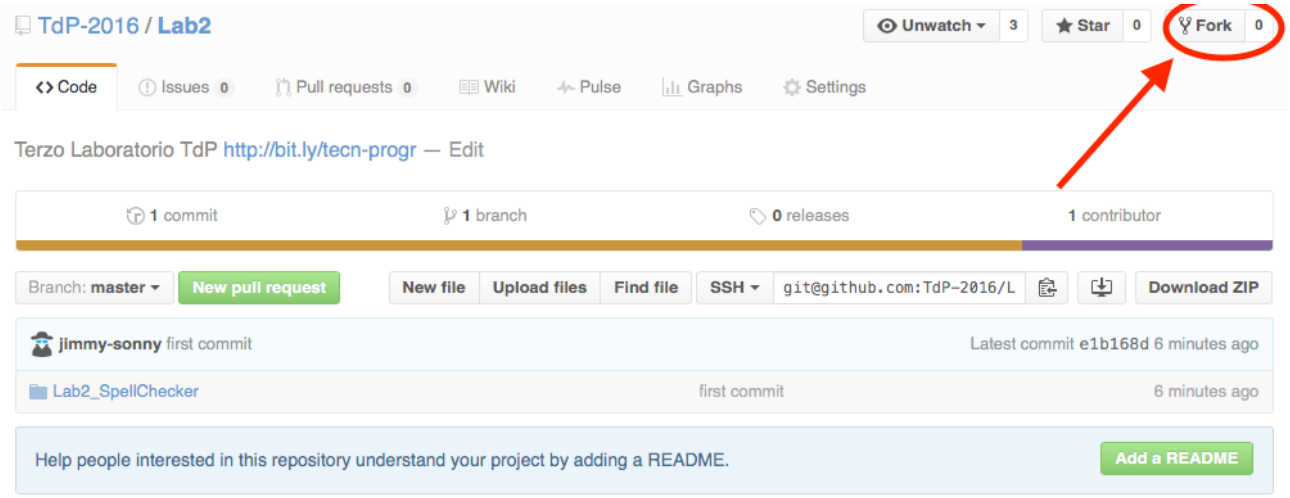
03FYZ TECNICHE DI PROGRAMMAZIONE
Istruzioni per effettuare il fork di un repository GitHub

- Effettuare il login su GitHub utilizzando il proprio username e password.
- Aprire il repository su GitHub relativo al decimo laboratorio:

  https://github.com/TdP-2016/Lab9
- Utilizzare il pulsante *Fork* in alto a destra per creare una propria copia del progetto.



L'azione di Fork crea un nuovo repository nel proprio accout GitHub con una copia dei file necessari per l'esecuzione del laboratorio.

- Aprire Eclipse, andare su *File -> Import*. Digitare *Git* e selezionare *Projects from Git -> Next -> Clone URI -> Next*.
- Utilizzare la URL del **proprio** repository che si vuole clonare (**non** quello in TdP-2016!), ad esempio:

  https://github.com/my-github-username/Lab9
- Fare click su *Next*. Selezionare il branch (*master* è quello di default) fare click su *Next*.
- Selezionare la cartella di destinazione (quella proposta va bene), fare click su *Next*.
- Selezionare *Import existing Eclipse projects*, fare click su *Next* e successivamente su *Finish*.
- Il nuovo progetto Eclipse è stato clonato ed è possibile iniziare a lavorare.
- A fine lavoro ricordarsi di effettuare Git commit e push, utilizzando il menù *Team in Eclipse*.

ATTENZIONE: solo se si effettua Git **commit** e successivamente Git **push** le modifiche locali saranno propagate sui server GitHub e saranno quindi accessibili da altri PC e dagli utenti che ne hanno visibilità.

03FYZ TECNICHE DI PROGRAMMAZIONE

Esercitazione di Laboratorio 09 – 11 maggio 2016

Obiettivi dell'esercitazione:

- Utilizzo della libreria JGraphT
- Cicli
- Connection Pooling

ESERCIZIO 1:

Scopo dell'esercitazione: Modificare l'applicazione JavaFX del Laboratorio 08 per visualizzare in output un percorso ciclico che tocchi il maggior numero di stazioni della metropolitana di Parigi.

Esercizio 1.1 Aggiungere nell'interfaccia grafica un pulsante *"Calcola ciclo"*. Utilizzare la libreria JGraphT per visualizzare in output la lista delle fermate e delle linee della metropolitana da prendere per percorrere il ciclo con maggior numero di stazioni nel grafo.

Esercizio 1.2 Modificare la classe *DBConnect.java* per sfruttare il meccanismo del "connection pooling" per la connessione al database facendo uso della libreria **c3p0**. [Fare riferimento: alle slide del corso (pg. 63): http://goo.gl/PdP32B ed all'esercizio svolto a lezione: https://goo.gl/zHjGCr]

Esercizio 1.3 (opzionale) Implementare la soluzione proposta da Giovanni Squillero a lezione per risolvere l'Esercizio 2 del Laboratorio 8.

Descrizione: Passare dalla rappresentazione in *Fig 1* a quella in *Fig 2*. Per ogni oggetto *Fermata* aggiungere due oggetti *FermataSuLinea* virtuali (P ed A) collegati come in *Fig. 2* utilizzando due linee "virtuali" con peso 0. Il termine "virtuale" è inteso come fermata o linea non esistente nella reale rete metropolitana.

Supponendo di voler calcolare l cammino minimo tra la stazione "Fermata 1" e "Fermata 2" con questa rappresentazione è possibile chiamare una sola volta l'algoritmo di Dijkstra utilizzando come stazione di partenza "Fermata1/P" e come stazione di arrive "Fermata2/A".

Soluzione: https://goo.gl/YLCbw5

Fig 1

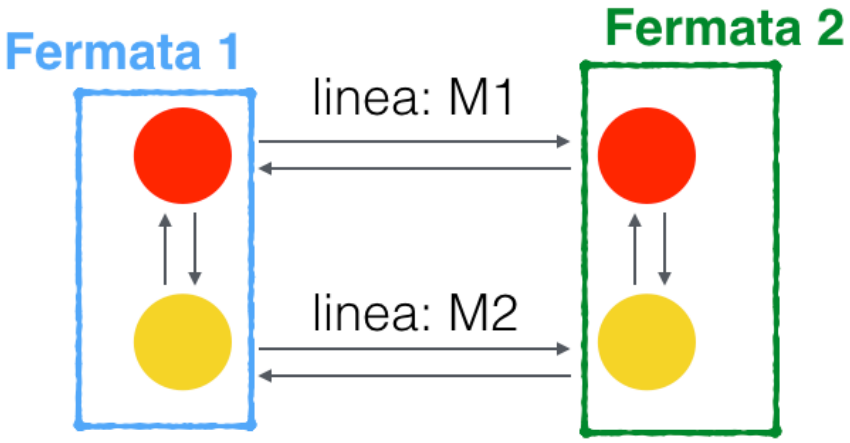
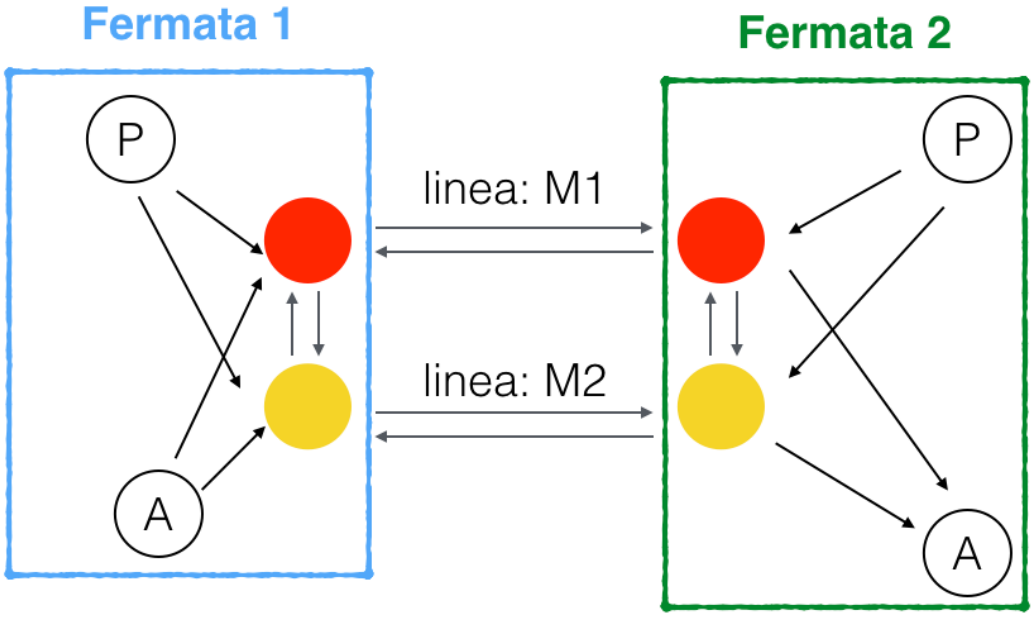

Fermata 1
Fermata 2
linea: M1
linea: M2

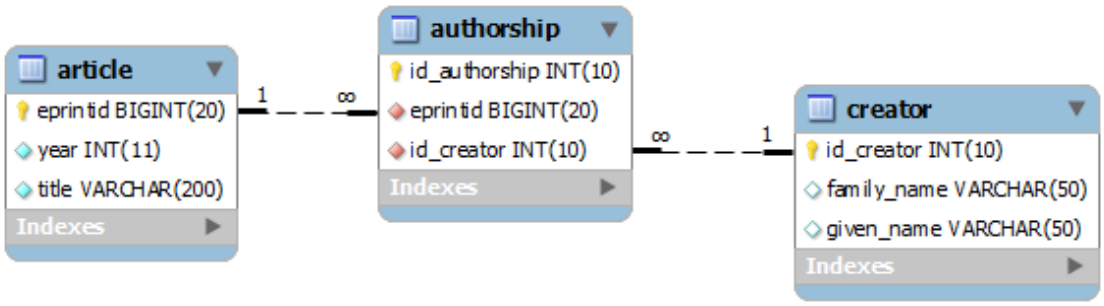
Fig 2


Fermata 1
Fermata 2
P
P
linea: M1
linea: M2
A
A

ESERCIZIO 2:

Si consideri il data-set **porto.sql** [GitHub - DB/porto.sql] che contiene una lista di alcune pubblicazioni scientifiche del Politecnico di Torino. In *Fig 1* è mostrata la struttura delle tabelle.

Fig 1



Scopo dell'esercitazione: Realizzare un'applicazione in JavaFX che permetta di visualizzare alcune informazioni estratte dal data-set delle pubblicazioni: coaturi, numero delle componenti connesse nel grafo delle pubblicazioni, articoli di collegamento tra autori non coautori.

Descrizione dall'applicazione: L'interfaccia grafica dell'applicazione deve presentare due menù a tendina con la lista degli autori, tre pulsanti ("*Visualizza coautori*", "*Visualizza cluster*", "*Visualizza articoli*") ed una TextArea in cui stampare l'output del programma. L'applicazione deve fornire le seguenti funzionalità:

1) L'utente seleziona in uno dei due menù a tendina un *autore*. Facendo click sul pulsante "*Visualizza coaturi*" nell'area di testo sottostante vengono visualizzati tutti gli autori con cui l'autore selezionato ha scritto almeno un articolo.

2) L'utente non seleziona nessun autore nei due menù a tendina. Facendo click sul pulsante "*Visualizza Cluster*" nell'area di testo sottostante viene stampato il numero di componenti connesse all'interno del grafo e gli autori appartenenti a ciascuna di esse.

3) L'utente seleziona due autori dai due menù a tendina. Facendo click sul pulsante "*Visualizza articoli*" vengono visualizzati gli articoli che collegano i due autori (non coaturi) selezionati.

Esercizio 1.1 Realizzare un'interfaccia grafica con *JavaFx* simile a quella descritta precedentemente. Strutturare il programma secondo i pattern MVC e DAO come spiegato a lezione. Fare uso del meccanismo del "connection pooling".

Esercizio 1.2 Utilizzare il file **porto.sql** [GitHub - DB/porto.sql] per ottenere le informazioni su articoli ed autori delle pubblicazioni del Politecnico di Torino.

IMPORTANTE: prestare particolare attenzione a sostituire le chiavi primarie ed esterne SQL con le regole di Object-Relational Mapping degli oggetti JavaBean salvati in memoria. [pg. 56 in http://goo.gl/PdP32B]

Esercizio 1.3 Utilizzando la libreria *JGraphT* per creare un grafo in cui ogni autore rappresenta un vertice ed un arco collega due autori solo se sono coautori dello stesso articolo (utilizzare le informazioni della tabella *authorship*).

Esercizio 1.4 Implementare le funzionalità 1, 2 e 3 sopra proposte.