ESAME DI FONDAMENTI DI INFORMATICA T-2 del 3/7/2024
Proff. E. Denti – R. Calegari – A. Molesini
Tempo a disposizione: 3h30

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)
NOME CARTELLA PROGETTO: CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE: CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)
NOME JAR DA CONSEGNARE: CognomeNome-matricola.jar (es. RossiMario-0000123456.jar)

Si devono consegnare DUE FILE: *l'intero progetto Eclipse* e *il JAR eseguibile*

Si ricorda che compiti *non compilabili*, o che *non passino almeno 2/3 dei test* o siano *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO".

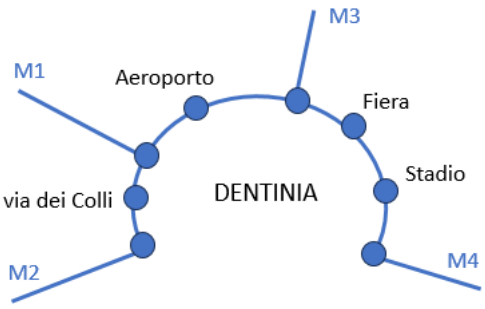
La rete autostradale di Dentinia è articolata in *quattro autostrade a pedaggio* che originano a raggiera dalla *tangenziale*, che avvolge buona parte la città (v. figura), senza comunque richiudersi su sé stessa. La tangenziale è utilizzata 1) dal traffico cittadino, che entra/esce dai caselli della tangenziale senza però utilizzare le autostrade; 2) dal traffico di puro transito fra un'autostrada e l'altra, che la usa solo come connessione fra tronchi autostradali; 3) dal traffico di entrata/uscita in/dalla città da/verso le autostrade, che esce/entra da un casello della tangenziale da/verso un'autostrada, dando luogo a un percorso misto.

Il comune di Dentinia ha ottenuto dal gestore autostradale che la tangenziale sia *gratuita per l'uso urbano* (caso 1) ma *a pagamento* per l'attraversamento (caso 2) e, entro i limiti sotto discussi, per l'uso ibrido (caso 3).

È quindi stato richiesto di sviluppare un'applicazione che consenta di determinare il percorso fra una qualsiasi coppia di caselli della rete, calcolandone il chilometraggio e il costo.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Come illustrato a lato, la rete autostradale di Dentinia è costituita da quattro autostrade (M1, M2, M3, M4): esse convergono tutte sulla tangenziale, che avvolge la città da sud-ovest a sud-est. Sulla tangenziale sono presenti quattro caselli urbani (via dei Colli, Aeroporto, Fiera, Stadio) oltre, ovviamente, ai quattro allacciamenti per le quattro autostrade. Ogni autostrada prevede ovviamente una serie di caselli (non mostrati in figura).



Tutte le autostrade sono a pedaggio, mentre per la tangenziale vige un sistema misto, discusso in dettaglio sotto; in ogni caso, il pedaggio si calcola moltiplicando i *chilometri utili* per il costo chilometrico unitario (noto).

Per quanto riguarda la tangenziale, valgono le seguenti regole:

1. L'uso esclusivamente urbano è sempre gratuito: quindi, entrando/uscendo dai quattro caselli urbani, i km utili ai fini del pedaggio sono sempre zero.
2. L'uso come pura connessione fra autostrade è sempre a pagamento: in tal caso, quindi, i km compresi fra i due allacciamenti autostradali sono sempre tutti utili ai fini del pedaggio.
3. Nel caso ibrido i km percorsi in tangenziale sono a pagamento solo per i percorsi in cui la quota autostradale superi i 40 km: ciò consente di non penalizzare i pendolari che vivano entro i 40 km dalla città, per i quali il tratto in tangenziale resta quindi gratuito.

ESEMPI: entrando a via dei Colli e uscendo a Stadio, non si paga nulla; se, invece, da Fiera si va verso M1, il tratto in tangenziale fino all'allacciamento con la M1 è a pagamento solo se i km su M1 sono più di 40, altrimenti è gratuito; infine, se si proviene da M2 e si va verso M3, il tratto di tangenziale fra i due allacciamenti è sempre a pagamento.

Il file di testo AutostradeDentinia.txt, nel formato descritto più oltre, contiene la descrizione delle varie autostrade e della tangenziale. Per ipotesi, nella rete autostradale esiste sempre una e una sola "Tangenziale".

TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO: 2h15 – 3h

PARTE 1 – Modello dei dati: Punti 13 [TEMPO STIMATO: 70-90 minuti]

PARTE 2 – Persistenza: Punti 11 [TEMPO STIMATO: 45-60 minuti]

PARTE 3 – Grafica: Punti 6 [TEMPO STIMATO: 20-30 minuti]

NUMERO MINIMO DI TEST CON SUCCESSO PERCHÉ IL COMPITO SIA CORRETTO

Considerando solo Percorso, MyPlanner e MyAutostradeReader	20 su 29
Considerandoli tutti:	32 su 48

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-21.0.2\lib"
--add-modules javafx.controls
```

Cose da ricordare

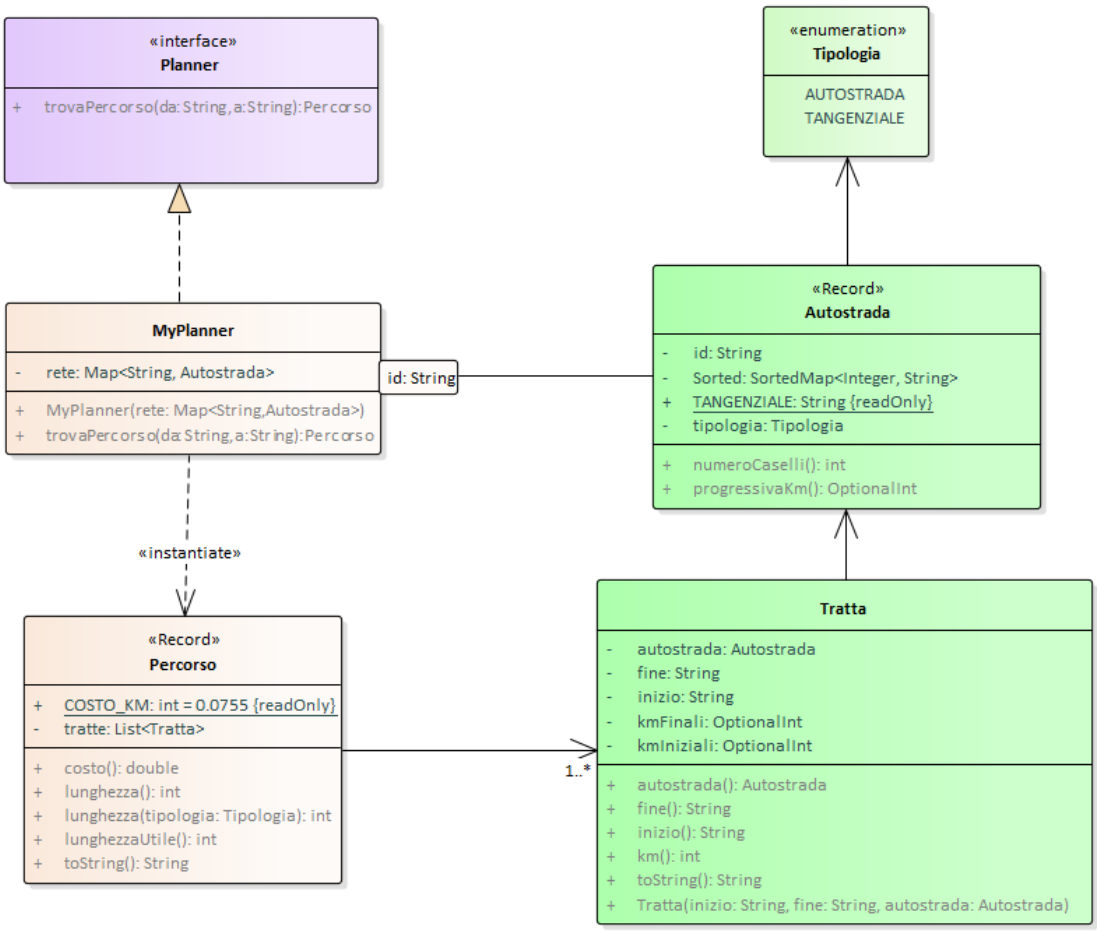
- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente come richiesto?
- Hai fatto un **unico file ZIP (NON .7z, rar o altri formati)** contenente l'intero progetto? In particolare, ti sei assicurato di aver incluso tutti i file .java (e non solo i .class)?
- **Hai consegnato DUE file distinti, ossia lo ZIP col progetto e il JAR eseguibile?**
- Su EOL, hai **premuto** il tasto "CONFERMA" per inviare il tuo elaborato?

Parte 1 – Modello dei dati (punti: 13)

Package: tangenziale.model **[TEMPO STIMATO: 70-90 minuti]**



SEMANTICA:

- L'enumerativo ***Tipologia*** (fornito) definisce semplicemente le due costanti AUTOSTRADA e TANGENZIALE
- Il record ***Autostrada*** (fornito) rappresenta un'autostrada (o tangenziale), caratterizzata da ***Tipologia,*** identificativo univoco (stringa) e una mappa <intero, stringa> che elenca i caselli e gli allacciamenti, associando ogni casello (identificato dal suo nome) alla progressiva chilometrica lungo l'autostrada stessa.
  - Il costruttore verifica che la mappa ricevuta non sia null né vuota (lanciando rispettivamente ***NullPointerException*** o ***IllegalArgumentException*** in caso contrario) e che non ci siano né due caselli omonimi, né due caselli definiti alla stessa progressiva chilometrica (lanciando anche in questo caso ***IllegalArgumentException*** in caso di violazione)
  - il metodo *numeroCaselli* restituisce il numero dei caselli/allacciamenti previsti in questa autostrada (cioè il numero di righe della mappa)
  - il metodo *getProgressivaKm(String nomeCasello)* restituisce, sotto forma di ***OptionalInt***, la progressiva chilometrica del casello o allacciamento specificato: se esso non esiste, restituisce ***OptionalInt.empty***
  - è inoltre definita la costante TANGENZIALE, che esprime il nome della (unica) tangenziale
- La classe ***Tratta*** (fornita) rappresenta una tratta autostradale su una specifica autostrada o tangenziale, compresa fra due caselli di inizio e fine (due stringhe).
  - Il costruttore verifica che gli argomenti non sia null né vuoti (lanciando rispettivamente ***NullPointerException*** o ***IllegalArgumentException*** in caso contrario)
  - una serie di accessor consente di recuperare gli argomenti passati al costruttore

- il metodo *km* restituisce la lunghezza della tratta
- un'apposita *toString* emette una rappresentazione stampabile della tratta stessa

- La classe ***Percorso*** (da completare nella parte algoritmica) esprime l'idea di percorso come sequenza di tratte e ne cattura le proprietà salienti. In particolare:
  - è definita la **costante pubblica COSTO_KM** che esprime il costo al chilometro del pedaggio sull'intera rete autostradale
  - il costruttore riceve la lista di tratte e verifica che non sia nulla, emettendo in caso contrario la classica ***NullPointerException***; parimenti non è valido un percorso con più di tre tratte o che abbia due tratte sulla stessa autostrada: in questi casi viene lanciata ***IllegalArgumentExcception***. È invece lecito il caso di lista vuota, che corrisponde a un percorso non trovato.
  - il metodo *lunghezza* restituisce la lunghezza totale del percorso
  - il metodo *lunghezza(Tipologia)* restituisce la lunghezza delle sole tratte della tipologia specificata
  - il metodo *costo* restituisce il costo (pedaggio) del percorso, ottenuto moltiplicando il costo chilometrico per i km utili
  - il metodo ***lunghezzaUtile*** *(da realizzare)* incorpora la logica di calcolo dei km utili ai fini del pedaggio, così come definita dal Dominio del Problema:
    1. l'uso urbano della Tangenziale è sempre gratuito: quindi, entrando/uscendo dai suoi quattro caselli urbani, i km utili ai fini del pedaggio sono sempre zero;
    2. l'uso della Tangenziale come pura connessione fra autostrade è sempre a pagamento;
    3. nel caso ibrido, i km percorsi in tangenziale sono a pagamento solo per i percorsi in cui la quota autostradale superi i 40 km.
- L'interfaccia ***Planner*** (fornita) rappresenta la vista esterna del pianificatore percorsi: essa dichiara il metodo *trovaPercorso(String da, String a)*, che calcola e restituisce il ***Percorso*** fra i due caselli o allacciamenti specificati secondo le specifiche e la tariffazione descritte nel Dominio del Problema. Si ricorda che:
  - se i caselli di entrata e di uscita sono sulla stessa autostrada (o tangenziale), il percorso è diretto (cioè, è composto da un'unica tratta);
  - se i caselli di entrata e di uscita sono su due autostrade diverse, il percorso è indiretto e comprende tre tratte, di cui senz'altro anche un tratto intermedio in Tangenziale;
  - se i caselli di entrata e di uscita sono uno su un'autostrada e l'altro sulla Tangenziale (o viceversa), il percorso è indiretto e composto di due tratte, una delle quali è senz'altro in Tangenziale.
- La classe ***MyPlanner*** *(da realizzare)* implementa il pianificatore sopra descritto. In particolare:
  - il costruttore riceve la mappa della rete autostradale, che non può essere né nulla, né vuota (altrimenti, viene lanciata rispettivamente ***NullPointerException*** o ***IllegalArgumentException***) e deve necessariamente contenere almeno la tangenziale (altrimenti, ***IllegalArgumentException***)
  - il metodo *trovaPercorso(String da, String a)* deve verificare accuratamente le precondizioni:
    - gli argomenti non possono essere nulli, altrimenti ***NullPointerException***
    - i caselli dati devono esistere nella rete autostradale, altrimenti ***IllegalArgumentException***

SEGUE PAGINA SUCCESSIVA

Parte 2 – Persistenza (punti: 11)

Package: tangenziale.persistence **[TEMPO STIMATO: 45-60 minuti]**

Il file di testo AutostradeDentinia.txt contiene la descrizione delle varie autostrade e della tangenziale. Per ipotesi, deve esistere sempre una e una sola "Tangenziale". Il formato è descritto di seguito e illustrato nell'esempio in calce:

- Ogni autostrada (inclusa la tangenziale) è descritta da un blocco di righe, che inizia con una riga contenente solo il nome dell'autostrada e termina con una riga contenente la sola parola "END".
- Le righe intermedie descrivono i caselli e gli allacciamenti di tale autostrada, e prevedono prima la progressiva chilometrica (un intero), poi, separato da una o più tabulazione, il nome del casello o allacciamento (che può contenere spazi).

```
M3
45      Petruvia Nord
40      Petruvia Sud
25      Aldino
1       Dentinia Ippodromo
0       Tangenziale
END

M4
0       Tangenziale
24      Castelbello
37      Molla
50      Forte
77      Ripalta
END

...

Tangenziale
0       M2
4       via dei colli
9       M1
13      Aeroporto
19      M3
23      Fiera
26      Stadio
32      M4
END
```

SEMANTICA:

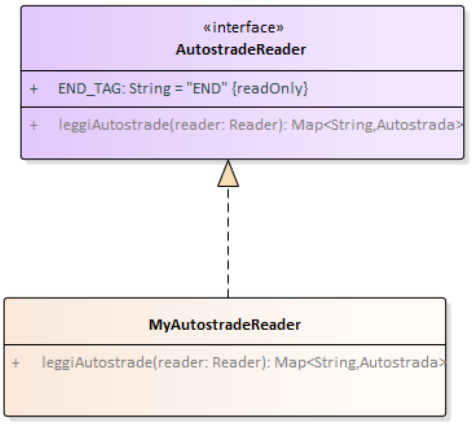
a) L'interfaccia ***AutostradeReader*** (fornita) dichiara:

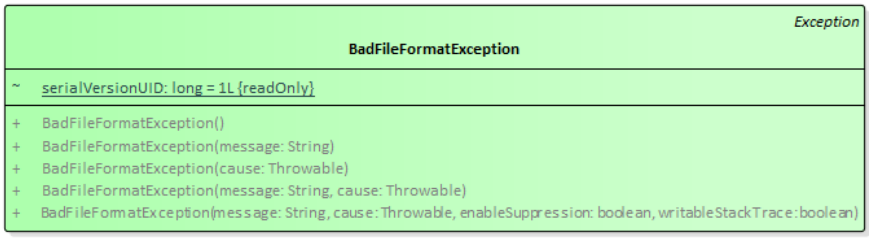
- il metodo *leggiAutostrade,* che legge dal ***Reader*** (già aperto) fornito i dati necessari e restitisce la mappa della rete autostradale, di tipo <***String***,***Autostrada***>, che associa ogni ***Autostrada*** al suo identificativo univoc, che funge da chiave; esso deve lanciare ***NullPointerException*** in caso di reader nullo, ***BadFile-FormatException*** con opportuno messaggio d'errore in caso di problemi di formato, o ***IOException*** in caso di problemi di I/O
- la costante stringa ***END_TAG*** che definisce la stringa corrispondente a "END" nel formato del file

b) La classe ***MyAutostradeReader*** (da completare) implementa ***AutostradeReader****:*

- non c'è alcun costruttore
- Il metodo *leggiAutostrade* legge le autostrade e le accumula via via nella mappa, che poi restituirà; esso deve effettuare due fondamentali controlli di consistenza:
  i. Uno su ogni singola autostrada, verificando che in ogni autostrada letta non vi siano due caselli o allacciamenti alla stessa progressiva chilometrica

ii. Un altro al termine della lettura, verificando che nella rete autostradale nel suo complesso ci sia sempre una e una sola tangenziale.

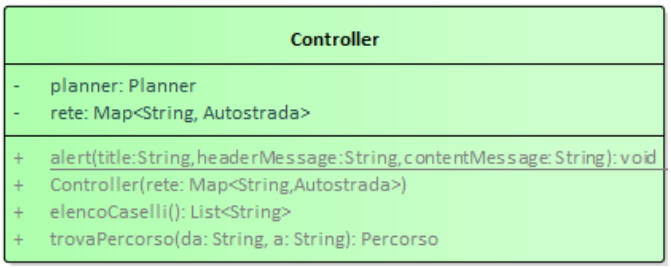




Parte 3 (punti: 6)

Package: tangenziale.controller *(punti 0)*

Il Controller funge semplicemente da front-end verso l'analizzatore, ed è quindi è organizzato come segue:



SEMANTICA:

La classe ***Controller*** (fornita) funge da ponte fra grafica e model e definisce i seguenti metodi:

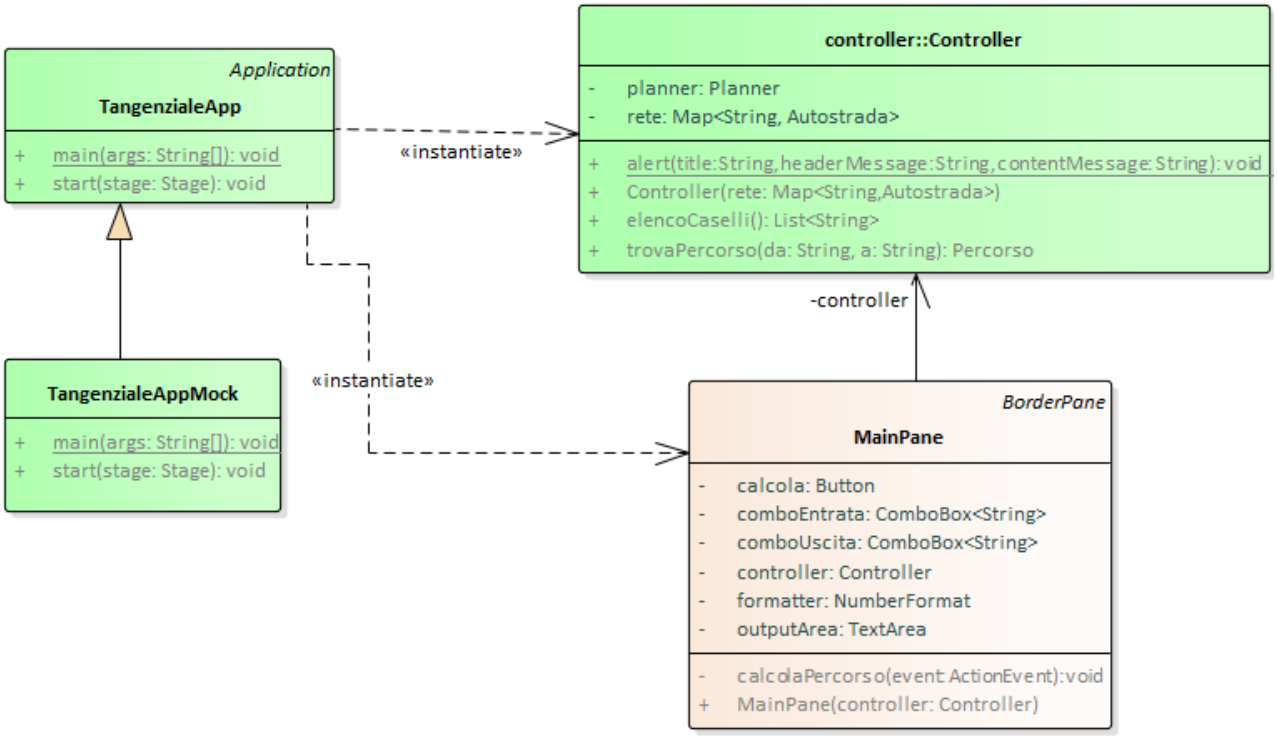
- Il costruttore riceve la mappa contenente la rete autostradale: verifica che essa non sia nulla né vuota, lanciando le opportune eccezioni in caso contrario; se tutto è ok, istanzia e mantiene al proprio interno il ***Planner*** che calcolerà i percorsi e i costi
- *trovaPercorso(String da, String a)* che banalmente delega all'omonimo metodo del ***Planner***
- *elencoCaselli* che restituisce la lista dei soli caselli, esclusi gli allacciamenti fra autostrade e tangenziale

Il metodo statico *alert*, utilizzabile dal ***MainPane*** quando è attiva la grafica, consente di far comparire all'utente una finestra di dialogo che mostri il messaggio d'errore specificato.

Package: tangenziale.ui **[TEMPO STIMATO: 20-30 minuti]** *(punti 6)*

La classe ***TangenzialeApp*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire i file, creare il controller e incorporare il ***MainPane***. Per consentire di collaudare la GUI anche in assenza / in caso di malfunzionamento della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***TangenzialeAppMock***

L'interfaccia utente è illustrata nelle figure seguenti e segue il modello sotto illustrato*:*



- <u>in alto</u>, due combo popolate con l'elenco dei caselli consentono di scegliere i caselli di entrata e di uscita
- <u>al centro</u>, il pulsante "Calcola percorso" attiva il planner: se uno o entrambi i caselli non sono selezionati, dev'essere mostrato (tramite il metodo *alert* del ***Controller***) un opportuno messaggio d'errore (Fig. 3)
- <u>in basso</u>, un'area di testo mostra il risultato, costituito dal dettaglio del percorso e dal relativo costo (pedaggio), formattato secondo lo standard italiano.

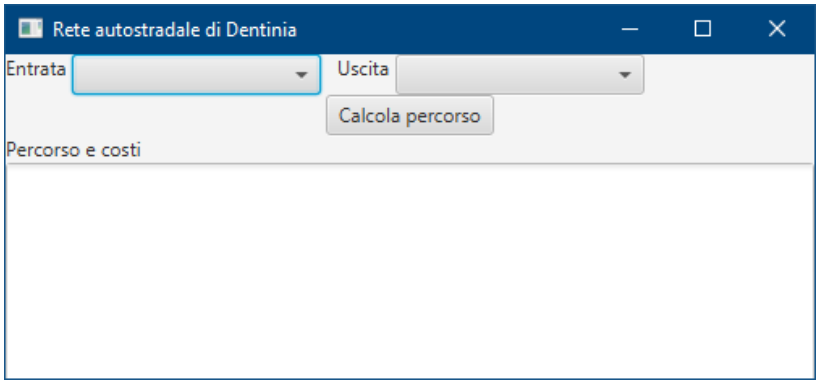


Fig. 1: vista generale della GUI: sopra le combo per scegliere i caselli di entrata e uscita, in mezzo il pulsante di attivazione del planner, in basso l'area destinata a mostrare i risultati.

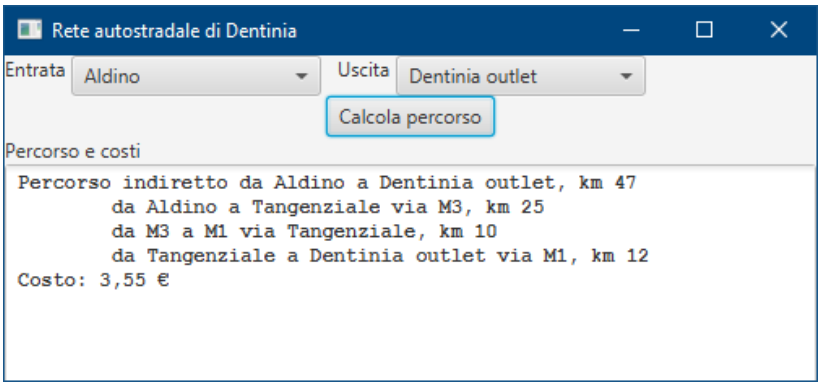


Fig. 2: la GUI dopo aver scelto due caselli e aver premuto il pulsante "Calcola percorso".

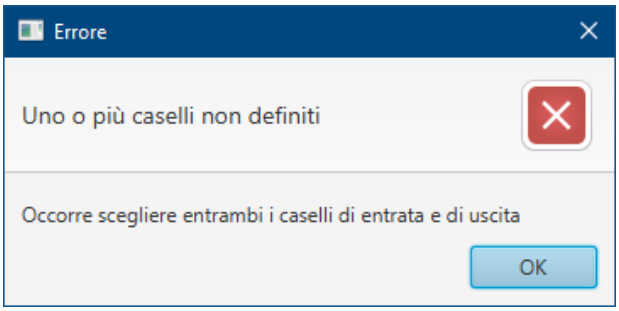


Fig. 3: il messaggio d'errore che compare se si preme il pulsante senza aver prima selezionato uno o entrambi i caselli di entrata e uscita.

Il MainPane è fornito *parzialmente realizzato*. Rimangono da realizzare le seguenti parti:

- il popolamento iniziale delle due combo
- l'aggancio del pulsante al gestore degli eventi (metodo privato *calcolaPercorso*)
- il gestore dell'evento *calcolaPercorso,* che deve innanzitutto verificare che entrambi i caselli di entrata e uscita siano selezionati: se così non è, deve emettere il messaggio d'errore sopra descritto e terminare senza fare nulla; se invece tutto è regolare, deve far calcolare il percorso al ***Controller*** ed emettere sull'area di testo il messaggio con il risultato, formattato come nelle figure, completo dell'indicazione di costo formattata secondo lo standard italiano

Cose da ricordare

- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"…
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente come richiesto?
- Hai fatto un **unico file ZIP (NON .7z, rar o altri formati)** contenente l'intero progetto? In particolare, ti sei assicurato di aver incluso tutti i file .java (e non solo i .class)?
- **Hai consegnato DUE file distinti, ossia lo ZIP col progetto e il JAR eseguibile?**
- Su EOL, hai **premuto** il tasto "CONFERMA" per inviare il tuo elaborato?