UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DIEM

SECONDA PROVA IN ITINERE – 21-12-2020
CORSO DI PROGRAMMAZIONE AD OGGETTI
A.A. 2020/2021
PROFF. GENNARO PERCANNELLA / LUCA GRECO

Si richiede di realizzare un'applicazione JavaFX FXML per la gestione di una TO-DO list giornaliera persistente (che recupera automaticamente al riavvio gli ultimi elementi salvati).

L'applicazione deve avere un'interfaccia grafica conforme a quanto rappresentato in Figura 1.

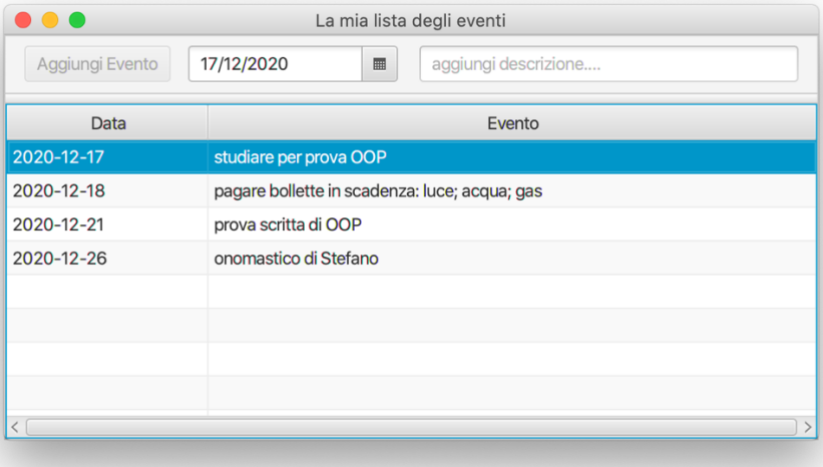

Figura 1

OVERVIEW

L'applicazione dovrà consentire l'aggiunta di eventi caratterizzati da una *data* e una *descrizione* (colonna Evento), mostrandoli **sempre** in ordine cronologico. E' possibile inserire eventi precedenti alla data corrente. Gli eventi inseriti con la stessa data vengono mostrati in ordine di inserimento. Per gli eventi inseriti è possibile editare il solo campo *descrizione* direttamente dalla tabella.

E' prevista la possibilità di rimuovere gli elementi inseriti uno per volta.

I contenuti vengono salvati automaticamente ogni 30 secondi su un file binario in maniera serializzata, in modo da consentire al successivo riavvio di riproporre in maniera automatica la lista di elementi salvati.

L'applicazione consente inoltre di importare ed esportare la lista di eventi in formato CSV.

DETTAGLI SUL COMPORTAMENTO

Al primo avvio, l'applicazione si presenta come in Figura 2. Il pulsante "Aggiungi Evento" si abilita contestualmente all'aggiunta di una descrizione. Il DatePicker punta alla data corrente. Dopo l'aggiunta di un nuovo evento il campo descrizione si ripulisce in maniera automatica, rendendo il pulsante di aggiunta disabilitato. Gli eventi inseriti compaiono in ordine crescente rispetto alla data. A parità di data, vengono mostrati in ordine di inserimento.

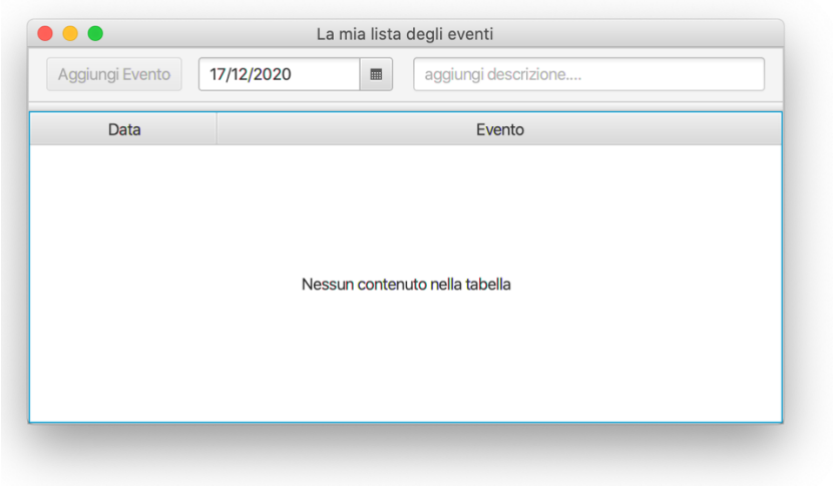

Figura 2

Quando la lista è vuota, è possibile importare una lista precedentemente salvata in formato CSV tramite il menu contestuale[1] associato alla tabella (Figura 3). Al menu contestuale si accede cliccando sul tasto destro del mouse.

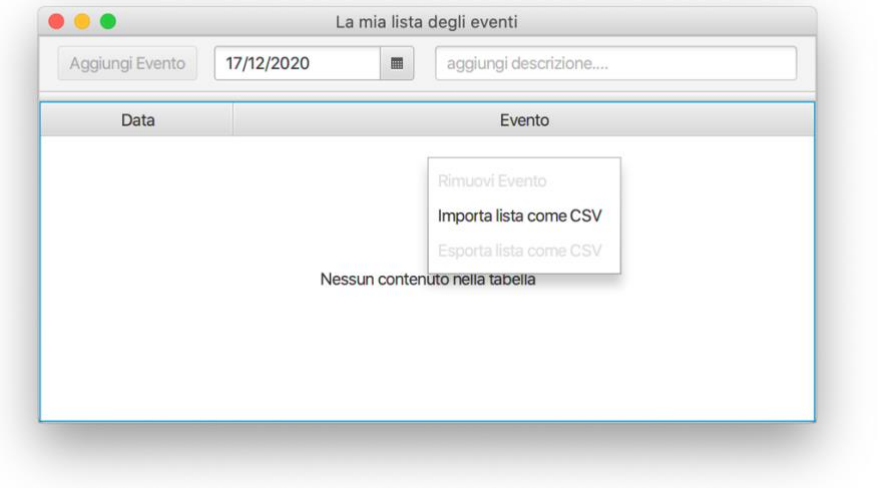

Figura 3

Oltre all'opzione di importazione, il menu contestuale prevede altre due possibilità (che appaiono disabilitate quando la tabella è vuota[2]): rimozione dell'elemento selezionato ed esportazione della lista in formato testuale CSV (Figura 4). **Il formato CSV da adottare prevede il carattere ";" come carattere di delimitazione.** Tuttavia, l'applicazione deve supportare l'inserimento di descrizioni che contengono il carattere ";" senza generare ambiguità di interpretazione (Figura 1).

Il file CSV prodotto conterrà quindi il carattere "|" in sostituzione di eventuali caratteri ";" inseriti dall'utente nel campo descrizione.

[1] Si fa riferimento al componente ContextMenu

[2] Per il comportamento descritto, può essere utile consultare la documentazione relativamente alla classe SimpleListProperty

L'importazione dei contenuti quando la tabella non è vuota comporta la sovrascrittura dei contenuti presenti.

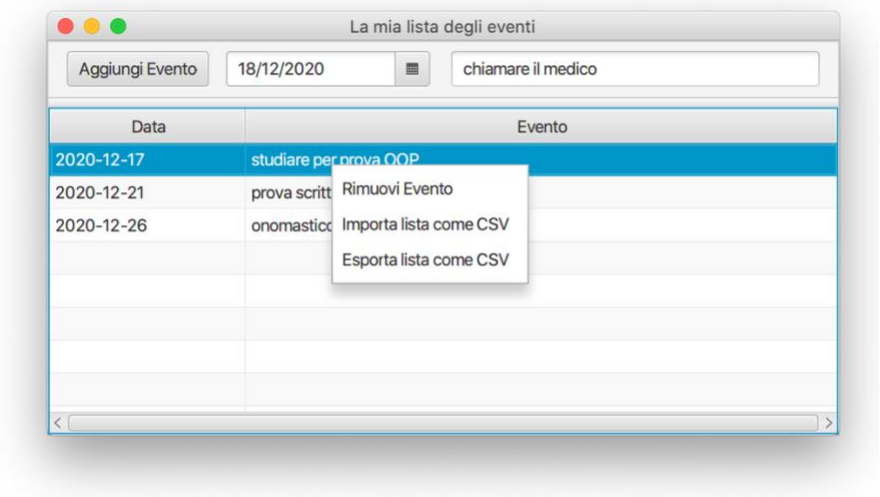

Figura 4

L'applicazione consente l'editing di eventi già caricati solo per il campo descrizione (Figura 5) tramite doppio click sul campo corrispondente.

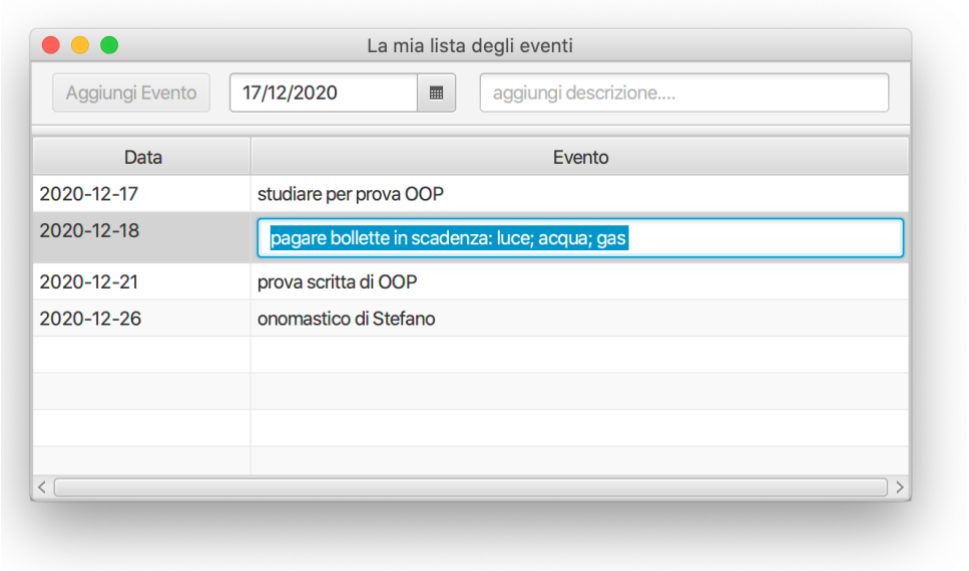

Figura 5

La funzionalità di salvataggio automatico periodico dei contenuti (ogni 30 sec) si avvia contestualmente all'avvio dell'applicazione (anche con lista vuota) e **persiste per tutto il suo utilizzo**. Il salvataggio avviene su file binario in formato serializzato. Naturalmente, al primo avvio viene creato un nuovo file.

NOTE SULL'IMPLEMENTAZIONE

- Un evento è rappresentato dalla classe *Evento* che deve presentare (almeno) gli attributi *data* e *descrizione*. Scegliere per essi un tipo di dato che si ritiene opportuno. L'evento può presentare come ulteriore attributo un *id* numerico, impostato automaticamente in fase di creazione dell'evento. La classe Evento deve implementare l'interfaccia *Comparable*, specificando quale relazione d'ordine naturale tra gli eventi la *data*; eventi che presentano la stessa data verranno confrontati rispetto all'*id* numerico univoco. Prevedere i getter e i setter per tutti gli attributi.
- La logica di funzionamento dell'applicazione è gestita da una classe *Controller* che implementa l'interfaccia *Initializable*. Il metodo astratto *initialize* dovrà essere implementato allo scopo di inizializzare tutti gli elementi utili alla creazione dell'interfaccia e per gestire eventuali operazioni preliminari necessarie per il corretto funzionamento. Il controller conterrà inoltre i metodi di gestione degli eventi associati ai componenti grafici per realizzare i comportamenti richiesti.
- La funzionalità di salvataggio automatico temporizzato andrà realizzata a partire dall'implementazione di una classe (da denominare *TimedSaving*) che implementa l'interfaccia *Runnable*.
- Dovranno inoltre essere prodotti: il file FXML che contiene le specifiche della *View* e la classe principale dell'applicazione JavaFX da denominare *ToDoList*.

PRIMA DI CREARE IL PROGETTO LEGGERE LE ISTRUZIONI PER LA CONSEGNA!!!

ISTRUZIONI PER LA CONSEGNA

Per almeno una delle classi prodotte, deve essere presente un commento in formato Javadoc della classe e di ciascuno dei suoi metodi pubblici (si veda il seguente link per una descrizione del formato Javadoc).

https://www.oracle.com/technetwork/articles/java/index-137868.html

Il progetto dovrà essere impostato in modo che tutte le classi prodotte siano in un package denominato "**gruppoXX**"

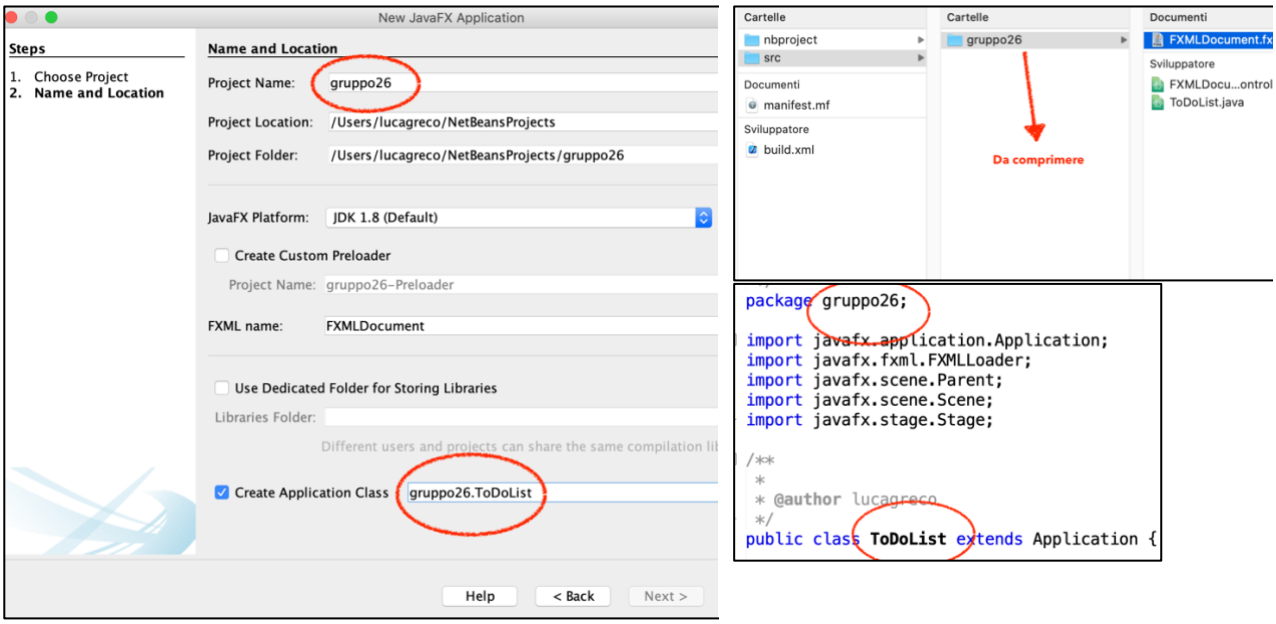

Figura 6 Esempio di impostazione per il gruppo26 in NetBeans IDE

Per la consegna, la cartella di base contenente **i file sorgenti** (`.java` e `.fxml`) di tutto il progetto (quindi, la cartella "gruppoxx"), deve essere compressa in formato .ZIP (attenzione, non in formato .RAR o altri formati di compressione), e deve essere rinominata "gruppoXX.zip", dove XX è il numero del gruppo (esempio: gruppo09.zip).

Solo uno dei membri di ciascun gruppo deve effettuare la consegna sulla piattaforma.