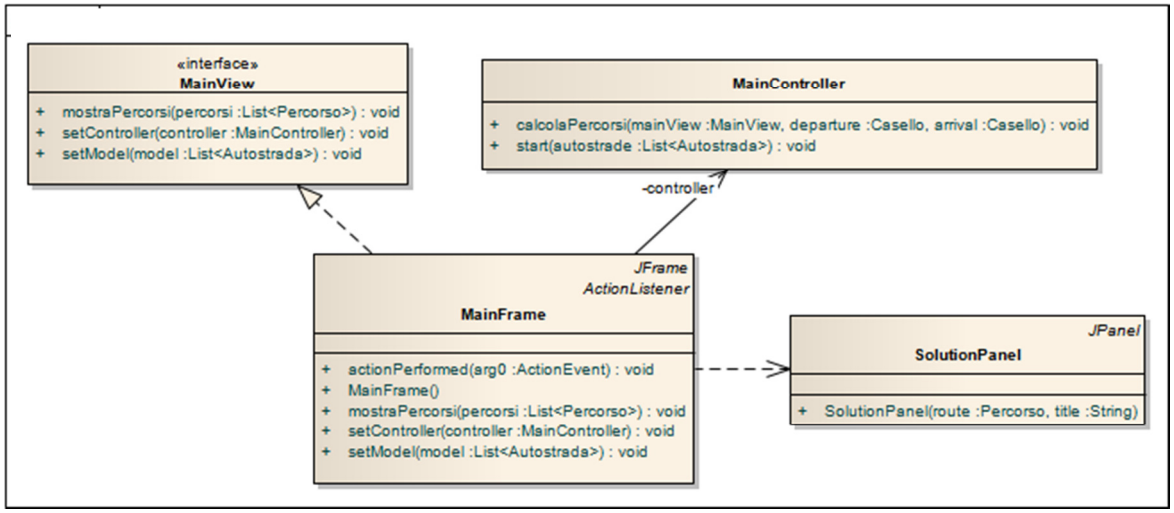
a) il costruttore riceve in ingresso una lista di autostrade

b) il metodo ***start*** fa partire l'applicazione

c) il metodo ***calcolaPercorsi*** prende in ingresso una ***MainView*** (v. dopo), un casello di entrata e un casello di uscita, e calcola il percorso fra tali caselli, popolando opportunamente la vista controllata.

GUI (namespace zannonia.ui) *(punti: 10)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato in figura. La classe ***Program*** (fornita ma **non mostrata** nel diagramma UML) contiene il *main* di partenza dell'intera applicazione.



La classe ***MainFrame* (da realizzare)** implementa l'interfaccia ***MainView*** e realizza la finestra principale, che deve consentire all'utente di selezionare i caselli di entrata e di uscita: a tal fine prevede due coppie di combobox (una per la scelta dell'autostrada, una per la scelta del casello in quell'autostrada), una per l'entrata e una per l'uscita.

Chiaramente, le due combo con l'elenco dei caselli devono essere (ri)popolate ogni volta che si sceglie una diversa autostrada. A tal fine, si possono sfruttare i seguenti metodi:

- ***setController*** per impostare il ***MainController***;
- ***setModel*** per popolare le combo, a partire da una lista di autostrade;
- ***mostraPercorsi*** per visualizzare i percorsi ricevuti in ingresso; questo metodo riceve una lista di percorsi e crea per ciascuno un nuovo ***SolutionPanel*** (fornito nello start kit) che mostra uno specifico percorso: a tal fine, il costruttore di ***SolutionPanel*** riceve in ingresso un ***Percorso*** e una stringa da mostrare nell'intestazione del pannello, nel formato "Soluzione *N*" (essendo *N* un numero progressivo). I pannelli così costruiti dovranno essere aggiunti alla finestra principale, ovviamente eliminando prima gli eventuali ***SolutionPanel*** precedenti (tramite il metodo ~~***clearAll***~~ di ***Panel***); dopo l'operazione occorre invocare i metodi ***repaint*** e ***validate*** sulla finestra principale per forzare il ridisegno.

Il pulsante ***CERCA PERCORSI*** scatena la ricerca percorsi fra i caselli selezionati utilizzando il metodo ***calcolaPercorsi*** di ***MainController***: esso provvede già anche all'aggiornamento della finestra principale chiamando al proprio interno il metodo ***mostraPercorsi*** di ***MainView***.