ESAME DI FONDAMENTI DI INFORMATICA T-2 del 16/07/2012
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)

Il *Gruppo Camminatori Appenninici* ha richiesto lo sviluppo di un'applicazione che aiuti i propri soci a comporre itinerari escursionistici in montagna. A tal fine l'applicazione attinge a un insieme di sentieri, con le relative caratteristiche: lo scopo è assistere l'utente nella composizione dell'itinerario, assicurandosi che esso rispetti alcuni vincoli (durata massima, lunghezza massima, dislivello massimo) e non ecceda le capacità fisiche della persona (difficoltà massima sopportabile); deve anche essere possibile stampare su un file gli itinerari (corretti) generati.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA.
Ogni *sentiero* è caratterizzato da un identificativo univoco, un punto di partenza con relativa altitudine, un punto di arrivo con relativa altitudine, la lunghezza (km), la difficoltà (scala da 1 a 4) e il tempo di percorrenza (ore).
Un *itinerario* è composto da una sequenza di sentieri, ognuno dei quali deve iniziare nel punto in cui termina il precedente, ed è caratterizzato da:

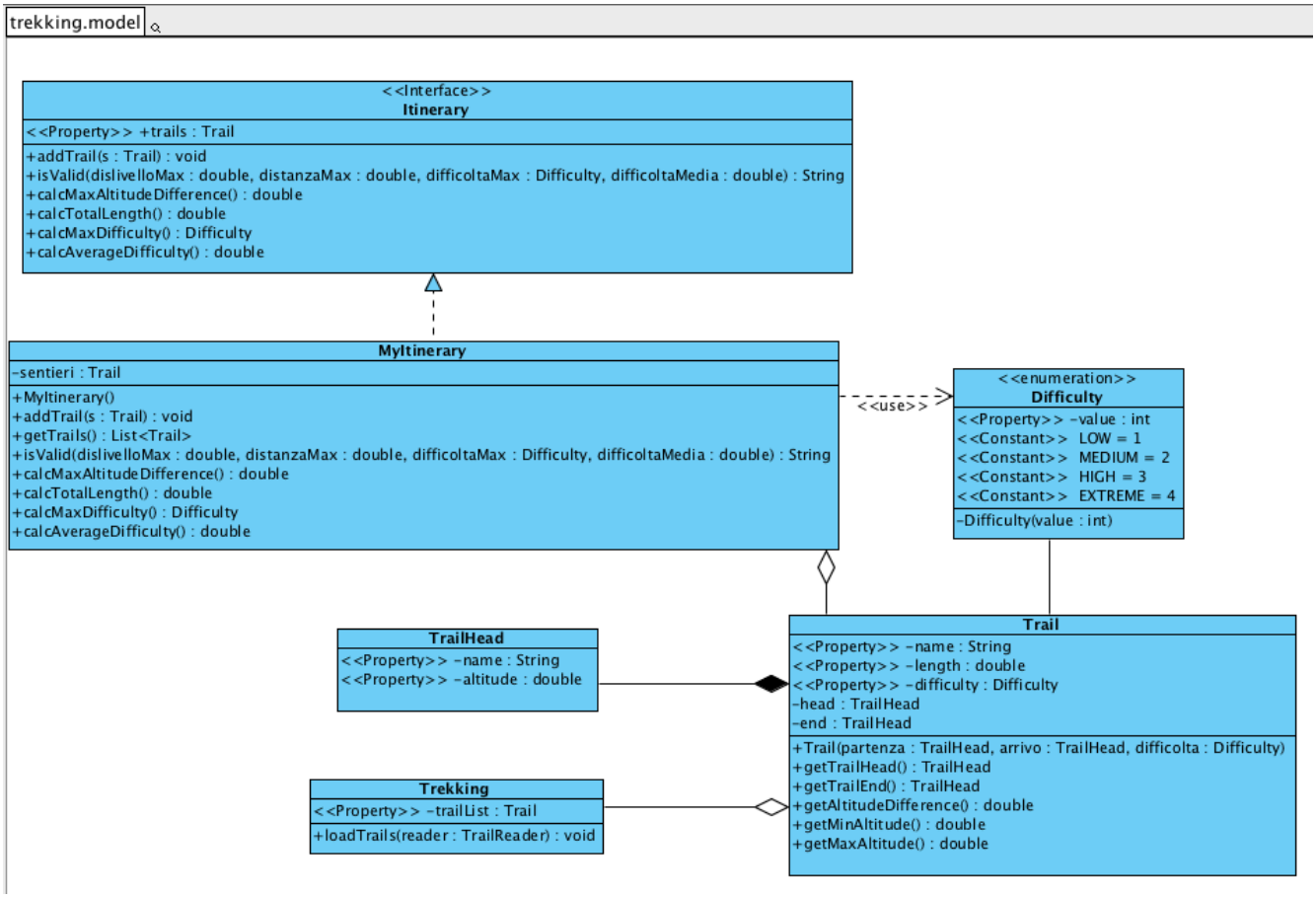
- un titolo (frase che può contenere spazi e ogni altro carattere)
- il luogo di partenza
- l'elenco dei sentieri da percorrere
- la difficoltà media, ottenuta componendo le difficoltà dei singoli sentieri che lo compongono (nel modo più oltre specificato)

Il file di testo Trails.txt contiene la descrizione dei diversi sentieri, nel formato più oltre specificato.

Parte 1 (punti: 16)

Dati (package trekking.model) *(punti: 9)*

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



SEMANTICA:

a) l'enumerativo ***Difficulty*** (fornito nello start kit) definisce i possibili livelli di difficoltà, da Low (1) a Extreme (4);

b) la classe ***Trail*** (fornita nello start kit) rappresenta il sentiero, caratterizzato delle proprietà di cui sopra, recuperabili tramite opportuni metodi accessor;

c) La classe ***TrailHead*** (fornita nello start kit) rappresenta l'estremità di un sentiero (ogni sentiero ha quindi due ***TrailHead***) ed è caratterizzata dalle proprietà nome e altezza, recuperabili tramite opportuni metodi accessor;

d) l'interfaccia ***Itinerary*** (fornita nello start kit) rappresenta la visione astratta di un itinerario e dichiara i metodi:

- ***addTrail***, per aggiungere un sentiero (in coda) all'itinerario; esso solleva ***IllegalArgumentException*** in caso di incoerenza, ossia se il sentiero da aggiungere non inizia dove termina il sentiero precedente;
- ***getTrails***, per recuperare la lista dei sentieri che costituiscono l'itinerario;
- ***isValid***, per verificare la validità dell'itinerario, controllando che non si superi il dislivello massimo o la durata massima o la lunghezza massima o la difficoltà massima o la difficoltà media; restituisce una stringa null nel caso di itinerario valido, in caso di itinerario non valido la stringa contiene la descrizione del problema riscontrato
- ***calcMaxAltitudeDifference***, ***calcTotalTime***, ***calcTotalLength***, ***calcMaxDifficulty***, ***calcAverageDifficulty*** per calcolare le omonime proprietà dell'itinerario attuale; in particolare, la difficoltà media è definita come media pesata delle difficoltà dei sentieri componenti, assumendo come peso il prodotto *lunghezza*dislivello* di ogni sentiero, ovvero:

$$Difficoltà\ media = \frac{\sum_{s \in Itinerario} Lunghezza(s) \cdot Dislivello(s) \cdot Difficoltà(s)}{\sum_{s \in Itinerario} Lunghezza(s) \cdot Dislivello(s)}$$

e) la classe ***MyItinerary*** **(da realizzare)** implementa ***Itinerary***: l'itinerario è costruito inizialmente vuoto dal costruttore di default, in quanto i sentieri da percorrere saranno via via aggiunti tramite il metodo ***addTrail***. La classe deve implementare ovviamente i metodi sopra specificati, oltre agli opportuni metodi accessor;

f) la classe ***Trekking*** (fornita nello start kit) mantiene le strutture dati fondamentali del sistema; a tal fine definisce:

- un metodo ***loadTrails*** che, sfruttando il ***TrailReader*** ricevuto come argomento, carica i dati dal file Trails.txt;
- un metodo accessor ***getTrailList*** che recupera l'elenco dei sentieri.

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

Persistenza (package trekking.persistence) *(punti 7)*

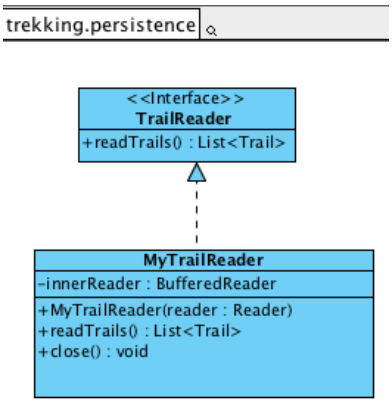
Il file di testo Trails.txt contiene la descrizione dei sentieri, uno per riga, nel seguente formato:

- Identificativo sentiero
- Località di partenza e, tra parentesi, altitudine
- Località di arrivo, tra parentesi, altitudine
- Lunghezza in km (con unità di misura)
- Difficoltà

```
ESTRATTO DAL FILE Trails.txt
GCA129, Segavecchia (925), Corno alle Scale (1725), 3.8 km, Difficolta 2
GCA000, Corno alle Scale (1725), Punta Corno (1732), 1.1 km, Difficolta 3
GCA121, Punta Corno (1732), Ponte del Ruscello (1005), 3.4 km, Difficolta 1
GCA111, Pian Verde (822), Val Bona (490), 5.0 km, Difficolta 2
GCA109, Val Bona (490), Cima Zanna (1233), 3.5 km, Difficolta 1
...
```

L' interfaccia ***TrailReader*** (fornita nello start kit) dichiara il metodo ***readTrails*** che legge da un ***Reader*** una lista di ***Trail;*** tale metodo dichiara di sollevare una ***InvalidTrailException*** (fornita nello start kit) destinata a incapsulare le specifiche eccezioni eventualmente riscontrate nella lettura da file.

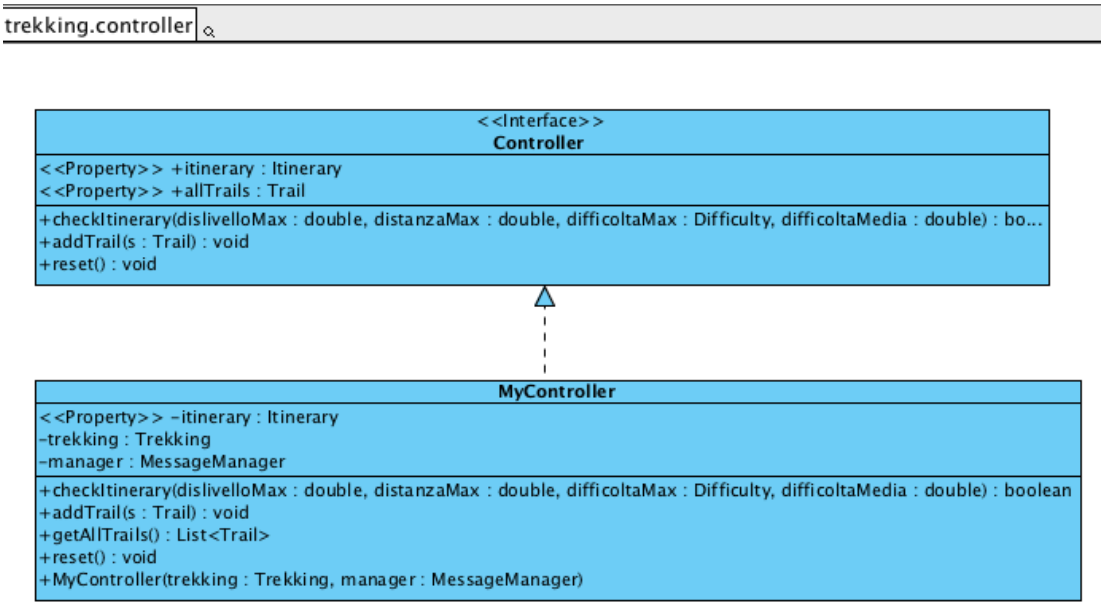
La classe ***MyTrailReader*** (**da realizzare**) implementa l'interfaccia e solleva l'eccezione richiesta quando opportuno.



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

Parte 2 (punti: 14)

Controller (package trekking.ui) *(punti 4)*



La classe ***MyController*** (**da realizzare**) implementa l'interfaccia ***Controller*** (fornita nello start kit): il costruttore riceve in ingresso un'istanza di ***Trekking*** e carica i dati dal file, nonché una istanza di ***MessageManager*** (interfaccia fornita nello start kit di cui è anche fornita una implementazione swing: ***SwingMessageManager***) che permette la visualizzazione di messaggi.

Il controller incapsula nel metodo ***checkItinerary*** il controllo globale di consistenza dell'itinerario che si va costruendo (delegandolo di fatto a ***Itinerary.isValid***).

L'interfaccia prevede anche metodi per recuperare l'elenco di tutti i sentieri (metodo ***getAllTrails***, da realizzare delegando la richiesta alla classe ***Trekking***) e per resettare l'itinerario che si sta costruendo ripartendo da un elenco sentieri vuoto (metodo ***reset***).

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

Interfaccia utente (namespace trekking.ui) *(punti 10)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nelle figure seguenti.

La classe ***Program*** (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 così articolata (Fig. 1):

- a sinistra, opportune combobox e campi di testo e bottoni controllano l'inserimento dei requisiti di utente (nell'ordine dall'alto in basso: *difficoltà max sopportabile dall'utente*, *dislivello max tollerabile dall'utente*, *lunghezza max accettabile dall'utente*, *difficoltà media sopportabile dall'utente*) e delle corrispondenti operazioni;
- in alto, una lista a discesa (combo) mostra tutti i sentieri disponibili, con le loro caratteristiche;
- al centro, un'area di testo mostra l'itinerario che si va costruendo, con le relative proprietà (lunghezza, durata, difficoltà, dislivello).

Ogni volta che l'utente chiede di aggiungere un sentiero, si controlla innanzitutto se esso è coerente con la parte di itinerario già costruita (ovvero, inizia dove termina il precedente): se così è, il sentiero viene aggiunto all'itinerario, che viene quindi aggiornato (Fig. 2); diversamente, viene visualizzato un opportuno messaggio d'errore (Fig. 3) e il sentiero non viene aggiunto [NB: per far apparire il messaggio d'errore si consiglia l'uso dell'interfaccia ***MessageManager*** e della corrispondente implementazione ***SwingMessageManager*** fornite nello start kit].

Il fatto che un sentiero sia stato aggiunto NON significa di per sé che l'itinerario rispetti i vincoli dell'utente: questa verifica viene svolta solo a richiesta, premendo il pulsante *Controlla*. Il risultato della verifica viene mostrato con opportuno messaggio, sia nel caso negativo (Fig. 4) che nel caso positivo (Fig. 5). Se l'esito è negativo, l'utente può decidere di rilassare un vincolo (ad esempio, accettando una difficoltà max superiore - Fig. 5) o può resettare tutto e ricominciare da capo (non è possibile eliminare un singolo sentiero già aggiunto all'itinerario).

Se in qualunque momento viene variato uno qualunque dei requisiti utente (ad esempio, aumentando o riducendo la difficoltà massima sopportabile dall'utente), l'intero itinerario deve essere immediatamente ri-verificato in automatico, come se si fosse premuto il pulsante *Controlla*.

In ogni momento, l'utente può stampare su file l'itinerario attuale (Fig. 6), che viene aggiunto *in modalità append* al file di testo Itineraries.txt: la stampa è nello stesso formato visualizzato nella GUI, mostrato anche in calce.

Il pulsante RESET consente naturalmente di svuotare tutto e ripartire dall'itinerario vuoto.

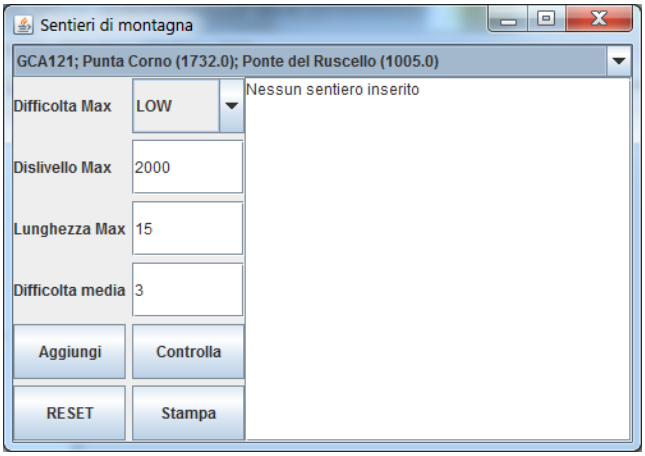


Fig. 1

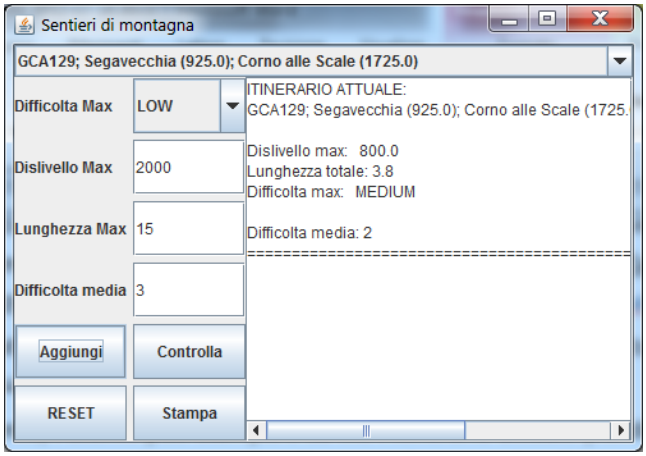
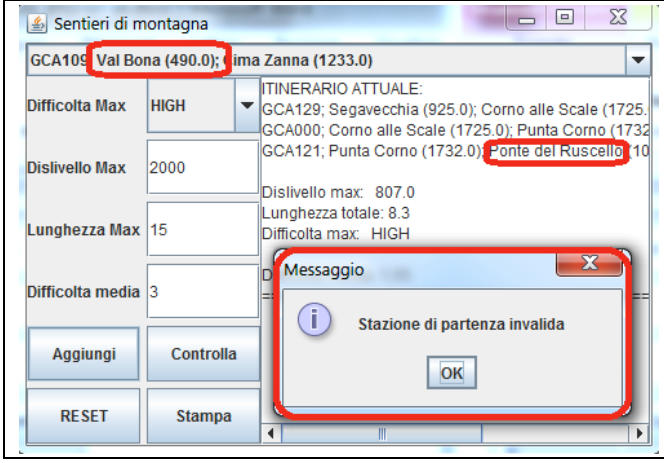


Fig. 2



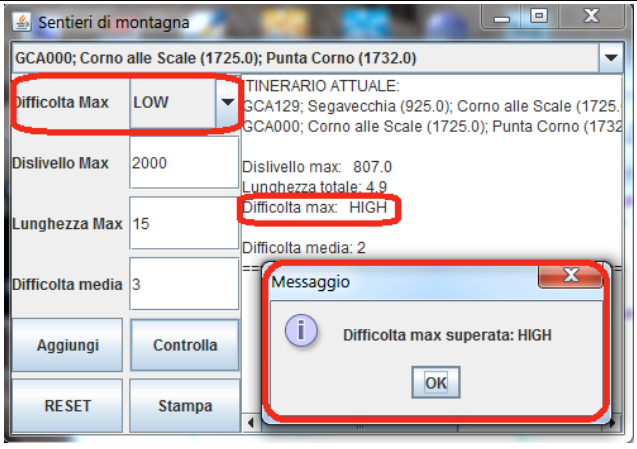


Fig. 3	Fig. 4

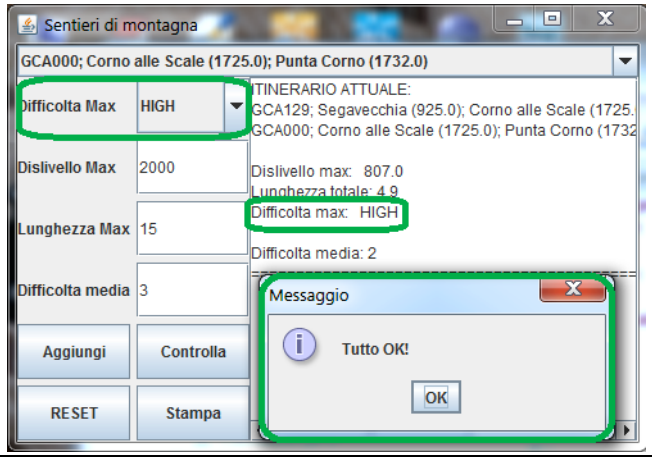
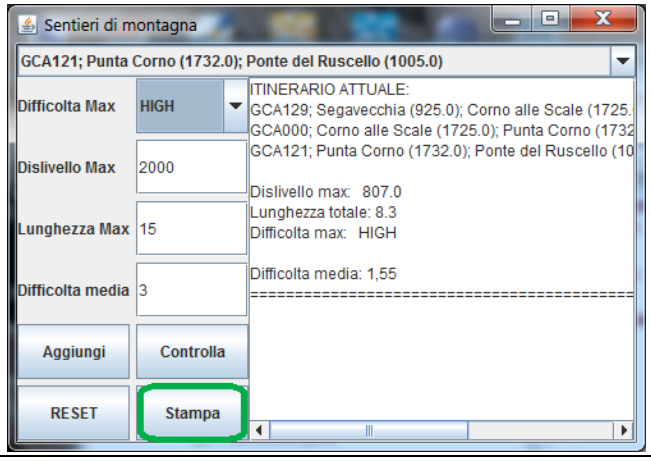

	

Fig. 5	Fig. 6

ESEMPIO DI ITINERARIO STAMPATO

```
ITINERARIO ATTUALE:
GCA129; Segavecchia (925.0); Corno alle Scale (1725.0)
GCA000; Corno alle Scale (1725.0); Punta Corno (1732.0)
GCA121; Punta Corno (1732.0); Ponte del Ruscello (1005.0)
Dislivello max:   807.0
Lunghezza totale: 8.3
Difficolta max:   HIGH
Difficolta media: 1,55
============================================================
```