ESAME DI FONDAMENTI DI INFORMATICA T-2 del 02/07/2014
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore

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

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

Il famoso ristorante *Dento's* ha richiesto un'applicazione per gestire i menù da proporre ai propri clienti.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

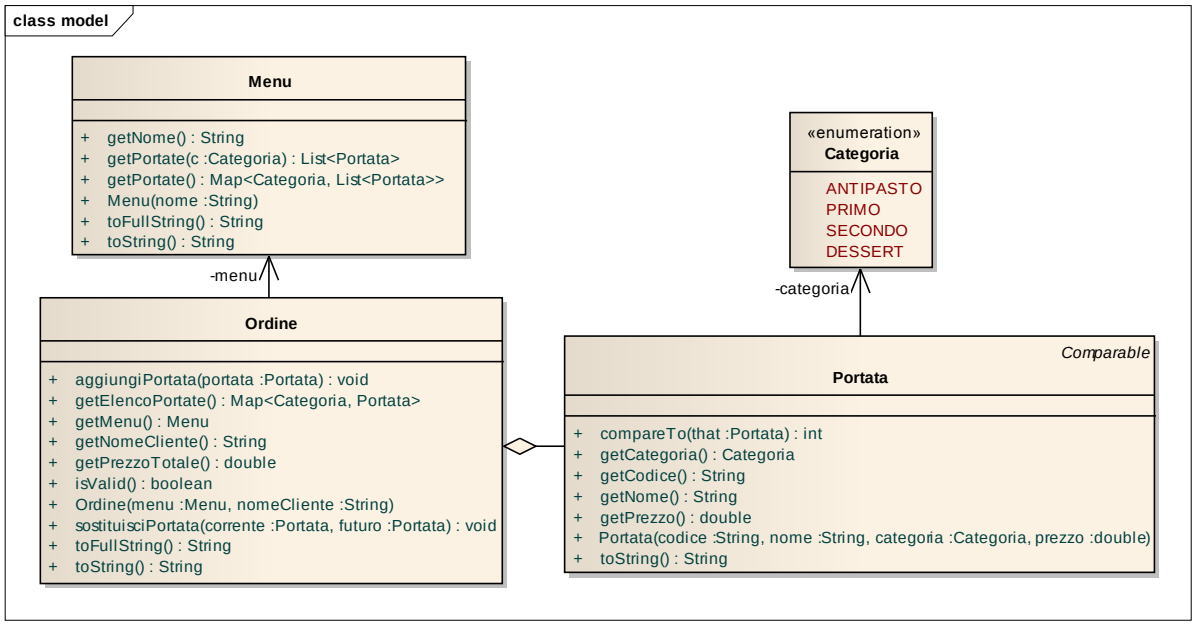
Il ristorante propone ai propri clienti una serie di *menù* prestabiliti. Ogni menù è costituito da un insieme di *portate*, appartenenti a ben precise *categorie (antipasti, primi, secondi,* ecc.). Ogni portata è caratterizzata da codice univoco, descrizione, categoria e prezzo. I clienti possono personalizzare il proprio menù scegliendo, all'atto dell'ordine, una (e una sola) portata fra quelle proposte in ciascuna categoria: il prezzo finale varierà di conseguenza.

I file di testo Portate.txt e Menu.txt contengono rispettivamente l'elenco delle portate e i menù già predisposti, con le possibili varianti (vedere più oltre per i dettagli).

Parte 1 (punti: 18)

Dati (namespace dentorestaurant.model) *(punti: 6)*

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



SEMANTICA:

a) L'enumerativo ***Categoria*** (fornito) definisce i valori ANTIPASTO, PRIMO, SECONDO, DESSERT*;*

b) La classe ***Portata*** (fornita) rappresenta una portata con le varie proprietà;

c) La classe ***Menu*** (fornita) rappresenta un menù con le rispettive proprietà, in particolare, un menù racchiude in una mappa un elenco di portate per ogni categoria: la mappa è recuperabile tramite il metodo ***getPortate*** senza parametri, mentre la lista di portate per categoria è recuperabile mediante il metodo ***getPortate(Categoria)***.

d) **La classe *Ordine* (da realizzare)** rappresenta il menu ordinato da uno specifico cliente, ossia l'insieme delle portate da lui scelte (la memorizzazione delle coppie deve avvenire in un'apposita mappa); un ordine può contenere al più una portata per categoria (ad es., non è possibile ordinare due primi). Oltre agli opportuni accessor, la classe espone i metodi:

- ***Ordine*** (costruttore), i cui argomenti sono il menù a cui ci si riferisce e il nome del cliente: entrambi gli argomenti non possono essere nulli e il nome del client non può essere la stringa vuota o una stringa composta da soli spazi;
- ***getElencoPortate***, che restituisce la mappa che mette in corrispondenza una categoria con la corrispondente portata ordinata
- ***aggiungiPortata***, che aggiunge una portata all'ordine: nel caso sia già presente una portata della stessa categoria, occorre lanciare una ***IllegalArgumentException***
- ***getNomeCliente***, restituisce il nome del cliente specificato nel costruttore
- ***sostituisciPortata***, che cambia una portata (primo argomento) con un'altra (secondo argomento) della stessa categoria; se l'operazione fallisce (se la portata da sostituire non è presente, se si tenta di sostituire una portata di una certa categoria con una portata di un'altra categoria o se la nuova portata non è presente nel menù di riferimento) si deve lanciare una ***IllegalArgumentException*** con messaggio d'errore adeguato alla circostanza;
- ***isValid***, che verifica se l'ordine contiene una portata per ogni categoria;
- ***getPrezzoTotale***, che restituisce il prezzo totale dell'ordine configurato in base alle portate scelte;
- ***toString***, che produce una rappresentazione compatta (solo nome e prezzo totale);
- ***toFullString***, che produce una rappresentazione estesa (nome, prezzo totale ed elenco delle portate.

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

Persistenza (namespace dentorestaurant.persistence) *(punti 12)*

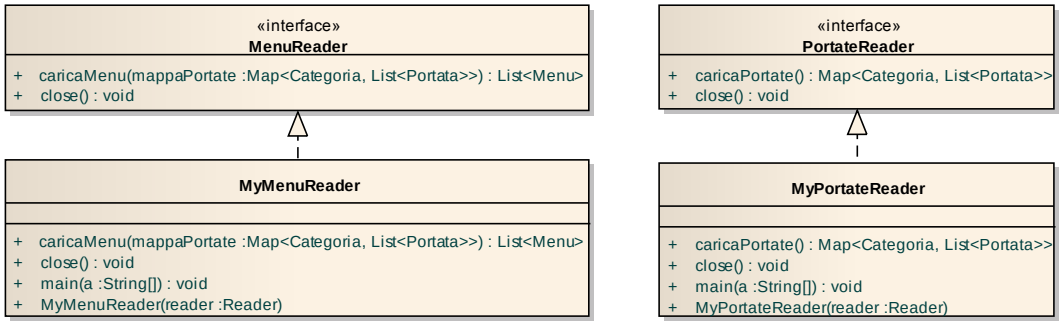
Come anticipato, Portate.txt e Menu.txt contengono rispettivamente le portate disponibili e i menù già predisposti.

Nel file Portate.txt ogni riga contiene nell'ordine codice univoco, descrizione, categoria e prezzo, separati da virgole:

```
Formato del file Portate.txt
A01, Prosciutto e melone, ANTIPASTO, 9.00
A02, Antipasto di mare, ANTIPASTO, 8.00
...
```

Il file Menu.txt contiene invece una serie di record che descrivono i vari menù: la prima riga contiene la parola chiave MENU seguita dal nome del menu, mentre le righe successive elencano le portate fra cui si può scegliere nelle diverse categorie (una categoria per riga), riportandone i codici univoci separati da virgole, come nell'esempio sotto; il record è poi chiuso dalla riga contenente "END MENU".

```
Formato del file Menu.txt
MENU base
ANTIPASTI: A01, A02
PRIMI: P02, P04
SECONDI: S03, S05
DESSERT: D01
END MENU
...
```



Le interfacce ***PortateReader*** e ***MenuReader*** (fornite) dichiarano rispettivamente i metodi ***caricaPortate*** (senza parametri) e ***caricaMenu*** (prende in ingresso una mappa di liste di portate ovvero l'output di ***caricaPortate***) che leggono da un ***Reader*** rispettivamente una mappa di liste di ***Portate*** indicizzata per ***Categoria*** (per ogni ***Categoria***, una lista di portate afferenti alla Categoria stessa) e una lista di ***Menu.***

Le classi *MyPortateReader* e *MyMenuReader* (da realizzare) implementano tali interfacce, effettuando i necessari controlli di formato ove opportuni e lanciando ***MalformedFileException*** (fornita) in caso di errore di formato, con opportuno messaggio d'errore specializzato in base all'accaduto. I costruttori di entrambe le classi prendono un ***Reader*** come parametro e lanciano ***IllegalArgumentException*** in caso in cui questo sia null. I metodi ***close*** dei reader implementati non devono fare altro che chiudere il ***Reader*** ricevuto in ingresso nel costruttore e sopprimere tutte le eventuali eccezioni.

Nota 1: in ***MyPortateReader*** per trasformare il token che rappresenta una categoria in un'istanza dell'enumerativo ***Categoria*** è opportuno usare il metodo statico ***valueOf*** degli enumerativi; nel caso in cui la stringa non corrisponda al nome di un valore dell'enumerativo, il metodo lancia una ***MalformedFileException***. Nella mappa restituita deve essere presente una entry per ogni categoria - dove, al limite, la lista corrispondente è vuota.

Nota 2: in ***MyMenuReader*** per popolare l'elenco di portate per categoria di un ***Menu*** occorre recuperare la lista di portate per la categoria mediante il metodo ***Menu.getPortate(Categoria)***, poi popolare direttamente la lista ricevuta, che è la lista interna alla classe ***Menu***. Un esempio d'uso è riportato di seguito:

```
Menu m = ...;
Portata p = ...;
List<Portata> portatePerCategoria = m.getPortate(Categoria.ANTIPASTI);
portatePerCategoria.add(p);
```

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

Parte 2 (punti: 12)

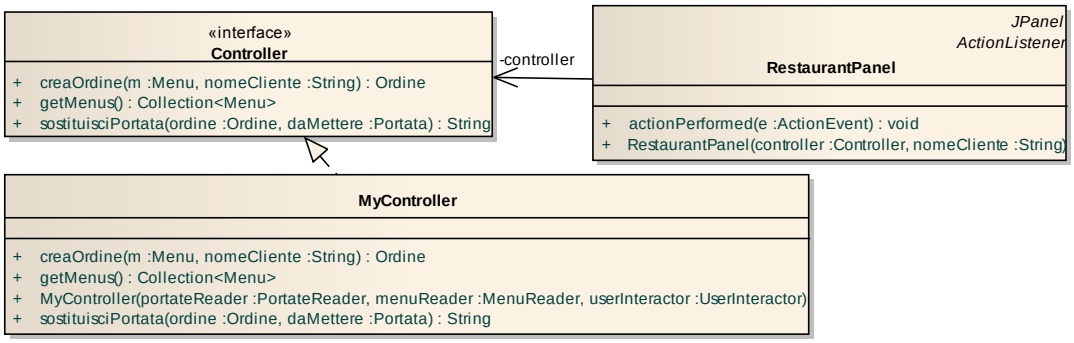
L'obiettivo dell'applicazione è permettere al cameriere (o al cliente dotato di tablet) di confezionare l'ordine di un cliente, partendo dal menu prescelto ed effettuando al suo interno le scelte fra le diverse portate offerte.

Controller (namespace dentorestaurant.pianidistudi.ui) *(punti 6)*

La classe *MyController* (da realizzare) implementa l'interfaccia ***Controller*** (fornita). Il costruttore riceve come argomenti i due reader e uno ***UserInteractor***, provvede al popolamento delle strutture dati necessarie e infine chiude i reader; in caso di errore nella lettura del file, notifica l'utente e termina l'applicazione (catturare le opportune eccezioni). Per terminare l'applicazione, usare il metodo ***shutDownApplication*** di ***UserInteractor.***

Il metodo ***getMenus*** restituisce una ***Collection*** di ***Menu*** (ottenuta nel costruttore mediante i reader).

Il metodo ***creaOrdine*** crea un ***Ordine*** partendo dal ***Menu*** e dal nome del cliente passati come parametro, usando l'opportuno costruttore di ***Ordine***. Il metodo ***sostituisciPortata*** effettua la sostituzione, nell'***Ordine*** passato come primo argomento, della ***Portata*** passata come secondo argomento: la portata da sostituire viene determinata cercando nell' ***Ordine*** passato la portata correntemente selezionata con la categoria della portata passata. Il metodo restituisce una stringa *null* in caso di successo, o contenente il messaggio opportuno in caso d'errore: in particolare, tale messaggio è ottenuto catturando le opportune eccezioni, e recuperando il relativo messaggio.



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

La classe ***Program*** (non mostrata nel diagramma UML, ma fornita nello start kit) contiene il main dell'intera applicazione. L'interfaccia utente deve essere simile (non necessariamente identica) agli esempi mostrati di seguito.

Appena l'applicazione parte, vengono caricati dai due file tutte le portate e i menù (classi ***Program*** + ***MyController***).

La GUI, espressa dalla **classe *RestaurantPanel* (da realizzare),** consente di:

- scegliere un menù fra quelli disponibili (nella combo box in alto, Fig. 1): come risultato, vengono popolate le successive combo delle portate, divise per categorie e viene calcolato il prezzo (Fig. 2);
- modificare una portata fra quelle di una data categoria, sostituendo quella preselezionata nella combo con un'altra: la modifica causa naturalmente l'aggiornamento del prezzo (Fig. 3);
- cambiare idea e selezionare un altro menù, con conseguente ri-aggiornamento generale e ri-popolamento delle combo e ricalcolo del prezzo (Fig. 4).

Tutte le operazioni sui dati devono essere effettuate mediante il ***Controller*** ricevuto in ingresso dal costruttore.

NB: essendo *RestaurantPanel* un pannello, è derivato necessariamente da *JPanel.* Esso viene poi integrato in un opportuno frame dalla classe ***Program*** fornita nello start kit.

Fig. 1

Fig. 2

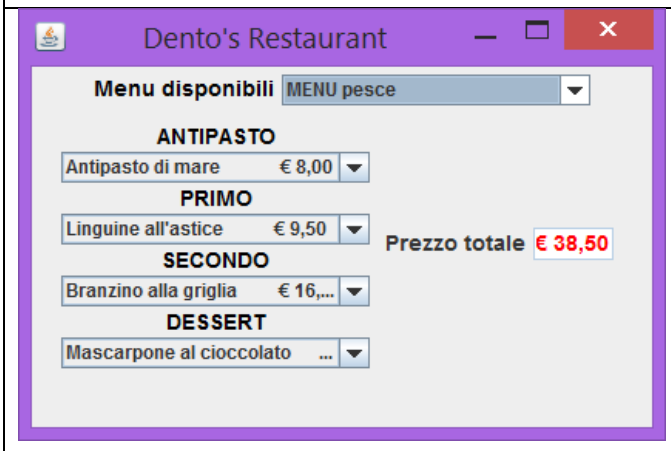


Fig. 3

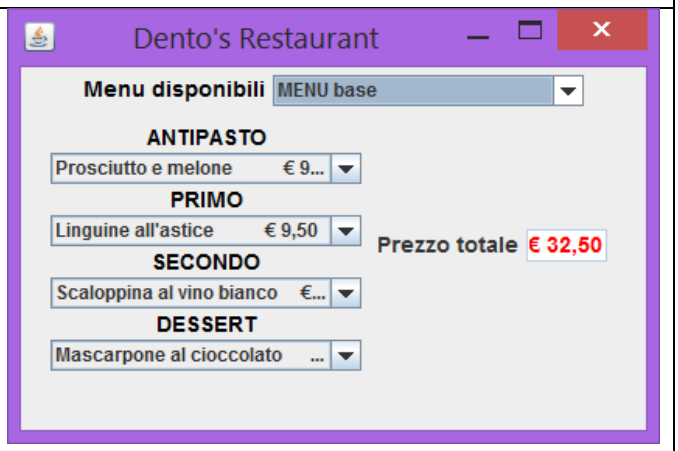


Fig. 4

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