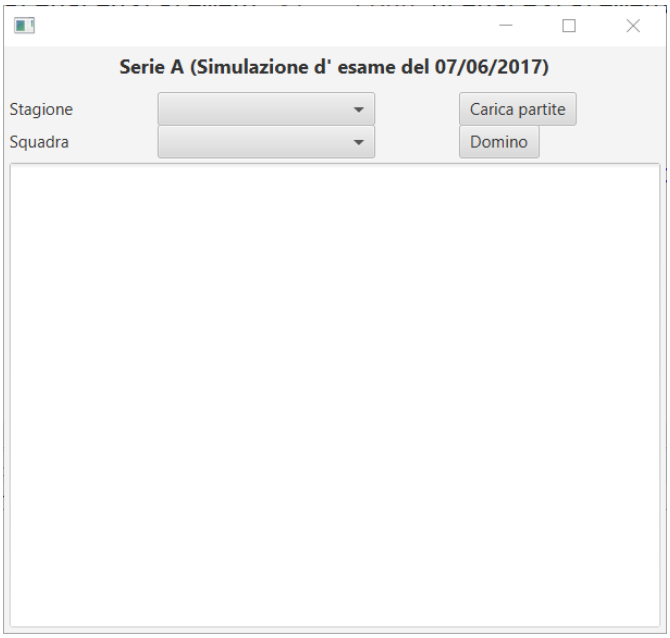
Simulazione d'esame del 07/06/2017



Si consideri il database "serie_a", contenente i risultati di tutte le partite di calcio della Serie A italiana, tra la stagione 2002/2003 e la stagione 2016/2017. Il database è strutturato secondo il diagramma ER della pagina seguente.

Si intende costruire un'applicazione JavaFX che permetta di interrogare tale base dati, e calcolare informazioni a proposito delle gare disputate. L'applicazione dovrà svolgere le seguenti funzioni:

PUNTO 1

a. Permettere all'utente di specificare una stagione di interessa (tramite menu a tendina) e di richiedere il caricamento delle partite.
b. Creare un grafo che rappresenti i risultati delle partite giocate nella stagione selezionata dall'utente. Il grafo dovrà essere orientato e pesato, con i vertici che rappresentino l'insieme di squadre che ha giocato nella stagione indicata, e gli archi che rappresentino il risultato della partita. Il peso dell'arco tra TeamA e TeamB deve valere +1 se TeamA ha battuto TeamB, 0 se hanno pareggiato, -1 se TeamA ha perso.
c. Analizzare il grafo e calcolare la classifica finale del campionato, assegnando 3 punti alle vittorie, 1 ai pareggi e 0 alle sconfitte. Stampare la classifica finale, in ordine decrescente di punteggio.

PUNTO 2

a. Quando l'utente seleziona la funzione "Domino", occorre ricercare e stampare la più lunga sequenza di partite "concatenate", in cui ciascuna squadra sconfitta in una partita diviene vincitrice nella partita successiva. Ad esempio, la sequenza potrà essere: TeamA, TeamC, TeamK, se nella partita TeamA-TeamC è risultato vincitore TeamA, e nella partita TeamC-TeamK è risultato vincitore TeamC.
b. Suggerimento: dal punto di vista del grafo, occorre trovare un *cammino (aperto)*, di lunghezza massima, che attraversi unicamente archi con peso +1. Gli archi con peso 0 oppure -1 non devono essere considerati.
c. Si noti che il cammino potrebbe non essere semplice (cioè lo stesso vertice potrebbe comparire più volte). Ad esempio: A-B, B-C, C-D, D-A, A-E, E-F. In questo caso A ha vinto su B e su E, ma ha perso da D. In ogni caso, ogni arco può essere utilizzato una sola volta (ad esempio, abbiamo potuto usare A-E, ma non sarebbe stato lecito ripetere di nuovo A-B).
d. Al termine della ricerca, si stampi tale cammino.

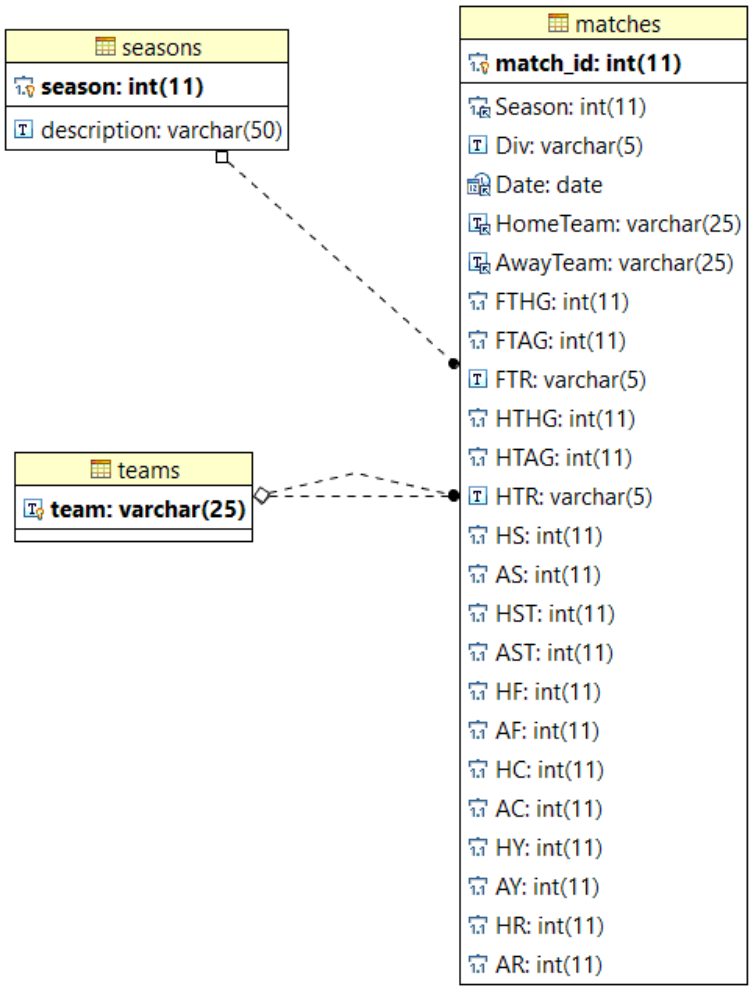
Nella realizzazione del codice, si lavori a partire dalle classi (Bean e DAO, FXML) e dal database contenuti nel progetto di base. È ovviamente permesso aggiungere o modificare classi e metodi.

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

Legenda (tabella `matches`):

- `Season` = Season year
  (foreign key to table `seasons`)
- `Div` = League Division
- `Date` = Match Date
- `HomeTeam` = Home Team
  (foreign key to table `teams`)
- `AwayTeam` = Away Team
  (foreign key to table `teams`)
- `FTHG` = Full Time Home Team Goals
- `FTAG` = Full Time Away Team Goals
- `FTR` = Full Time Result
  (**H**=Home Win, **D**=Draw, **A**=Away Win)
- `HTHG` = Half Time(*) Home Team Goals
- `HTAG` = Half Time(*) Away Team Goals
- `HTR` = Half Time(*) Result
  (**H**=Home Win, **D**=Draw, **A**=Away Win)

(*) tranne quanto la partita è assegnata a tavolino



Parametri aggiuntivi (non presenti in tutte le stagioni):

- HS = Home Team Shots
- AS = Away Team Shots
- HST = Home Team Shots on Target
- AST = Away Team Shots on Target
- HHW = Home Team Hit Woodwork
- AHW = Away Team Hit Woodwork
- HC = Home Team Corners
- AC = Away Team Corners
- HF = Home Team Fouls Committed
- AF = Away Team Fouls Committed
- HO = Home Team Offsides
- AO = Away Team Offsides
- HY = Home Team Yellow Cards
- AY = Away Team Yellow Cards
- HR = Home Team Red Cards
- AR = Away Team Red Cards