ESAME DI FONDAMENTI DI INFORMATICA T-2 del 5/7/2023

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 *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO"

L'Associazione Consumatori di Dentinia ha richiesto lo sviluppo di un'applicazione che agevoli l'analisi delle spese sanitarie, separandole per tipologia e indicando quelle detraibili e quelle non detraibili.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Fra i suoi molti servizi, l'Associazione Consumatori di Dentinia assiste anche gli utenti nell'analisi delle loro spese sanitarie, che si distinguono in diverse *tipologie*:

1) spese per farmaci
2) spese per dispositivi medici
3) spese per ticket
4) spese per libera professione (visite mediche, interventi di specialistica, etc.)
5) altre spese, diverse dalle precedenti.

Le prime quattro tipologie sono *detraibili ai fini fiscali*, mentre l'ultima non lo è.

Un *documento di spesa* può comprendere più *voci di spesa*, ciascuna con la propria tipologia.

Alla fine dell'anno, il Governatorato di Dentinia rende disponibile a ciascun cittadino, su un apposito file di testo, l'elenco di tutte le spese fatte nell'anno, specificando per ciascuna:

- la data in cui ogni documento di spesa è stato emesso
- l'indicazione dell'emittente (farmacia, medico, AUSL, etc.) di tale documento
- l'importo totale di tale documento
- l'elenco delle singole voci di spesa dettagliate in tale documento, ciascuna con la relativa tipologia e importo

Dovrà quindi essere possibile analizzare tali informazioni, precisamente:

- mostrando il totale delle spese effettuate
- calcolando separatamente il totale delle spese detraibili e di quelle non detraibili
- mostrando, a richiesta, il dettaglio delle singole spese di una data categoria a scelta dell'utente (ad esempio, solo quella per farmaci, solo quella per ticket, etc.)

Il file di testo spesesanitarie.txt, descritto più oltre, contiene i dati di una serie di documenti di spesa.

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

PARTE 1 – Modello dei dati: Punti 13 [TEMPO STIMATO: 60-75 minuti]

PARTE 2 – Persistenza: Punti 11 [TEMPO STIMATO: 55-65 minuti]

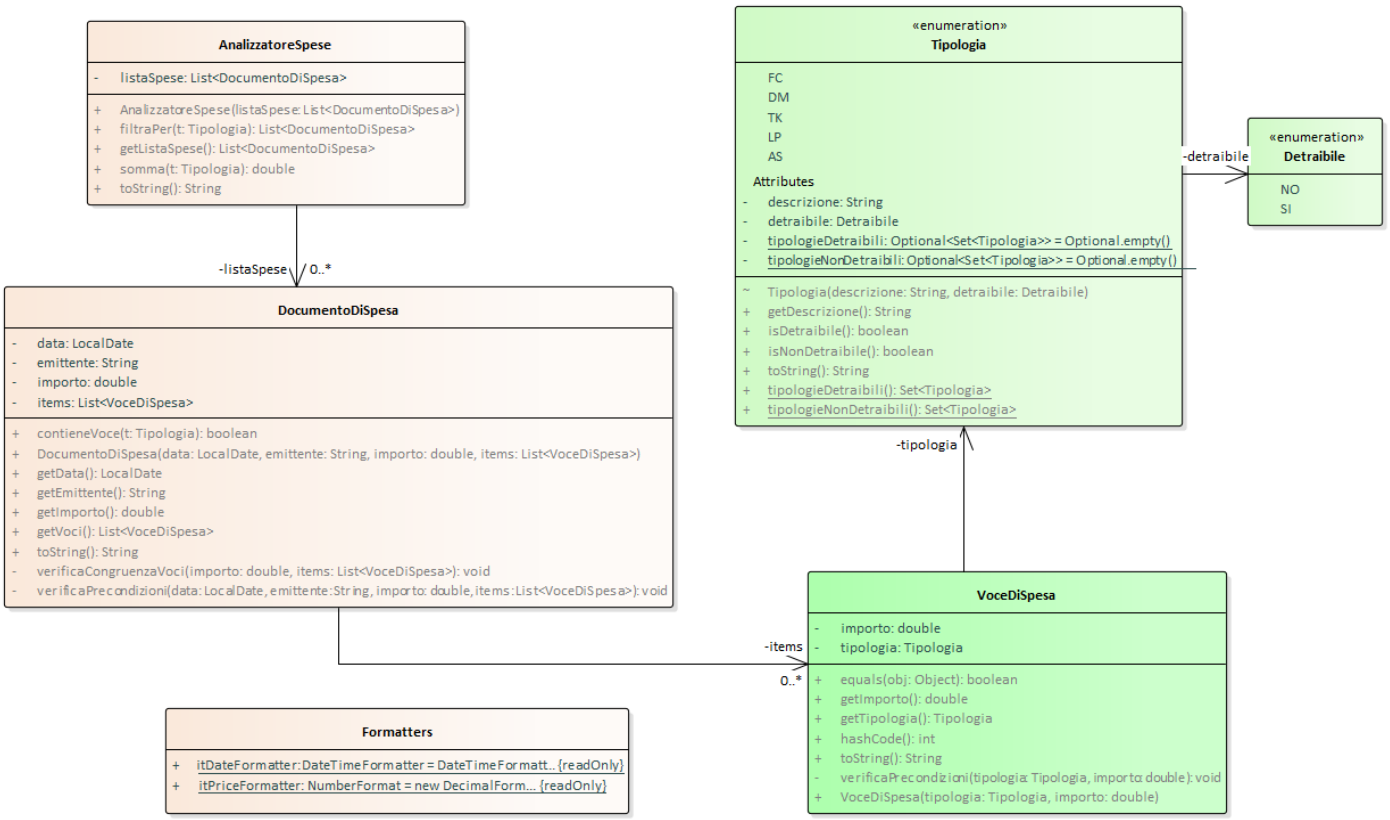
PARTE 3 – Grafica: Punti 6 [TEMPO STIMATO: 20-40 minuti]

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-19\lib"
--add-modules javafx.controls
```

Parte 1 – Modello dei dati (punti: 13)

Package: spesesanitarie.model **[TEMPO STIMATO: 60-75 minuti]**



SEMANTICA:

a) L'enumerativo ***Detraibile*** (fornito) definisce semplicemente due costanti Sì/No

b) L'enumerativo ***Tipologia*** (fornito) definisce le costanti corrispondenti ai tipi di spese definiti nel dominio del problema: a ciascuna sono associate una descrizione testuale e una proprietà booleana che specifica se tale tipologia di spesa sia detraibile o meno. La coppia di metodi *isDetraibile* / *isNonDetraibile* consente di verificare tale proprietà, mentre la coppia di metodi statici *tipologieDetraibili* / *tipologieNonDetraibili* restituisce l'insieme delle tipologie che, rispettivamente, sono / non sono detraibili.

c) La classe ***Formatters*** (da realizzare) definisce i due formattatori personalizzati usati da tutta l'applicazione: in particolare, *itDateFormatter* è un formattatore per data secondo lo standard italiano ma con l'anno su quattro cifre, mentre *itPriceFormatter* è un formattatore per prezzi in Euro col simbolo di valuta davanti al prezzo anziché dietro, che usa sempre esattamente due cifre decimali dopo la virgola.

d) La classe ***VoceDiSpesa*** (fornita) definisce una voce di spesa di uno scontrino o fattura, caratterizzata da tipologia e importo. Il costruttore verifica gli argomenti in ingresso, lanciando ***IllegalArgumentException*** con adeguato messaggio d'errore in caso di incoerenze. Appositi accessor consentono di recuperare le singole proprietà. Sono incluse opportune implementazioni di *equals*, *toString* ed *hashcode*.

e) La classe ***DocumentoDiSpesa*** (da completare nelle verifiche di precondizioni del costruttore e nel metodo *toString*) rappresenta un documento di spesa (scontrino, fattura, ricevuta fiscale, etc.), costituito da un elenco di una o più ***VoceDiSpesa***; il documento è caratterizzato altresì dalla data di emissione, dalla descrizione dell'emittente e dall'importo totale del documento stesso. Il costruttore riceve perciò tali argomenti (nell'ordine: data, emittente, importo totale e lista di voci), recuperabili tramite appositi accessor. Devono essere implementate le seguenti verifiche nel costruttore:

- Verifiche di precondizioni: che nessun argomento sia null, che la lista contenga almeno un elemento, e che l'importo sia un numero reale finito e non negativo
- Verifiche di congruenza: che la somma delle voci di spesa sia uguale all'importo

La classe definisce i vari accessor, nonché un'opportuna *toString*

f) La classe ***AnalizzatoreSpese*** (da completare) analizza una collezione di documenti di spesa. Il costruttore riceve una lista di ***DocumentoDiSpesa***, recuperabile tramite appositi accessor. Devono essere implementati:
   - il metodo *somma*, che restituisce il totale delle spese effettuate per la ***Tipologia*** specificata;
   - il metodo *filtraPer*, che restituisce la lista dei soli ***DocumentoDiSpesa*** che contengono almeno una spesa della ***Tipologia*** specificata.

Parte 2 – Persistenza (punti: 11)

Package: spesesanitarie.persistence **[TEMPO STIMATO: 55-65 minuti]**

Il file di testo spesesanitarie.txt contiene i dati di una serie di documenti di spesa, uno dietro l'altro. Ogni documento di spesa è organizzato su più righe: in tutte, gli elementi sono separati dal carattere "punto e virgola" (';').

La prima riga specifica quattro elementi:

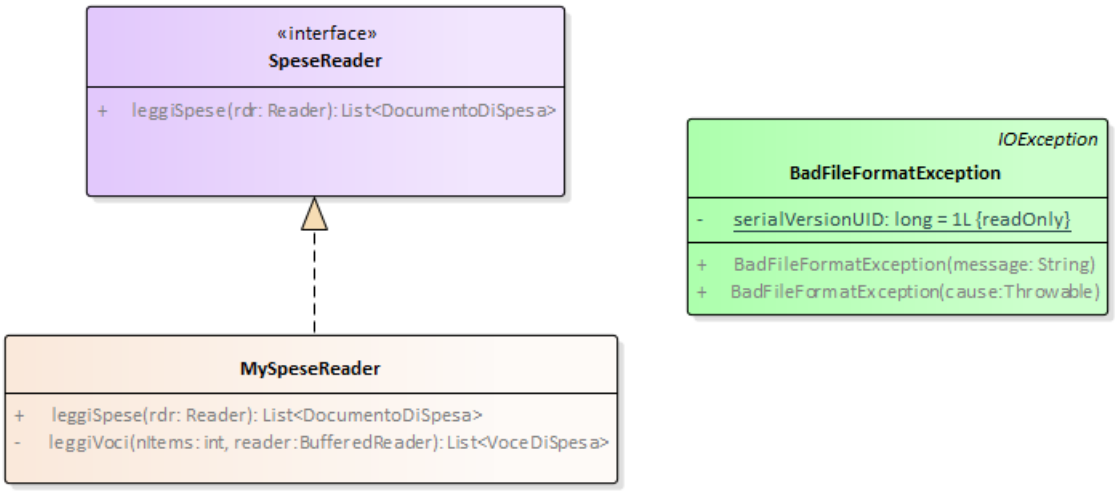
- la data di emissione del documento, nel formato GG/MM/AAAA;
- l'emittente della prestazione (una stringa);
- il numero (intero) di voci di cui il documento stesso è composto, non inferiore a 1;
- il totale in Euro del documento stesso, formattato col simbolo di valuta *davanti* al valore numerico, separato da uno spazio; a sua volta, il valore numerico è espresso secondo le convenzioni italiane.

Le righe successive (in numero uguale al numero di voci precedentemente specificato) riportano ciascuna:

- la tipologia della spesa (una sigla di due lettere identica alle costanti dell'enumerativo ***Tipologia***);
- il relativo importo in Euro, formattato come sopra.

ESEMPIO

```
25/01/2022;Dentista;1;€ 300,00
LP;€ 300,00
08/02/2022;Farmacia;4;€ 43,98
FC;€ 10,77
FC;€ 6,03
FC;€ 19,62
FC;€ 7,56
...
```



SEMANTICA:

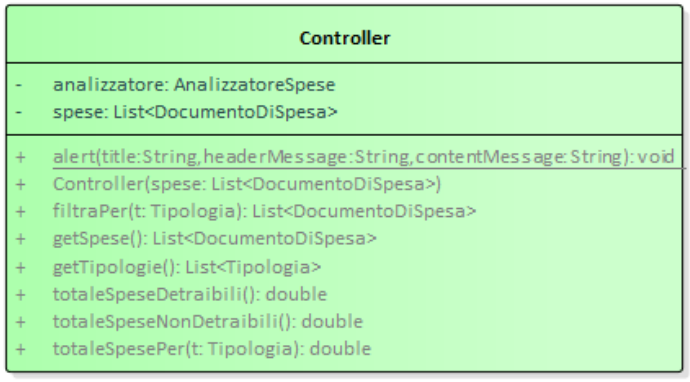
a) L'interfaccia ***SpeseReader*** (fornita) dichiara il metodo *leggiSpese* che carica da un apposito Reader (già aperto) i dati necessari, restituendo una lista di ***DocumentoDiSpesa*** che non è mai nulla. Il metodo lancia:
   - ***IllegalArgumentException*** con opportuno messaggio d'errore in caso di argomento (reader) nullo;
   - ***BadFileFormatException*** con messaggio d'errore appropriato in caso di problemi nel formato del file (mancanza/eccesso di elementi, errori nel formato delle date o dei prezzi, etc.);
   - una ***IOException*** in caso di altri problemi di I/O.

b) La classe ***MySpeseReader*** (**da realizzare**) implementa ***SpeseReader*** secondo le specifiche sopra descritte.

Parte 3 (punti: 6)

Package: spesesanitarie.controller *(punti 0)*

Il Controller è organizzato secondo il diagramma UML seguente:



SEMANTICA:

La classe ***Controller*** (fornita) riceve in fase di costruzione la lista dei ***DocumentoDiSpesa,*** che viene mantenuta nel suo stato interno unitamente all'**AnalizzatoreSpese** che si occuperà di analizzarla.

È fornita un'ampia serie di metodi per recuperare tutte le informazioni utili, molti dei quali delegano l'operazione all'analizzatore interno. In particolare *getSpese* restituisce la lista ricevuta dal costruttore, *getTipologie* restituisce la lista delle costanti definite nell'enumerativo ***Tipologia***, *getTotaleSpesePer* restituisce il totale delle spese della ***Tipologia*** specificata, *filtraPer* restituisce la lista dei soli ***DocumentoDiSpesa*** che contengono almeno una voce di spesa della ***Tipologia*** specificata.

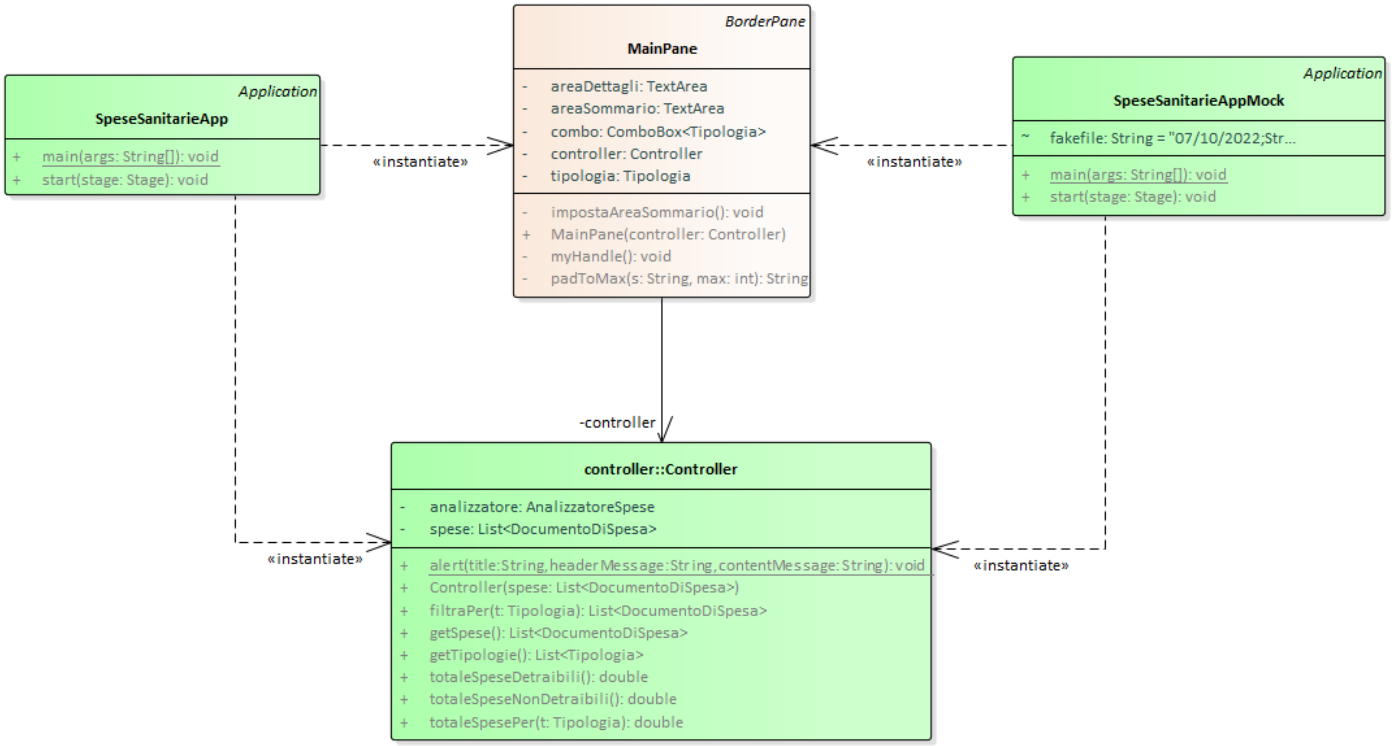
Inoltre, i due metodi *totaleSpeseDetraibili* / *totaleSpeseNonDetraibili* restituiscono, rispettivamente, il totale delle spese di tipologie detraibili / non detraibili.

Infine, il metodo statico *alert*, utilizzabile anche dal MainPane, consente di far comparire all'utente, ove occorra, una finestra di dialogo con opportuno messaggio d'errore.

Package: spesesanitarie.ui **[TEMPO STIMATO: 20-40 minuti]** *(punti 6)*

La classe ***SpeseSanitarieApp*** (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 ***SpeseSanitarieAppMock***.

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



L'interfaccia grafica si presenta come segue:

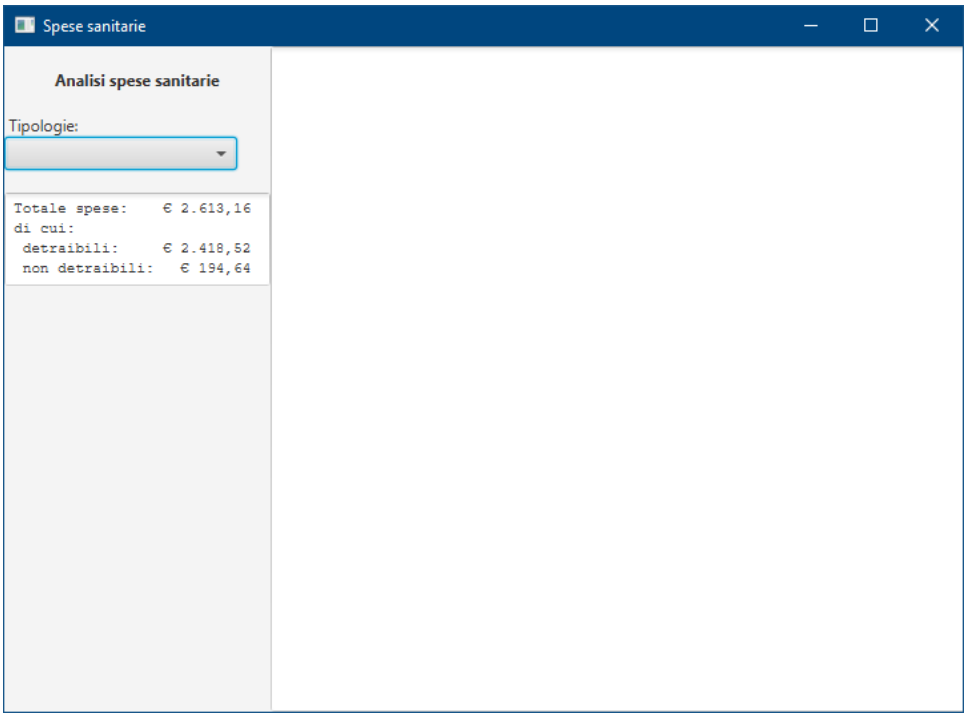
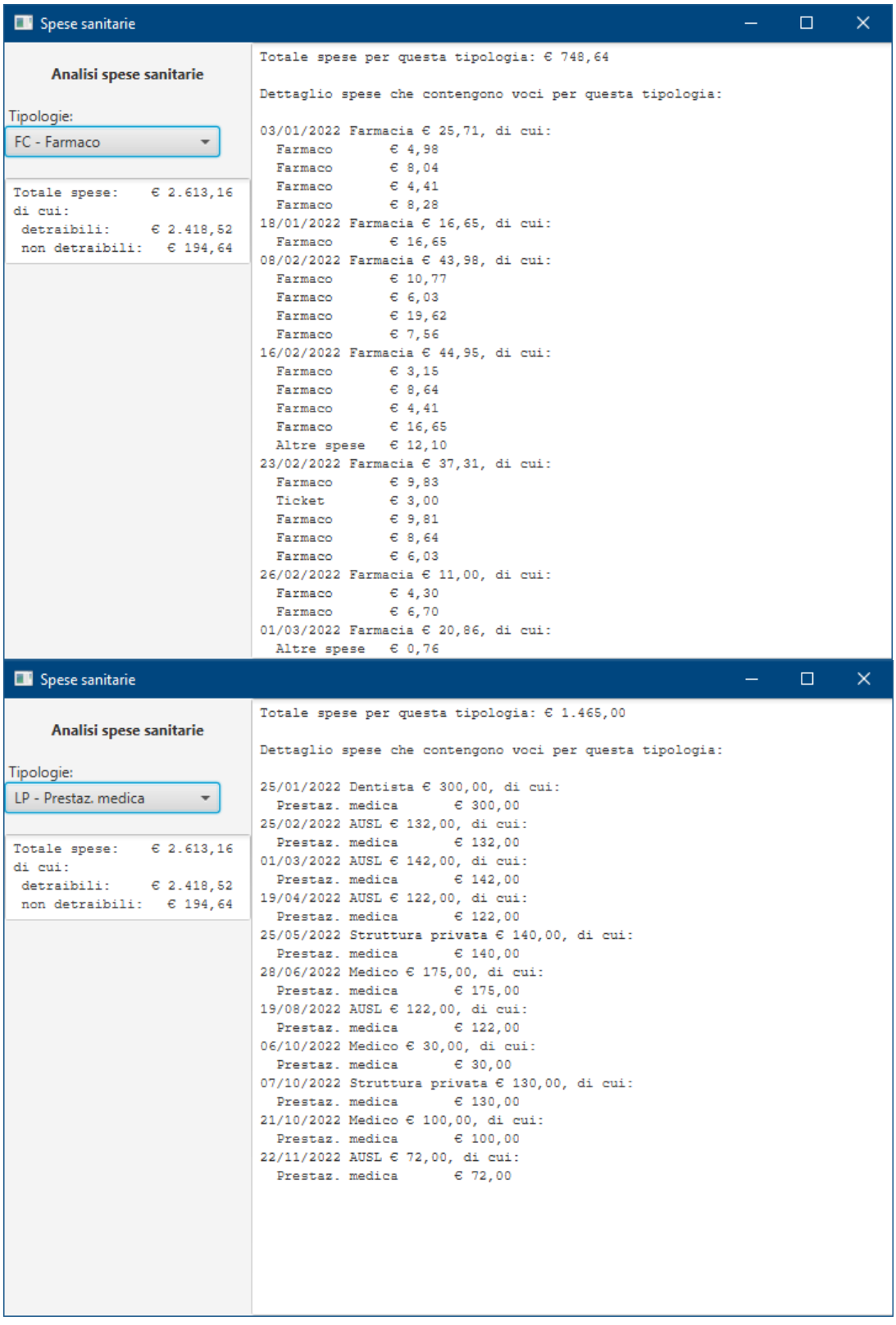


Fig. 1: la situazione iniziale della GUI, con combo tipologie vuota, prima di qualunque interazione con l'utente.

- a sinistra una combo elenca tutte le ***Tipologia*** disponibili; subito sotto, una piccola area di testo (non scrivibile dall'utente) riporta il totale delle spese, nonché i sub-totali di quelle detraibili/non detraibili;
- a destra, un'area di testo (inizialmente vuota e non scrivibile dall'utente) filtra i soli documenti di spesa che contengono almeno una voce della ***Tipologia*** selezionata.

Comportamento:

- la selezione di una ***Tipologia*** dalla combo deve causare immediatamente la visualizzazione dei soli documenti di spesa richiesti (Figg. 2,3), preceduta dal totale delle relative spese, con opportuni messaggi



Figg. 2 / 3.

Il MainPane è fornito *parzialmente realizzato*: è presente quasi tutta l'impostazione strutturale, mentre sono da completare la configurazione di alcuni componenti e la gestione degli eventi.

In particolare, ***MainPane*** estende ***BorderPane*** e prevede:

1) a sinistra, una ***VBox*** per le varie combo, aree di testo ed etichette ausiliarie
2) a destra, una ***VBox*** con la sola area di output.

La **parte da completare** riguarda:

1) la configurazione iniziale della combo
2) la configurazione dell'area di testo riassuntiva (metodo privato *impostaAreaSommario*)
3) l'aggancio dell'ascoltatore degli eventi, rappresentato dal metodo *myHandle*
4) la logica di gestione dell'evento, incapsulata nel metodo privato *myHandle*

In particolare, la gestione dell'evento principale, tramite *myHandle*, deve:

- recuperare dalla combo la tipologia selezionata e, sulla base di quella, recuperare l'elenco filtrato dei soli documenti di spesa di interesse, per le successive operazioni
- calcolare, tramite il ***Controller***, il totale delle spese per tale tipologia
- emettere nell'area di testo il totale ora calcolato (con adeguato messaggio) e, subito sotto, l'elenco dei documenti di spesa corrispondenti, *opportunamente formattati*.

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?