# 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 quarto laboratorio:
  - https://github.com/TdP-2022/Lab04
- 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 account 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-2022!), ad esempio:
  - https://github.com/my-github-username/Lab04
- 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 29/30 Marzo 2022

## Obiettivi dell'esercitazione:

- Utilizzo del Pattern MVC
- Utilizzo di JDBC
- Introduzione al Pattern DAO

#### ESERCIZIO 1

Dopo aver fatto il fork del progetto relativo al quarto laboratorio, realizzare in linguaggio Java un'applicazione dotata di interfaccia grafica che permetta la gestione degli studenti e dei corsi di ateneo. L'applicazione permette di cercare informazioni relative ad un corso o ad uno studente e di iscrivere uno studente ad un corso. Prestare attenzione alla gestione di tutte le possibili condizioni di errore.

## Punto 1

Realizzare un'interfaccia grafica con *JavaFx* simile a quella mostrata in *Fig 1*. I campi *Nome* e *Cognome* sono dei *TextField* non editabili. Il pulsante *Reset* permette di cancellare il contenuto di tutti i campi.

Utilizzare il database *iscritticorsi.sql* (presente nella cartella del progetto), per <u>popolare il menù a tendina con tutti i corsi</u> disponibili. Inserire anche un campo vuoto da utilizzare quando non si vuole selezionare nessun corso in particolare.

Fare uso dei pattern MVC e DAO, come spiegato a lezione.

Ritorna uno studente : public Studente get Studente<br/>(int matricola)  $\{\ldots\}$ 

#### Punto 2

Implementare la <u>funzionalità di completamento automatico</u>: data la matricola di uno studente, facendo click sul pulsante verde adiacente, completare in automatico il nome e cognome di uno studente. (Esempio *Fig. I*)

public List<Studente> getStudentilscrittiAlCorso(Corso corso){....}

## Punto 3

Quindi ritornerà una lista di studenti iscritti al corso. List<Studente> non un void. Vado nella classe CorsoDAO; Implementare la <u>ricerca degli studenti iscritti ad un corso</u>: selezionato un corso dal menù a tendina, facendo click sul pulsante *Cerca iscritti corso*, vengono visualizzati tutti gli studenti iscritti a quel corso. (Esempio *Fig. 2*) Se nessun corso è selezionato, avvisare l'utente con un messaggio di errore.

Ritorna una List<Corso> al quale lo studente è iscritto : public List<Corso> getCorsiFromStudente(Studente studente) {...}

## Punto 4

Implementare la <u>ricerca dei corsi a cui è iscritto uno studente</u>: data la matricola, facendo click sul pulsante *Cerca corsi*, controllare se lo studente è presente nel database, ed in caso visualizzare tutti i corsi ai quali è iscritto. (Esempio *Fig. 3*) Se la matricola non è presente, visualizzare un messaggio di errore.

Ritorna un boolean, devo solo capire se lo studente è iscritto al corso : public boolean isStudenteIscrittoAlCorso(Studente studente, Corso corso);

#### Punto 5

Implementare la funzionalità di <u>ricerca se uno studente è iscritto ad un corso</u>: selezionato un corso ed inserita la matricola di uno studente, facendo click su *Cerca*, è possibile sapere se lo studente specificato è iscritto a quel corso. (Esempio Fig. 4)

# Continua nella pagina seguente!

Controllare di aver gestito tutte le possibili condizioni di errore. (Es. studente non presente, studente non iscritto ad alcun corso, studente già iscritto ad un corso, corso senza iscritti, caratteri non permessi nel campo matricola).

Punto 6 (OPZIONALE) Ritornerà un boolean perché voglio solo sapere se è andato a buon fine;

Implementare la funzione <u>di iscrizione di uno studente ad un corso</u>: selezionato un corso ed inserita la matricola di uno studente, facendo click sul pulsante *Iscrivi* è possibile iscrivere lo studente a quel corso. (Esempio *Fig.*5)

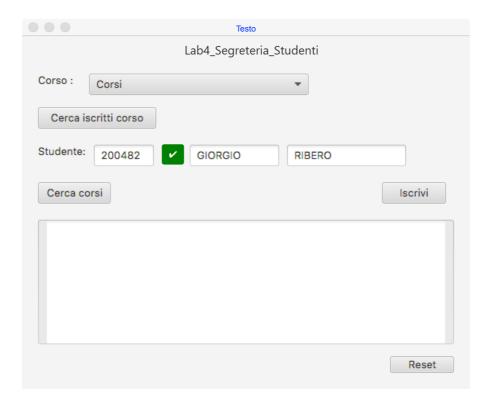


Fig. 1

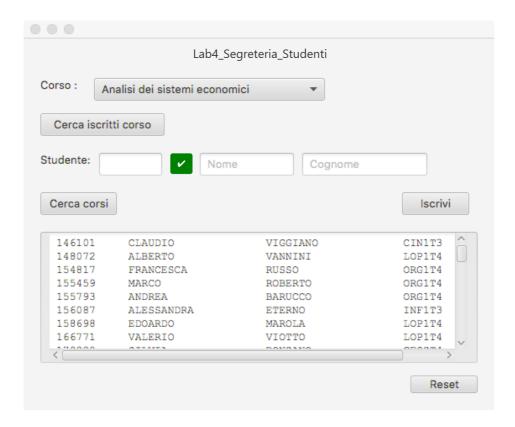


Fig. 2

		Lab4_Segreteria_Studenti	
Corso :	Corsi	•	
Cerca is	critti co	rso	
Studente:	2004	82 Nome Cognome	
Cerca co	rsi		scrivi
010VYP	G 8	Strategia, tecnologia e marketing	2
010VYP 01PDYP		Strategia, tecnologia e marketing Analisi e gestione dei sistemi produttivi	2 2
	G 8		_
01PDYP 02ANYP	G 8	Analisi e gestione dei sistemi produttivi Diritto commerciale	2
01PDYP 02ANYP 02AQJP	G 8	Analisi e gestione dei sistemi produttivi Diritto commerciale Economia dei sistemi industriali	2
01PDYP 02ANYP 02AQJP	G 8 G 8 G 8 G 8	Analisi e gestione dei sistemi produttivi Diritto commerciale Economia dei sistemi industriali	2 2 2
01PDYP 02ANYP 02AQJP 02PBVP	G 8 G 8 G 8 G 8	Analisi e gestione dei sistemi produttivi Diritto commerciale Economia dei sistemi industriali Analisi dei sistemi economici	2 2 2 1

Fig. 3

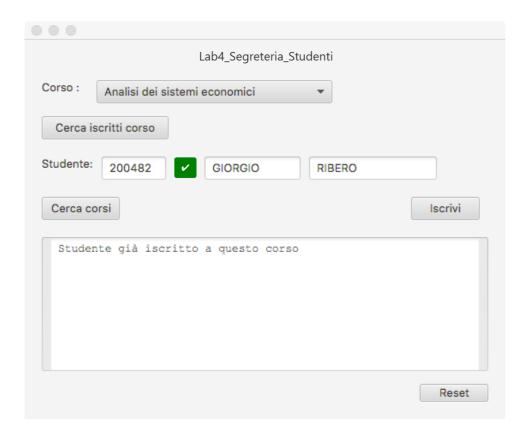


Fig. 4

	Lab4_Segreteria_Studenti					
Corso :	Gestione dei progetti ▼					
Cerca is	scritti corso					
Studente:	200482 GIORGIO RIBERO					
Cerca co	prsi	Iscrivi				
Studer	nte iscritto al corso!					
		Reset				

Fig. 5