ESAME DI FONDAMENTI DI INFORMATICA T-2 del 5/2/2014
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)

L'associazione di consumatori *ComproMeglio* ha richiesto un'applicazione per monitorare i prezzi di alcuni beni di largo consumo, in modo da sapere in ogni istante dove acquistarli ai prezzi migliori.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Ogni bene è caratterizzato da una *categoria*, un *codice univoco* e una descrizione. Quando i volontari dell'associazione girano nei supermercati, campionano i prezzi dei beni e compilano la corrispondente *rilevazione*: ogni rilevazione è a sua volta caratterizzata dal *codice univoco del bene* a cui si riferisce, *luogo* e *data* della rilevazione, *prezzo* rilevato.

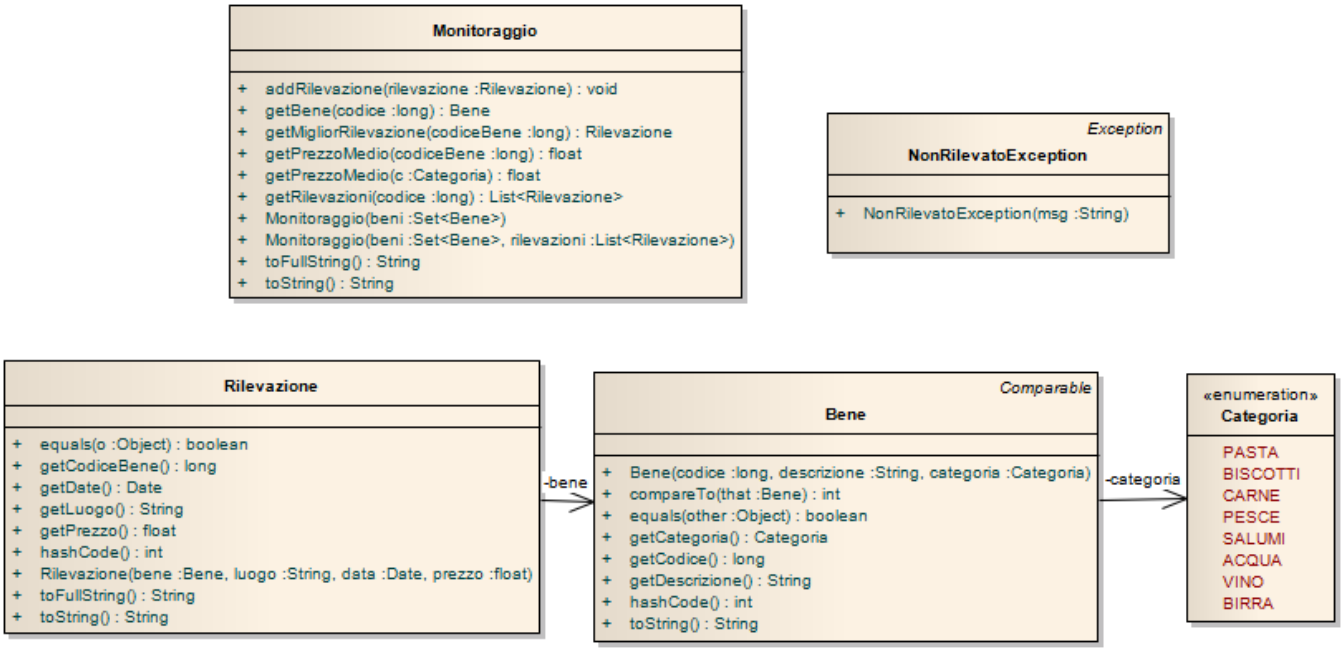
In ogni momento deve essere possibile sapere il *miglior prezzo* di un certo bene e in che luogo lo si possa trovare, il *prezzo medio* di quel bene nei vari supermercati campionati, nonché il prezzo medio di una *categoria* di beni.

I file di testo BeniDiConsumo.txt e Rilevazioni.txt contengono rispettivamente l'elenco dei beni di consumo monitorati e le rilevazioni disponibili.

Parte 1 (punti: 18)

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

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

a) L'enumerativo ***Categoria*** (fornito) elenca le categorie usate per classificare i beni (pasta, biscotti, carne, ecc.);

b) La classe ***Bene*** (fornita) rappresenta un bene con le sue proprietà;

c) La classe ***Rilevazione*** (fornita) rappresenta una rilevazione relativa a un certo bene, effettuata in un dato luogo e in una certa data; si noti che una ***Rilevazione*** si costruisce passando al costruttore, fra i vari argomenti, il ***Bene*** corrispondente, in modo che da ogni oggetto ***Rilevazione*** si possa ottenere il codice del ***Bene*** associato (metodo ***getCodiceBene***);

d) **La classe *Monitoraggio* (da realizzare)** rappresenta uno specifico monitoraggio effettuato da *ComproMeglio*, inteso come insieme di beni con le rispettive rilevazioni effettuate in un dato arco temporale. In dettaglio:

- ***Monitoraggio(Set<Bene>)*** crea un nuovo monitoraggio per l'insieme di beni indicato, senza rilevazioni;
- ***Monitoraggio(Set<Bene>, List<Rilevazione>)*** crea un nuovo monitoraggio per l'insieme di beni indicato, associando a ogni bene le rilevazioni a esso corrispondenti;

La classe espone inoltre i seguenti metodi:

- ***getBene(long)*** ritorna il ***Bene*** gestito avente il codice passato come argomento, o ***null*** se non si sta gestendo il ***Bene*** richiesto;
- ***getRilevazioni(long)*** ritorna la lista delle rilevazioni associate al ***Bene*** avente il codice passato come argomento, o ***null*** se non si sta gestendo tale codice;
- ***getMigliorRilevazione(long)*** ritorna la ***Rilevazione*** che offre il miglior prezzo rilevato per il ***Bene*** avente il codice passato come argomento: se non esiste alcuna rilevazione per esso, solleva un'eccezione di tipo ***NonRilevatoException*** (fornita nello start kit);
- ***getPrezzoMedio(long)*** ritorna il prezzo medio, in Euro, rilevato per il ***Bene*** avente il codice passato come argomento: se non esiste alcuna rilevazione per esso, solleva un'eccezione di tipo ***NonRilevatoException***;
- ***getPrezzoMedio(Categoria)*** ritorna il prezzo medio, in Euro, rilevato per quella categoria di beni: se non esistono rilevazioni per tale ***Categoria***, solleva un'eccezione di tipo ***NonRilevatoException***;
- ***addRilevazione(Rilevazione)*** aggiunge la rilevazione passata come argomento alla lista delle rilevazioni associate a un certo bene (deducibile dalla ***Rilevazione*** stessa), lanciando ***IllegalArgumentException*** se risulta già effettuata una rilevazione per lo stesso bene nello stesso luogo e data: in tal caso, il messaggio associato all'eccezione deve descrivere con precisione il motivo del rifiuto;
- ***toString*** genera l'elenco dei beni per i quali esiste almeno una rilevazione, sfruttando opportunamente il metodo ***toString*** della classe ***Bene***;
- ***toFullString*** deve generare l'elenco delle rilevazioni effettuate, sfruttando opportunamente il metodo ***toString*** della classe ***Rilevazione.***

Persistenza (package compromeglio.persistence) *(punti 9)*

Come già anticipato, i file di testo BeniDiConsumo.txt e Rilevazioni.txt contengono rispettivamente l'elenco dei beni di consumo monitorati e le rilevazioni disponibili per essi. Nel primo file ogni riga descrive un bene, tramite più campi (codice, categoria, descrizione) separati da tabulazioni:

```
Formato del file BeniDiConsumo.txt
27991   PASTA       Spaghetti Dentilla n. 3   500g
29228   PASTA       Rigatoni Zannoni sempre i più buoni
...
```

Nel secondo file, ogni riga descrive una *rilevazione,* caratterizzata a sua volta da *codice del bene rilevato*, *luogo*, *data* e *prezzo* in Euro, separati da virgole:

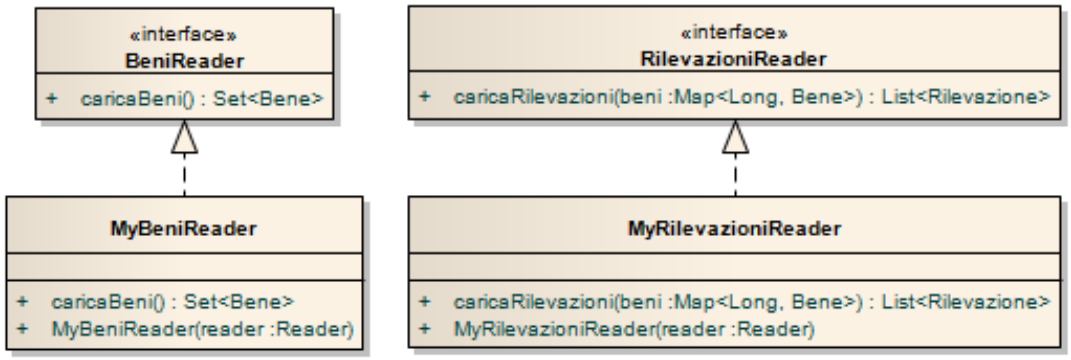
```
Formato del file Rilevazioni.txt
27991,  Market Dentinia, 18/01/2014, 0.95
27991,  Market Zannonia, 19/01/2014, 1.05
...
```

Le due interfacce ***BeniReader*** e ***RilevazioniReader*** (fornite) dichiarano i metodi di lettura per leggere i dati dal ***Reader*** passato come argomento: più precisamente, ***caricaBeni*** genera e restituisce l'insieme dei beni, mentre

caricaRilevazioni carica la lista delle rilevazioni effettuate prendendo come parametro una mappa codice bene / bene opportunamente popolata.

Le due classi *MyBeniReader* e *MyRilevazioniReader* (da realizzare) implementano tali interfacce lanciando ***MalformedFileException*** (fornita, ma non mostrata nel diagramma UML per brevità) in caso di errore nel file, con messaggio specializzato in base all'accaduto.

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



Parte 2 (punti: 12)

L'applicazione deve consentire di sapere in ogni momento:

1) il *miglior prezzo di un certo bene* e in che luogo lo si possa trovare;
2) il *prezzo medio di un certo bene* nei vari supermercati campionati;
3) il prezzo medio di una *categoria* di beni.

Per facilitare la navigazione, la GUI è organizzata su due combo e vari campi di testo (vedere figure): più esattamente, si prevede che l'utente scelga dapprima la *categoria* che interessa da un'apposita combo, con ciò causando il popolamento dinamico di un'altra combo con i soli *beni* della categoria prescelta, e successivamente che egli scelga eventualmente uno specifico bene da tale seconda combo.

A seguito di tali selezioni, tre campi di testo sottostanti mostrano le seguenti informazioni:

a) il campo di testo *prezzo medio per categoria*, aggiornato non appena l'utente sceglie la categoria dalla prima combo, mostra il prezzo medio per la categoria di beni prescelta;

b) il campo *miglior prezzo*, aggiornato invece ogni qual volta l'utente sceglie uno specifico bene dalla seconda combo, mostra il miglior prezzo di quel bene, con il dettaglio di dove sia stato rilevato e quando;

c) similmente, il campo *prezzo medio*, anch'esso aggiornato ogni volta che l'utente sceglie uno specifico bene dalla seconda combo, mostra il prezzo medio di quel bene;

d) una TextArea centrale, aggiornata ogni qual volta l'utente sceglie uno specifico bene dalla seconda combo, mostra i dettagli della rilevazione migliore a cui si riferisce anche il campo di testo "miglior prezzo".

La scelta di uno specifico bene, dopo aver selezionato la categoria, non è obbligatoria: in tal caso, l'applicazione mostrerà solo il prezzo medio per categoria, lasciando vuoti gli altri due campi.

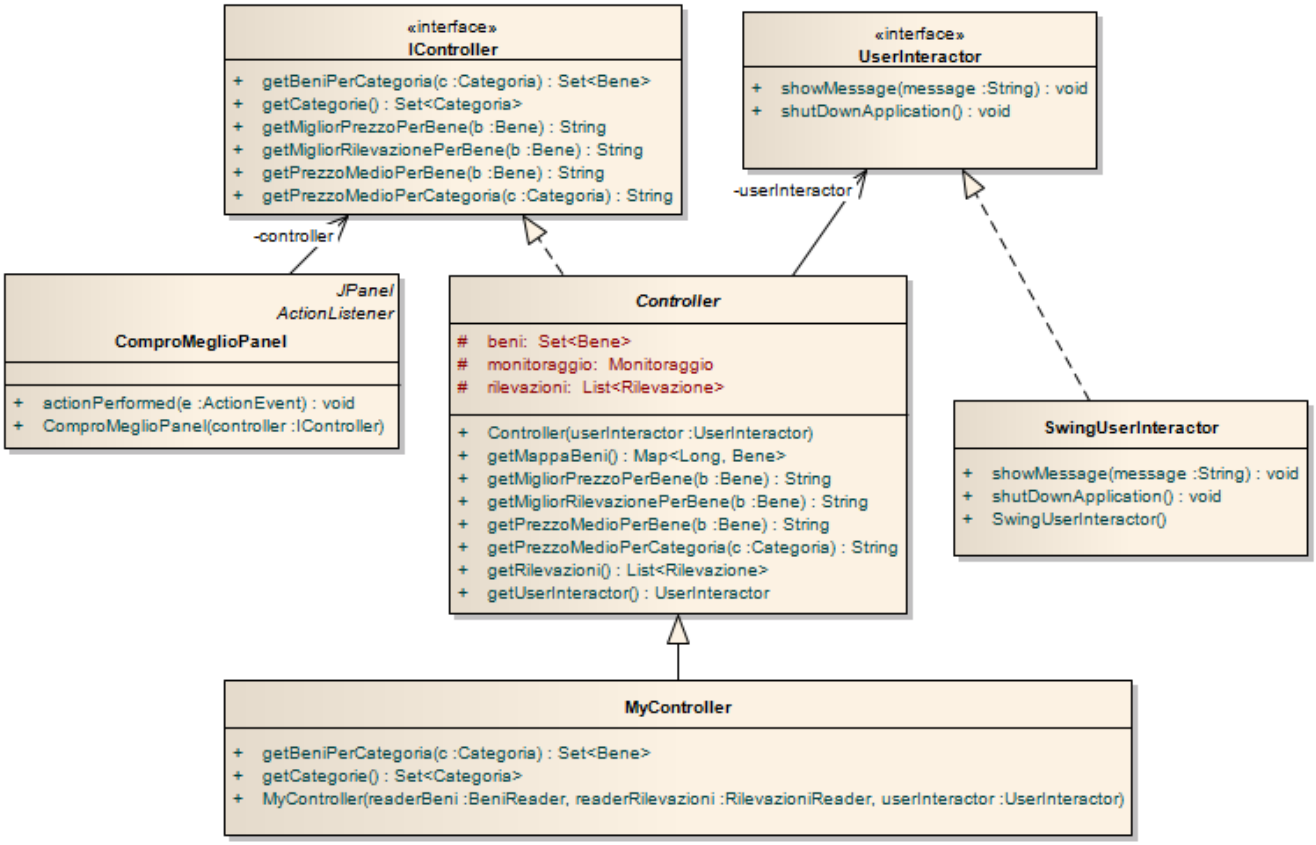
Si noti che è possibile selezionare un ***Bene*** dalla seconda combo per il quale non sono presenti rilevazioni, nel qual caso le aree di testo specifiche per il ***Bene*** devono rimanere vuote.

→→ SEGUE → →

Controller (package compromeglio.ui) *(punti 6)*

La classe *MyController* (da realizzare) concretizza la classe astratta ***Controller*** (fornita) che a sua volta implementa l'interfaccia ***IController*** (pure fornita) rispettando i seguenti vincoli:

- il costruttore prende in ingresso un ***BeniReader***, un ***RilevazioniReader***, uno ***UserInteractor*** per mostrare un messaggio all'utente (***showMessage***) e chiudere l'applicazione (***shutDownApplication***), e un ***Monitoraggio***: questo costruttore invoca il costruttore della classe base ***Controller*** (il quale effettua la lettura dei beni e delle rilevazioni) e assegna il ***Monitoraggio*** ad un opportuno campo privato della classe;
- ***getBeniPerCategoria*** (**da implementare**) restituisce l'elenco dei ***Bene*** appartenenti ad una data ***Categoria*** passata come argomento;
- ***getCategorie*** (**da implementare**) restituisce l'elenco delle categorie di bene disponibili;
- ***getMappaBeni*** (ereditato da ***Controller***) restituisce la mappa codice bene / ***Bene*** dei beni correntemente gestiti;
- ***getPrezzoMedioPerCategoria*** (ereditato da ***Controller***) restituisce il prezzo medio di una certa ***Categoria*** passata come argomento;
- ***getMigliorPrezzoPerBene*** (ereditato da ***Controller***) restituisce il miglior prezzo rilevato di un certo ***Bene*** passato come argomento;
- ***getPrezzoMedioPerBene*** (ereditato da ***Controller***) restituisce il prezzo medio di un certo ***Bene*** passato come argomento;
- ***getMigliorRilevazionePerBene*** (ereditato da ***Controller***) restituisce a migliore rilevazione rilevato di un certo ***Bene*** passato come argomento;
- ***getUserInteractor*** (ereditato da ***Controller***) restituisce uno ***UserInteractor***.



Interfaccia utente (package compromeglio.ui) ***(punti 6)***

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

> PREMESSA: alla partenza, si caricano dai file i beni e le rilevazioni: un eventuale errore di formato dev'essere segnalato immediatamente tramite finestra di dialogo, nel qual caso l'applicazione esce senza neanche mostrare la GUI.

Se il caricamento preliminare ha esito positivo, compare la finestra principale dell'applicazione (Fig. 1); la classe ***ComproMeglioPanel*** **(da realizzare)** implementa il pannello centrale del frame e prevede il seguente funzionamento:

- in primo luogo si deve scegliere una categoria di beni dalla combo box delle categorie (in alto, Fig. 1), ottenendo in risposta il popolamento della combo sottostante dei beni (in alto, Fig. 2) e il prezzo medio per categoria nell'apposito campo di testo sottostante (sulla sinistra, Fig. 2);
- successivamente, si sceglie uno specifico bene di quella categoria dalla combo dei beni (Fig. 3), ottenendo in risposta (Fig. 4), se esistono rilevazioni per quel bene: a) i dettagli della miglior rilevazione, nell'area centrale; b) il miglior prezzo, nell'area a sinistra al centro; c) il prezzo medio di quel bene, nell'area a sinistra in basso.

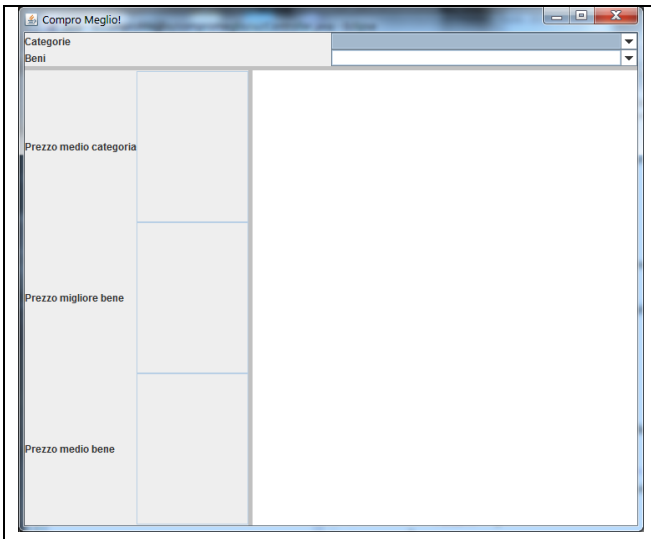


Fig. 1

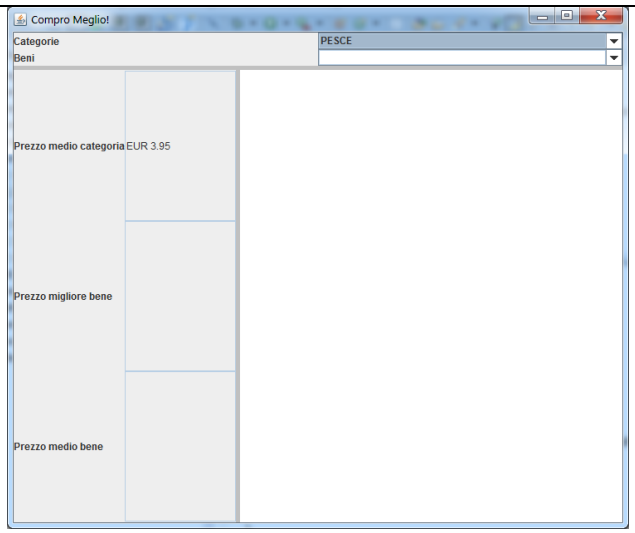


Fig. 2

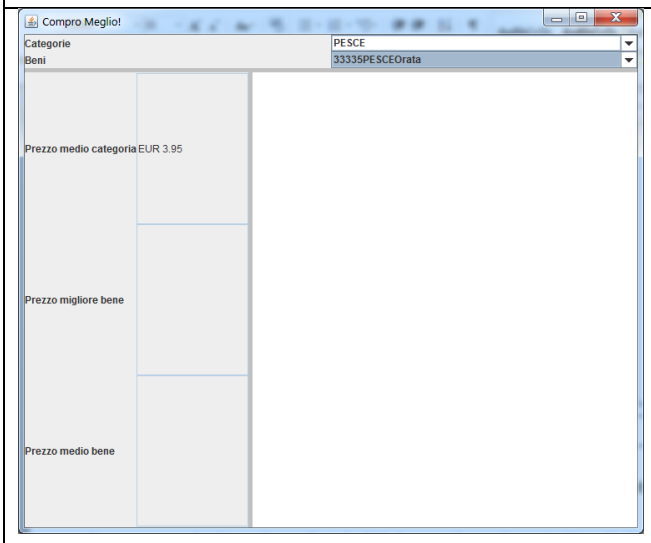


Fig. 3

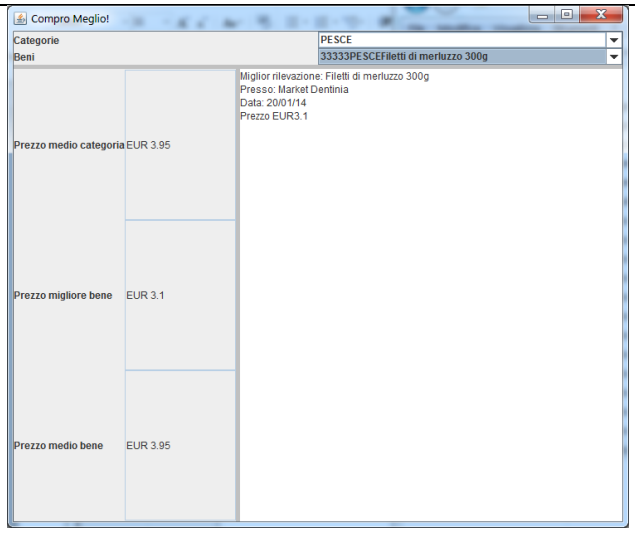


Fig. 4

Nel caso che invece non ci siano rilevazioni disponibili per il bene selezionato, viene mostrato solo il prezzo medio della categoria (Fig. 3), lasciando vuoti gli altri campi.

Come sempre, la classe che realizza la GUI riceve nel proprio costruttore il riferimento al ***Controller*** associato.

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