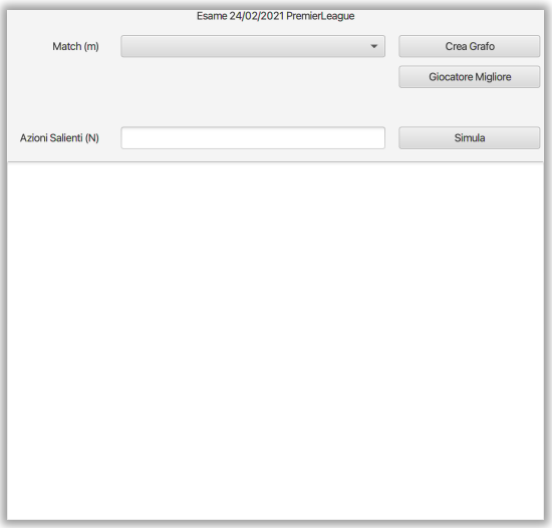
Tema d'esame 24/02/2021 - PremierLeague

Si consideri il database **PremierLeague.sql**, contenente informazioni sul campionato di calcio inglese della stagione 2011/2012. Il database è strutturato secondo il diagramma ER riportato nell'ultima pagina. Si intende costruire un'applicazione JavaFX che permetta di interrogare tale base dati. L'applicazione dovrà svolgere le seguenti funzioni:

PUNTO 1

a. Permettere all'utente di selezionare, dall'apposita tendina, un Match (*m*) tra tutti quelli inclusi nel database.

b. Alla pressione del bottone "Crea Grafo", si crei un grafo **semplice, pesato e orientato** in cui nodi siano tutti i giocatori che hanno preso parte alla partita *m*. A tal proposito, si considerino i giocatori che compaiono nella tabella ***Actions***: ogni riga di questa tabella riassume le statistiche di gioco di un particolare giocatore per una determinata partita.



c. Sempre considerando la tabella ***Actions***, si definisca l'efficienza e_i di un giocatore *i* all'interno della partita *m* come la somma dei suoi passaggi riusciti (*SP*, campo ***totalSuccessfulPassesAll***) e dei suoi assists (*AS*, campo ***assists***), divisa per il numero di minuti giocati dallo stesso giocatore (*TP*, campo ***timePlayed***):

$$e_i(m) = \frac{SP_i(m) + AS_{i(m)}}{TP_i(m)}$$

d. Nel creare il grafo, un giocatore *i* va collegato tutti i giocatori *avversari j* affrontati da *i* durante la partita selezionata *m*. Dati due giocatori *i* e *j*, il peso dell'arco, sempre positivo, rappresenta la differenza Δ tra le rispettive efficienze:

$$\Delta = |\, e_i(m) - e_j(m) \,|$$

L'arco deve essere orientato dal giocatore con efficienza maggiore efficienza verso quello con minor efficienza.

e. Alla pressione del bottone "Giocatore Migliore", stampare il giocatore il cui Δ complessivo di efficienza sia massimo. A questo proposito si calcoli il Δ complessivo di efficienza di un giocatore come la differenza tra la somma dei pesi dei suoi archi uscenti e la somma dei pesi dei suoi archi entranti. Si stampino sia il giocatore che il Δ calcolato.

Suggerimento. In un grafo orientato, è possibile utilizzare i seguenti metodi:

- *outDegreeOf*: restituisce il numero di archi uscenti da un determinato vertice;
- *inDegreeOf*: restituisce il numero di archi entranti in un determinato vertice;
- *outgoingEdgesOf*: restituisce il set di archi uscenti da un determinato vertice;
- *incomingEdgesOf*: restituisce il set di archi entranti in un determinato vertice;
- *Graphs.successorListOf*: restituisce la lista di vertici raggiunti dagli outgoingEdges;
- *Graphs.predecessorListOf*: restituisce la lista di vertici raggiunti dagli incomingEdges.

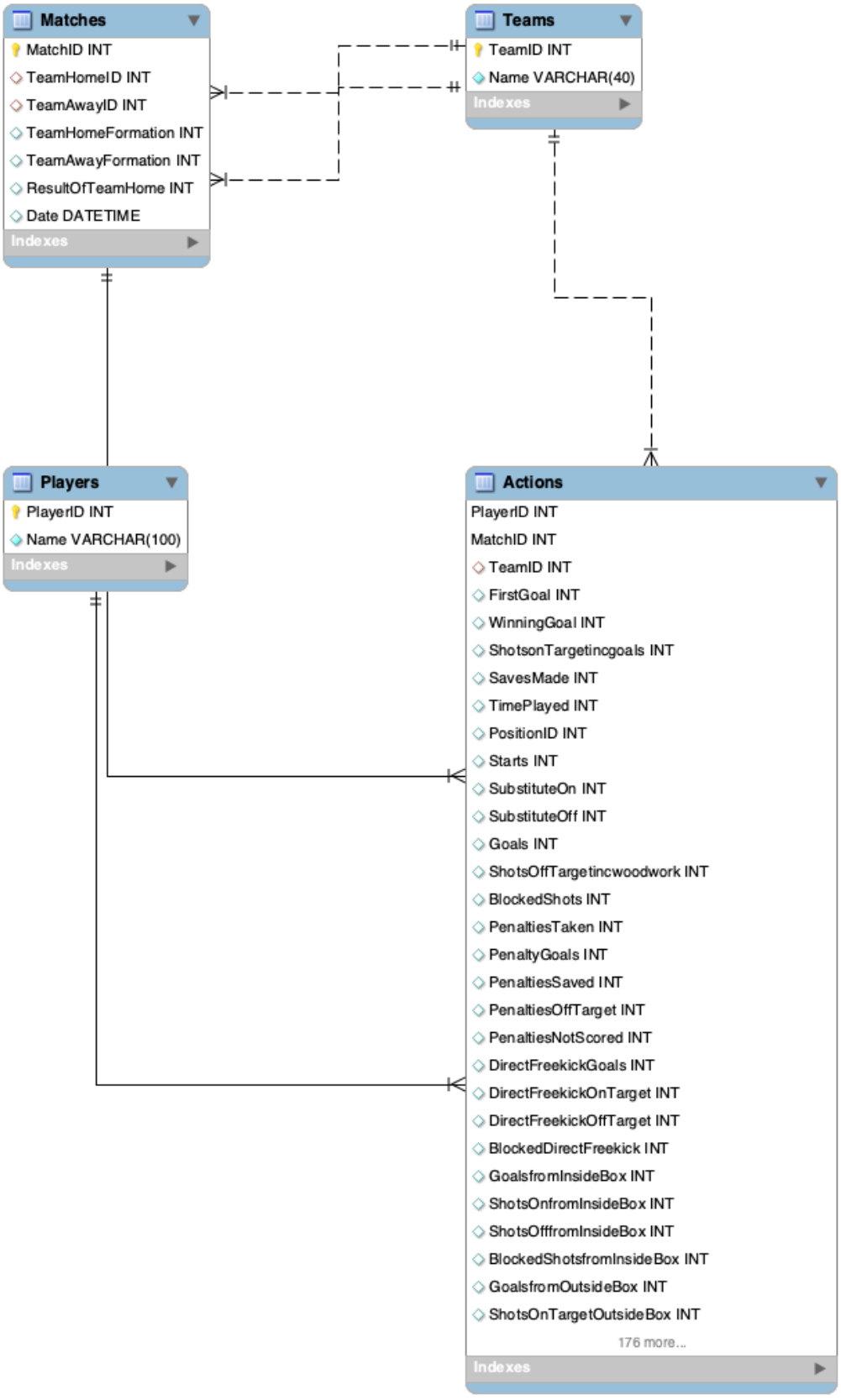
PUNTO 2

a. Si vuole esterndere il programma per effettuare delle simulazioni delle partite del campionato di calcio inglese. A tal proposito, si permetta all'utente di inserire, nell'apposita casella di testo, un numero *N* di azioni salienti da simulare per la partita *m* inclusa nel grafo.
b. Alla pressione del bottone "Simula", simulare *N* azioni salienti della partita nel seguente modo (N.B.: le azioni sono da considerarsi "inventate", e non hanno alcuna relazione con le tabelle incluse nel database):
    I. con una probabilità del 50%, un'azione saliente si traduce in un **GOAL** di una delle due squadre coinvolte. La squadra che segna il goal è sempre quella che ha più giocatori in campo in quel momento. Durante una determinata azione saliente, in particolare, ogni squadra ha in campo 11 giocatori meno i suoi eventuali espulsi nelle azioni precedenti. Nel caso in cui le due squadre abbiano lo stesso numero di giocatori attualmente in campo, ad esempio perché non ci sono ancora state espulsioni, la squadra che segna il goal è quella a cui appartiene il giocatore migliore (vedere punto 1.e).
    II. con una probabilità del 30%, un'azione saliente si traduce in una **ESPULSIONE** di un giocatore di una delle squadre coinvolte. Al 60%, l'espulsione coinvolge la squadra a cui appartiene il giocatore migliore (vedere punto 1.e), mentre al 40% coinvolge un giocatore dell'altra squadra.
    III. Nel rimanente 20% dei casi, un'azione saliente si traduce in un **INFORTUNIO** di uno dei giocatori in campo. Quando si verifica un infortunio, l'arbitro è costretto ad allungare il recupero concesso alla fine della partita, e il numero *N* di azioni salienti da simulare cresce, con eguale probabilità, di 2 o 3 unità.
c. Al termine della simulazione, stampare il risultato della partita (goal di entrambe le squadre) e il numero di espulsi delle due squadre.

Nella realizzazione del codice, si lavori a partire dalle classi (Bean e DAO, FXML) e dal database contenuti nel progetto di base. È ovviamente permesso aggiungere o modificare classi e metodi.

Tutti i possibili errori di immissione, validazione dati, accesso al database, ed algoritmici devono essere gestiti, non sono ammesse eccezioni generate dal programma.

Le tabelle **Teams**, **Players** e **Matches** contengono rispettivamente informazioni su squadre, giocatori e partite. Ogni riga della tabella **Actions**, invece, contiene le informazioni statistiche di un determinato giocatore in una determinata partita (minuti giocati, goal fatti, ecc.)



ESEMPI DI RISULTATI PER CONFRONTARE LA PROPRIA SOLUZIONE