ESAME DI FONDAMENTI DI INFORMATICA T-2 del 17/01/2012
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)

Una concessionaria autostradale ha richiesto lo sviluppo di un'applicazione per il rispetto dei limiti di velocità. Tale sistema, denominato *Raptor,* è basato su un insieme di telecamere fisse installate su portali dell'autostrada, capaci di leggere le targhe dei veicoli in transito.

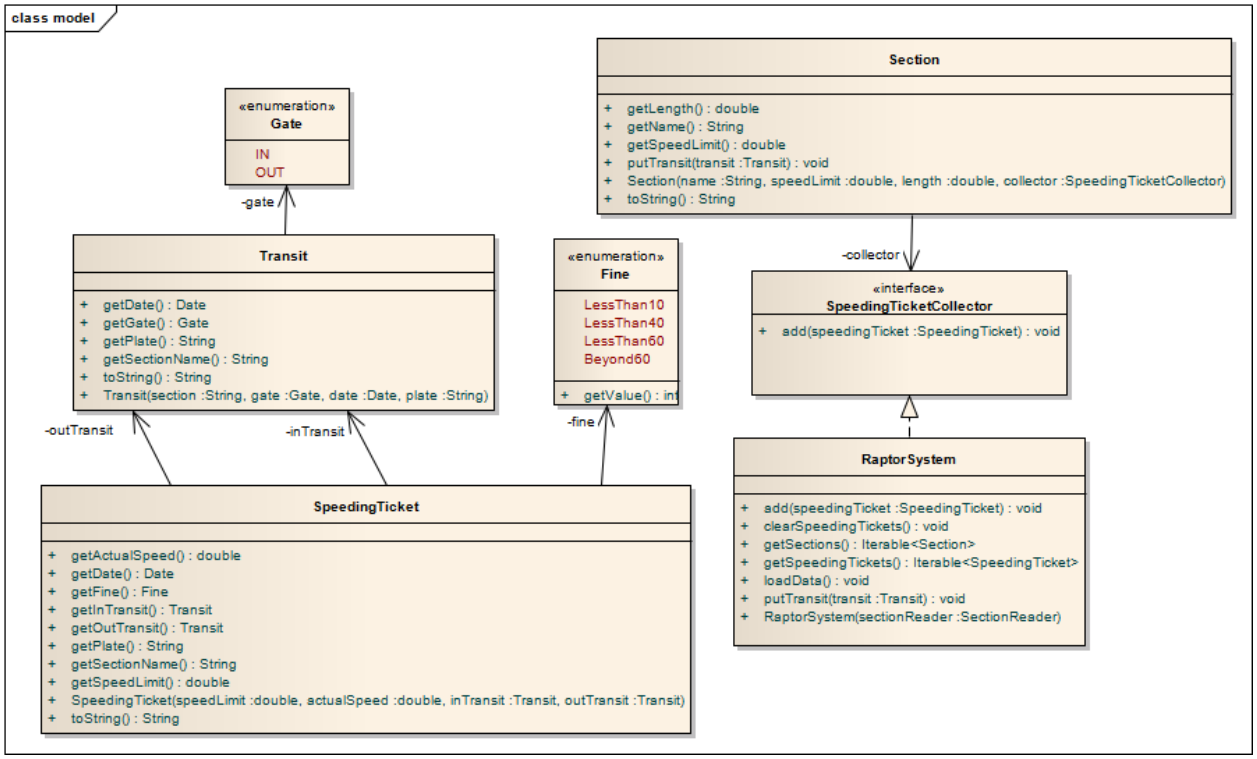
DESCRIZIONE DEL DOMINIO DEL PROBLEMA.
Il sistema *Raptor* è fisicamente costituito da un insieme di telecamere fisse installate su portali dell'autostrada, capaci di leggere le targhe dei veicoli in transito. Ogni sezione di autostrada sottoposta a controllo (*Section)* è caratterizzata da una sigla univoca, dalla sua lunghezza e dalla velocità massima consentita. I due portali alle estremità della sezione sono convenzionalmente denominati "IN" e "OUT", rispettivamente per l'entrata e l'uscita dalla sezione.
Quando un veicolo risulta aver percorso una sezione a una velocità media superiore a quella ammessa, si produce una multa per eccesso di velocità (*SpeedingTicket*).

I file di testo sections.txt e transits.txt contengono rispettivamente le sezioni sottoposte a controllo e i relativi transiti con le rispettive proprietà, nel formato più oltre specificato.

Parte 1 (punti: 19)

Dati (namespace raptor.model) *(punti: 13)*

Il modello dei dati deve essere organizzato secondo il diagramma UML di seguito riportato.



SEMANTICA:

a) l'enumerativo ***Gate*** (fornito nello start kit) rappresenta i portali: i valori possibili sono IN (per il portale d'ingresso) e OUT (per il portale d'uscita);

b) la classe ***Transit*** (fornita nello start kit) rappresenta il passaggio di un veicolo di targa nota sotto un portale di osservazione: è caratterizzato da data/ora, targa del veicolo (*plate*), portale interessato e fornisce i necessari metodi accessor;

c) la classe ***Section*** **(da realizzare)** rappresenta la sezione autostradale sottoposta a controllo, caratterizzata da un nome univoco, dalla velocità massima permessa e dalla lunghezza della sezione stessa. Il costruttore riceve quindi tali argomenti, nonché un'istanza di ***SpeedingTicketCollector*** – un oggetto che rappresenta il deposito delle violazioni accertate, ossia il luogo in cui devono essere archiviate le multe da emettere. Oltre ad alcuni ovvi metodi accessor e a un opportuno metodo ***toString***, ***Section*** prevede il metodo ***putTransit*** che incapsula l'algoritmo di controllo vero e proprio e agisce come segue:

- se il transito passato come argomento riguarda il primo portale (IN – transito di ingresso nella sezione), memorizza tale transito in una lista di transiti in ingresso;
- se, invece, il transito passato come argomento riguarda il secondo portale (OUT – transito di uscita dalla sezione), recupera ed elimina dalla lista il corrispondente transito dello stesso veicolo sul primo portale (transito in ingresso) mediante match sulla targa, calcola la velocità media del veicolo, la confronta col limite vigente e, in caso di violazione, aggiunge allo ***SpeedingTicketCollector*** una nuova multa (opportuna istanza di ***SpeedingTicket***) con tutti i dettagli necessari.

d) la classe ***SpeedingTicket*** **(da realizzare)** rappresenta la multa per eccesso di velocità, ed è caratterizzata dalla serie di proprietà dettagliate nel diagramma UML, con i relativi metodi accessor (di sola lettura) e a un opportuno metodo ***toString***. Essa si appoggia all'enumerativo ***Fine*** più sotto dettagliato. *È compito del costruttore di SpendingTicket calcolare la fascia (e dunque l'importo) della sanzione in base all'eccesso di velocità registrato*. La data/ora della sanzione è la stessa del transito in uscita.

e) l'interfaccia ***SpeedingTicketCollector*** (fornita nello start kit) modella il tipo "deposito di multe";

f) l'enumerativo ***Fine*** (**da realizzare**) rappresenta un tipo a quattro valori, corrispondenti alle quattro fasce di violazione (si veda il diagramma UML per i dettagli). A ogni valore dell'enumerativo dev'essere associato il valore della corrispondente multa (recuperabile tramite il metodo ***getValue***) secondo il seguente schema:

- eccesso di velocità di non oltre 10 km/h: € 40
- eccesso di velocità fra 11 e 40 km/h: € 155
- eccesso di velocità fra 41 e 60 km/h: € 500
- eccesso di velocità oltre i 60 km/h: € 780

g) la classe ***RaptorSystem*** (fornita nello start kit) definisce e gestisce le due strutture dati fondamentali per il funzionamento dell'intero sistema (con ciò costituendo anche un'implementazione di ***SpeedingTicketCollector***):

- un elenco di ***Section*** per rappresentare le sezioni attive;
- un elenco di ***SpeedingTicket*** per mantenere le multe da emettere.

Questa classe definisce:

- il costruttore e tutti i necessari metodi accessor
- un metodo ***loadData*** che, utilizzando il ***SectionReader*** ricevuto in ingresso dal costruttore, carica tutte le ***Section*** dal file sections.txt.
- un suo metodo ***putTransit*** (diverso da quello di ***Section***) per aggiungere un transito a una ***Section***

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Persistenza (namespace raptor.persistence) *(punti 6)*

Come già anticipato, il file di testo sections.txt contiene l'elenco delle sezioni sottoposte a controllo, una per riga: ogni riga contiene perciò, in questo ordine, l'identificatore univoco della sezione, la lunghezza in km della sezione, la velocità massima ammessa in km/h, separati fra loro da spazi.

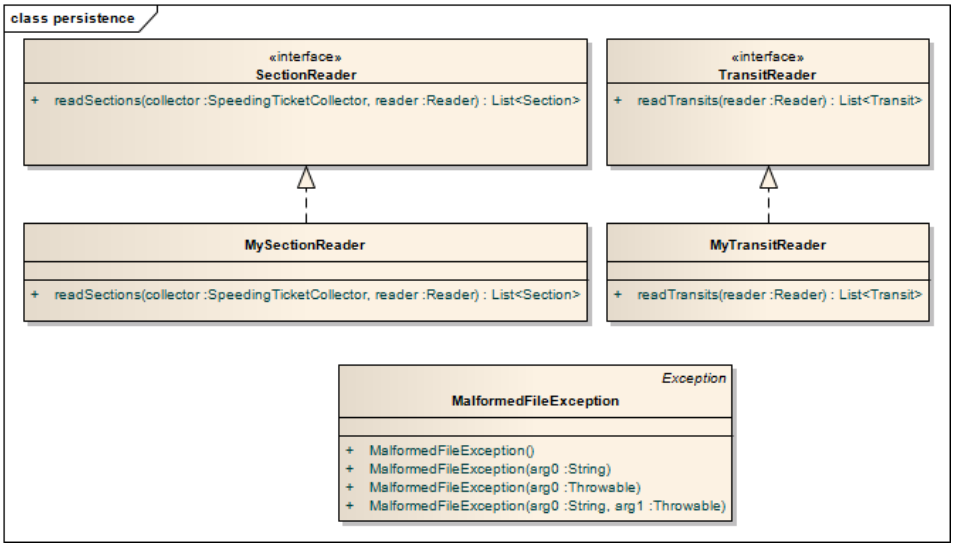
```
ESEMPIO DEL FILE sections.txt
A14-BO1-N 5 130
A14-BO1-S 6.5 110
...
```

Un secondo file di testo, transits.txt, contiene i transiti registrati: ogni riga rappresenta un transito e contiene nell'ordine l'identificativo univoco della sezione, il portale di sezione attraversato dal veicolo (IN/OUT), la targa del veicolo, la data (nel formato GG/MM/AAAA) e ora (nel formato HH.MM.SS) del passaggio, separate da uno spazio.

```
ESEMPIO DEL FILE transits.txt
A14-BO1-N IN ED999AP 10/01/2012 12.34.56
A14-BO1-N IN MP333BO 10/01/2012 12.34.57
A14-BO1-N OUT MP333BO 10/01/2012 12.36.55
A14-BO1-N OUT ED999AP 10/01/2012 12.37.31
```

Le interfacce ***SectionReader*** e ***TransitReader*** (fornite) dichiarano rispettivamente i metodi ***readSections*** e ***readTransits*** che leggono da un ***Reader*** rispettivamente una lista di ***Section*** e di ***Transit***.

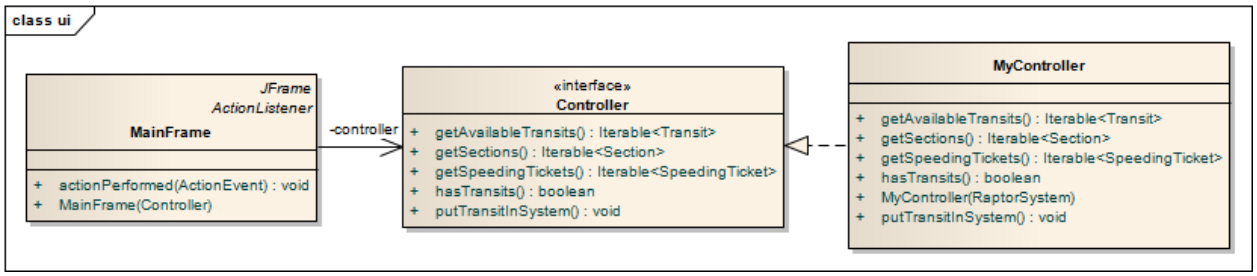
Le classi ***MySectionReader*** e ***MyTransitReader*** (**da realizzare**) implementano tali interfacce.



Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Parte 2 (punti: 11)

Controller (namespace raptor.ui) *(punti 5)*



La classe ***MyController*** (**da realizzare**) implementa l'interfaccia ***Controller*** (fornita): in particolare il costruttore riceve in ingresso un'istanza di ***RaptorSystem***, e carica il file dei transiti. Due metodi (***getSections*** e ***getSpeedingTickets***) delegano al ***RaptorSystem*** interno l'esecuzione dei loro compiti; il metodo ***putTransitInSystem*** estrae e rimuove il primo transito della collezione, e lo inserisce nel suo ***RaptorSystem***; il metodo ***getAvailableTransits*** restituisce la collezione dei transiti caricati dal costruttore; infine, il metodo ***hasTransits*** indica se vi sono ancora transiti da elaborare.

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Interfaccia utente (namespace raptor.ui) *(punti 6)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato in figura.

La classe ***GuiMain*** (non mostrata nel diagramma UML, ma fornita nello start kit) contiene il main di partenza dell'intera applicazione.

La classe ***MainFrame*** (**da realizzare**) realizza la finestra principale, ed è articolata in quattro parti principali – tre aree di testo e un pulsante. Una prima area in alto contiene l'elenco delle sezioni sotto controllo, una seconda area subito sotto mostra i transiti registrati, un pulsante centrale ELABORA che consente di analizzare i transiti uno ad uno, e un'area in basso in cui mostrare le violazioni.

Inizialmente (Fig. 1) le due aree superiori sono popolate coi dati letti dai rispettivi file, e l'area inferiore è vuota. Successivamente, ogni volta che si preme il pulsante viene analizzato un nuovo transito, che viene rimosso dall'area centrale (Fig. 2): se il transito riguarda un passaggio da un portale IN l'elaborazione prosegue internamente ma non viene mostrato nulla, mentre se riguarda un passaggio da un portale OUT si verifica la regolarità o meno del transito, mostrando in caso di irregolarità il messaggio di violazione nell'area in basso (Figg. 3 e 4). Quando tutti i transiti sono stati elaborati, il pulsante ELABORA dev'essere disabilitato (Fig. 4).

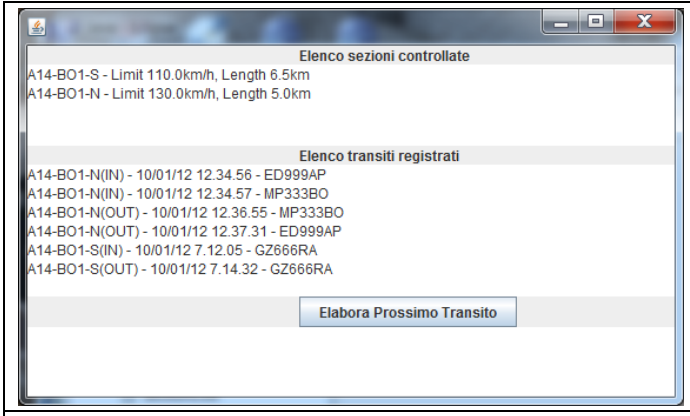


Fig. 1

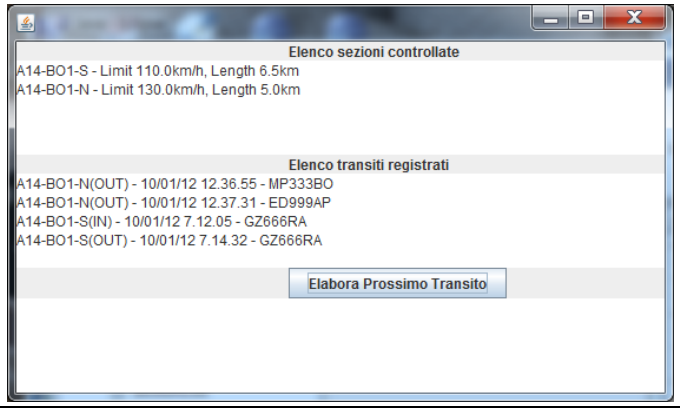


Fig. 2

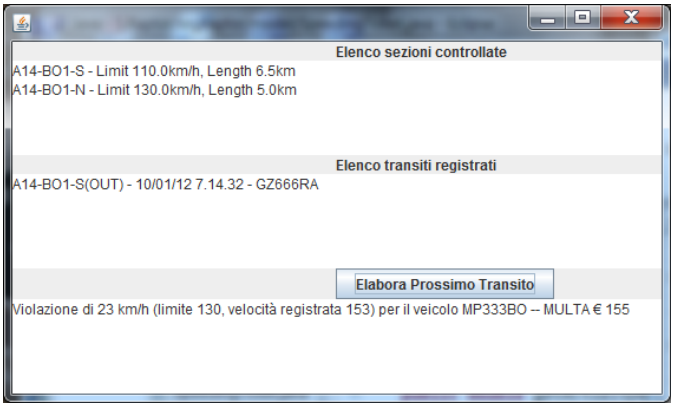


Fig. 3

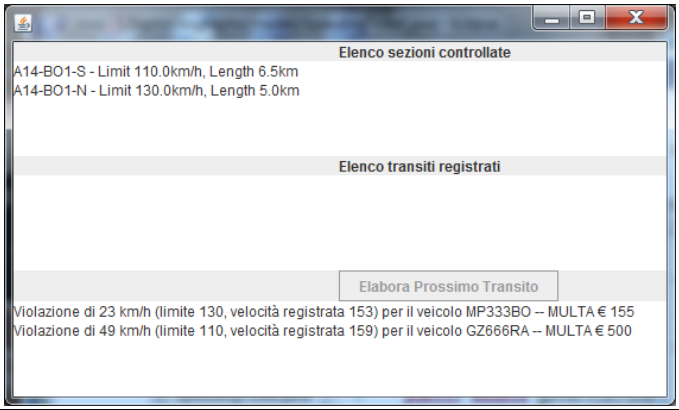


Fig. 4