



Università degli Studi di Salerno - Dipartimento DIEM  
**Corso di Programmazione Java Avanzata**

## Concurrency in JavaFX - Sync Assignment

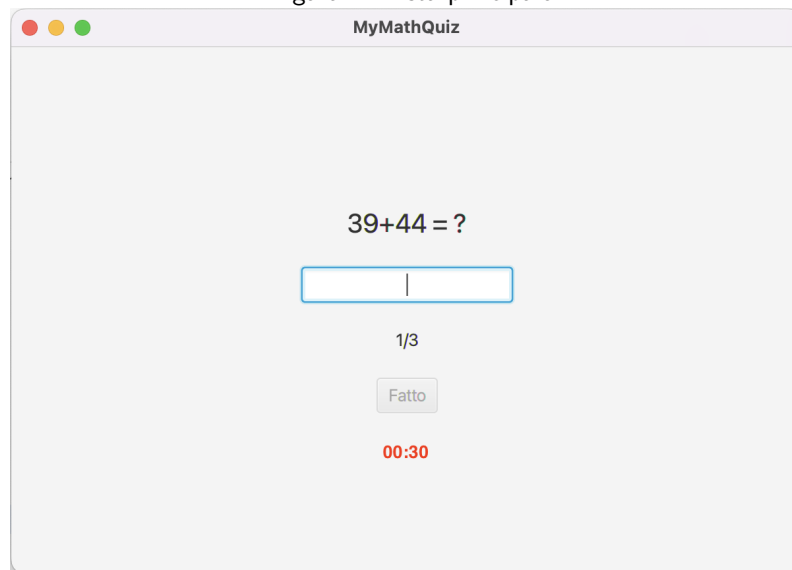
Prof. Luca Greco

A.A. 2024/2025

### 1 Overview

Si richiede di realizzare un'applicazione Java con interfaccia grafica (JavaFX) per la gestione di quiz numerici. L'applicazione deve avere un'interfaccia grafica conforme a quanto rappresentato in Figura 1, con finestra principale di dimensioni 600px X 400px.

Figura 1: Vista principale



L'applicazione dovrà generare le domande in maniera randomica, prevedendo le sole operazioni di addizione e sottrazione tra interi<sup>1</sup>.

Dopo una fase di accounting dell'utente, l'applicazione dovrà mostrare le domande una per volta raccogliendo le risposte mediante un textfield. Infine, dovrà mostrare la tabella di riepilogo indicando per ciascuna domanda il risultato conseguito dall'utente e l'esito (corretto o sbagliato). Il riepilogo dovrà poter essere esportato su un file di testo.

<sup>1</sup>Per semplificare il testing si può scegliere di generare valori con un bound di propria scelta (esempio da 0 a 50).

## 2 Dettagli sul comportamento

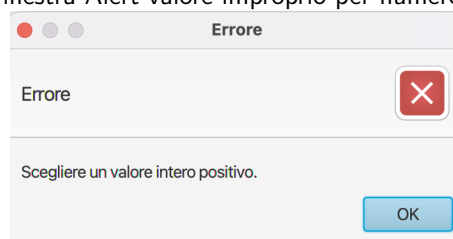
All'avvio l'applicazione presenta una schermata di accounting dove è richiesto all'utente di inserire nome e cognome e il numero di domande da generare (Figura 2). Il bottone "Inizia" si abilita contestualmente all'inserimento dei valori richiesti. In caso di valore improprio per il numero di domande da generare (es. numero negativo o testo non convertibile in numero), viene mostrata una finestra Alert come riportato in Fig. 3.

Figura 2: Schermata iniziale



Cliccando sul bottone "Inizia", l'interfaccia si aggiorna mostrando la prima domanda. L'applicazione presenta quindi una schermata<sup>2</sup> come quella riportata in Fig. 1. L'utente potrà inserire la risposta mediante un TextField. Se il valore fornito non è un valore di tipo appropriato, l'applicazione mostra una finestra Alert segnalando di correggere il valore inserito. Nella schermata è presente una Label che mostra lo stato di completamento del questionario, indicando l'indice (one-based) della domanda corrente rispetto al numero di domande totali: nell'esempio in Figura 1 si sta rispondendo alla prima di 3 domande generate dal sistema. Viene anche mostrato un timer<sup>3</sup> (conto alla rovescia) di 30 secondi, allo scadere del quale il sistema mostra la domanda successiva imputando una risposta sbagliata.

Figura 3: Finestra Alert valore improprio per numero di domande.



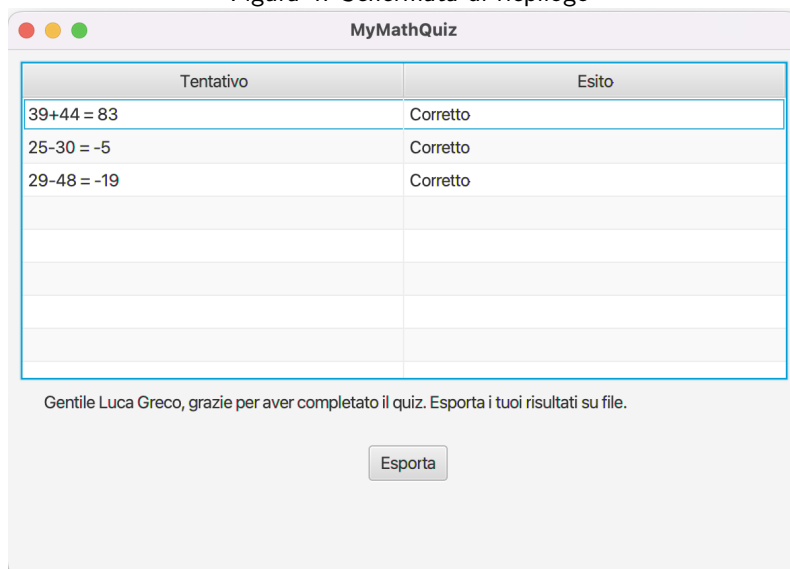
Quando si clicca sul pulsante "Avanti", l'applicazione mostra la domanda successiva (aggiornando la label di avanzamento). Al termine del questionario è mostrata la tabella di riepilogo delle risposte date. La prima colonna riporta il tentativo, la seconda fornisce indicazione della correttezza della risposta data dall'utente (Figura 4).

Il pulsante "Esporta" consente di esportare l'esito dettagliato del quiz su un file di testo (usare estensione ".txt"), il cui nome può essere specificato dall'utente tramite una finestra di salvataggio [FileChooser](#). Il file di testo prevede campi separati da tabulazione ed è strutturato come segue:

<sup>2</sup>Si segnala il componente [StackPane](#) da usare come radice del grafo della scena per la gestione di schermate alternative.

<sup>3</sup>Si segnala il componente [Timeline](#)

Figura 4: Schermata di riepilogo



The screenshot shows a window titled "MyMathQuiz" with a table of results. The table has two columns: "Tentativo" (Attempt) and "Esito" (Result). It contains three rows of data, all marked as "Corretto" (Correct). Below the table, there is a message in Italian and an "Esporta" (Export) button.

Tentativo	Esito
39+44 = 83	Corretto
25-30 = -5	Corretto
29-48 = -19	Corretto

Gentile Luca Greco, grazie per aver completato il quiz. Esporta i tuoi risultati su file.

Esporta

```
TENTATIVO;RISULTATO CORRETTO;ESITO
39+44 = 83;83;Corretto
25-30 = -5;-5;Corretto
29-48 = -19;-19;Corretto
```

### 3 Note sull'implementazione

- Una domanda numerica è rappresentata dalla classe `NumericQuestion` che deve presentare almeno gli attributi `num1`, `num2` - gli operandi interi - e `operator` - un carattere che rappresenta l'operatore (+ o -). Scegliere per essi un tipo di dato che si ritiene opportuno, prevedendo accesso privato. Prevedere getter per tutti gli attributi. Prevedere inoltre un metodo `randomInit()` che consente di inizializzare gli operandi e l'operatore in maniera randomica (in modo da generare una domanda random). Può essere utile predisporre un metodo `getResult()` che restituisce il risultato dell'operazione. Ridefinire in maniera conveniente il metodo `toString()`.
- Un tentativo di risposta alla domanda da parte dell'utente è rappresentato dalla classe `NumericQuestionAttempt` che ha come attributi `question` di tipo `NumericQuestion` e `givenAnswer` - la risposta numerica formulata dall'utente come esatta. Scegliere i tipi di dato opportuni. Oltre ai metodi getter di utilità per la visualizzazione, la classe `NumericQuestionAttempt` dovrà prevedere almeno il metodo `boolean isCorrect()` - che verifica se la risposta data corrisponde a quella esatta - e il metodo `String getResult()` - che restituisce la stringa "Corretto" o "Sbagliato" sulla base della correttezza della risposta fornita.
- L'applicazione va progettata seguendo il pattern MVC. 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. Dovranno inoltre essere prodotti: il file FXML che contiene le specifiche della View e la classe principale dell'applicazione JavaFX da denominare `MyMathQuizApp`.

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

## 4 Istruzioni per la consegna

**Rispettare la nomenclatura per le classi definita nella sezione precedente.**

Il progetto dovrà essere impostato in modo che tutte le classi prodotte siano in un package denominato "gruppoGX", senza ulteriori livelli:

Es. ciascun file sorgente del progetto sviluppato dal gruppo G1 dovrà contenere l'istruzione:

```
package gruppoG1;
```

Per la consegna, la cartella di base contenente i file sorgenti (.java e .xml) di tutto il progetto (quindi, la cartella "gruppoGX"), deve essere compressa in formato .ZIP (attenzione, non in formato .RAR o altri formati di compressione), e deve essere rinominata "gruppoGX.zip".