



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

---

Dipartimento di ingegneria elettrica e delle tecnologie  
dell'informazione  
U2355 - Object Orientation

## Progettazione e sviluppo di un programma per la gestione di corsi di formazione con esempi d'uso

**Prepared by:** Ariola Luigi N86003483 &  
Biondi Morgan N86003419

**Instructor:** Prof. Sergio di Martino

**Date:** November 30, 2025

# Contents

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Descrizione del problema . . . . .	1
<b>2</b>	<b>Diagramma delle classi</b>	<b>1</b>
2.1	Introduzione . . . . .	1
2.2	Class Diagram . . . . .	2
2.3	Associazioni Controller e classi DAO . . . . .	2
2.4	Associazioni Controller e GUI . . . . .	3
2.5	Classi CourseTableModel & StudenteTableModel . . . . .	4
2.6	Classi TestCellRenderer,PanelButtonMouseAdapter e TableRowFilter . . . . .	5
2.7	Overview . . . . .	7
<b>3</b>	<b>CRC Card</b>	<b>8</b>
3.1	Introduzione . . . . .	8
3.2	CRC Card . . . . .	8
<b>4</b>	<b>Sequence Diagram di due funzionalità</b>	<b>18</b>
4.1	Introduzione . . . . .	18
4.2	Sequence Diagram . . . . .	18
<b>5</b>	<b>Esempi d'uso</b>	<b>20</b>
5.1	Inserimento Studente . . . . .	20

## List of Figures

2.1	Controller e classi DAO . . . . .	2
2.2	Controller e classi DAO . . . . .	2
2.3	Controller e Gui . . . . .	4
2.4	CourseTableModel . . . . .	4
2.5	StudenteTableModel . . . . .	5
2.6	TestCellRenderer . . . . .	5
2.7	PanelButtonMouseAdapter . . . . .	6
2.8	TableRowFilter . . . . .	6
2.9	ClassDiagram . . . . .	7
4.1	Sequence metodo inserisciCorso() . . . . .	18
4.2	Sequence metodo leggiStudenti() . . . . .	19

# CRC Card

1	PanelAggiungiStudente . . . . .	8
2	PanelGestisciStudente . . . . .	8
3	PanelDettagliStudente . . . . .	9
4	PanelAggiungiCorso . . . . .	9
5	PanelGestisciCorso . . . . .	9
6	PanelDettagliCorso . . . . .	10
7	Attendance . . . . .	10
8	MainFrame . . . . .	11
9	MainPanel . . . . .	11
10	LoginFrame . . . . .	11
11	GestisciRegistrazione . . . . .	12
12	GestisciLezioneJDialog . . . . .	12
13	StudentIdoneiJDialog . . . . .	12
14	CourseTableModel . . . . .	12
15	StudentTableModel . . . . .	12
16	TableRowFilter . . . . .	13
17	TestCellRenderer . . . . .	13
18	PanelButtonMouseAdapter . . . . .	13
19	Controller . . . . .	13
20	Admin . . . . .	14
21	AreeTematiche . . . . .	14
22	Corso . . . . .	14
23	Lezione . . . . .	14
24	Studente . . . . .	14
25	AdminDAOImpl . . . . .	15
26	CorsoDAOImpl . . . . .	15
27	AreaTematicaDAOImpl . . . . .	15
28	LezioneDAOImpl . . . . .	15
29	StudenteDAOImpl . . . . .	16
30	AdminDAO . . . . .	16
31	LezioneDAO . . . . .	16
32	CorsoDAO . . . . .	16
33	AareaTematicaDAO . . . . .	16
34	StudenteDAO . . . . .	17
35	Connessione . . . . .	17
36	DBBuilder . . . . .	17

# 1 Introduzione

## 1.1 Descrizione del problema

Si vuole implementare un sistema in Java che collabora con una base di dati relazionale. Il sistema permette ad un admin di creare uno o più corsi di formazione. Ciascun corso è caratterizzato da un nome, una descrizione, un massimo di partecipanti, un numero di presenze obbligatorie, una data di inizio e da una o più lezioni. A sua volta ogni lezione presenterà un titolo, una descrizione, una data ed ora inizio. Ogni corso potrà essere definito da una o più aree tematiche, che sono definite dagli admin del sistema. Ciascun operatore potrà iscrivere uno o più studenti ai corsi, e impostare le assenze/presenze per ogni lezione. È possibile, inoltre, controllare l'andamento di un corso attraverso una serie di statistiche, tra le quali il numero medio, massimo e minimo di studenti a lezione. Queste saranno disponibili in una tabella che potrà essere filtrata per data, o parole chiave relative al corso. Inoltre è possibile verificare gli studenti che hanno raggiunto il numero di presenze obbligatorie e sono quindi idonei al superamento del corso, attraverso una tabella che elencherà il nome, il cognome ed il numero di presenze degli studenti stessi.

# 2 Diagramma delle classi

## 2.1 Introduzione

In questa parte si inizia la progettazione del sistema ad un livello di astrazione più alto. Dopo aver analizzato il problema, si arriverà alla costruzione di un Class Diagram delle classi che descrive la struttura del sistema, mostrando le classi, i loro attributi, i metodi e le relazioni tra gli oggetti. Il pattern che si è utilizzato per la progettazione è il **DAO (Data-Access-Object)**, in modo da poter isolare il livello dell'applicazione dal livello di persistenza del database relazionale. In particolare, la complessità delle esecuzioni di operazioni **CRUD** è nascosta al resto, consentendo agli altri livelli di evolversi senza sapere nulla l'uno dell'altro. Ogni entità avrà una classe DAO, ovvero un'API astratta che esegue le operazioni CRUD. Le DAO avranno un'implementazione specifica nelle classi **DAOImpl**. Mentre le entità e le DAO coesistono indipendentemente l'una dall'altra, per fare in modo che queste ultime mantengano il livello di persistenza nascosto dalla logica dell'applicazione, si creerà solo un'istanza di DAO per poter eseguire le operazioni sulle entità. La classe che in generale si occupa di mediare tra DAO e l'applicazione è il **Controller**, che avrà istanze di tutte le classi astratte in modo da poter richiamare i propri metodi, e poterli utilizzare all'interno del programma.

## 2.2 Class Diagram

### 2.3 Associazioni Controller e classi DAO

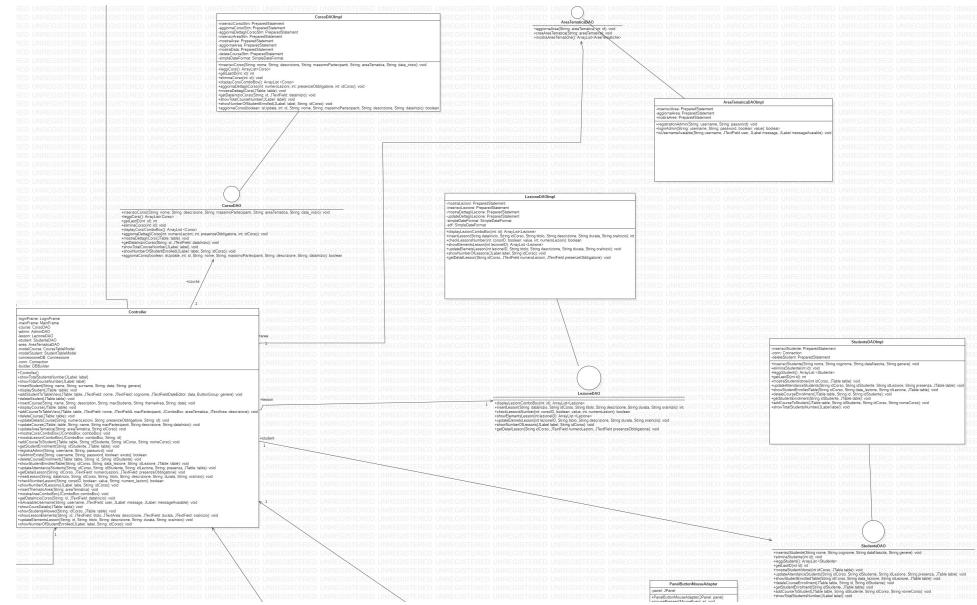


Figure 2.1: Controller e classi DAO

Nella prima fase di costruzione del Class Diagram, sono evidenziate le associazioni tra il Controller, la classe principale che si occupa dell'avvio del sistema, e le classi DAO.

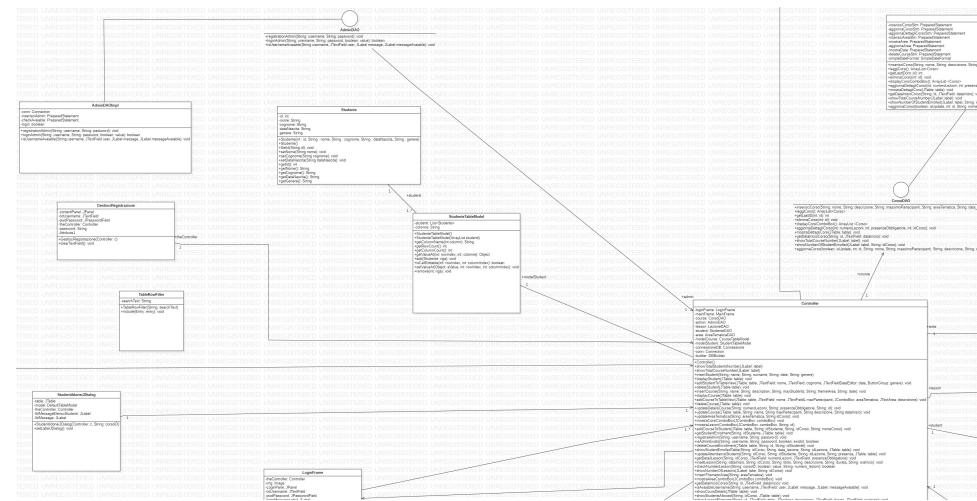


Figure 2.2: Controller e classi DAO

In particolare, il Controller è associato a CorsoDAO, in quanto fa uso dei suoi metodi, ed anche StudenteDAO, LezioneDAO, AreaTematicaDAO ed AdminDAO. I metodi utilizzati sono implementati nelle relative classi DAOImpl. Il Controller ha degli attributi di ciascuna classe ed attraverso cui ne richiamerà successivamente metodi.

## 2.4 Associazioni Controller e GUI

In seguito, per mediare tra DAO e GUI, il controller è stato associato ad ogni interfaccia progettata per l'applicazione.

L'approccio seguito per creare le interfacce è stato quello di implementare una sorta di contenitore con il quale poter scegliere il JPanel da visualizzare a seconda delle esigenze. Ogni interfaccia è associata alla classe MainFrame, che estende JFrame e rappresenta il Frame principale in cui saranno contenuti tutti i JPanel presenti nel programma. Il suo contenPane è suddiviso in due pannelli, che rappresentano due zone diverse dell'interfaccia. Sul lato sinistro c'è il **side\_panel**. All'interno sono stati inseriti vari pannelli, che costituiscono un vero e proprio menù con il quale interagire per cambiare finestra attraverso l'evento *mouseClicked()*. La parte centrale è occupata dal **mainContent\_panel**. Ad esso sono stati inclusi tutti i pannelli presenti. Al fine di cambiare JPanel da visualizzare nel MainFrame, è stato creato un metodo *menuSelected(JPanel selectedPanel)* che ha come parametro il Panel da visualizzare. A seconda della selezione, sarà eseguito *setVisible(false)* per tutti gli altri. In questo modo sarà possibile accedere ai vari pannelli, selezionandoli semplicemente dal menù laterale. Il menù infatti è costituito da due pannelli chiamati **studenti\_panel** e **corsi\_panel**. Selezionandone uno si aprirà il **course\_panel** o **student\_panel**, i quali nascondono altri JPanel che una volta cliccati renderanno visibile la parte centrale del **mainContent\_panel**. Per implementare questa sorta di menù a tendina, sono stati creati due metodi *nascondeSubMenu()* e *mostraSubMenu()* che gestiscono la visibilità ed i Bounds dei pannelli, in modo da poterli visualizzare al momento opportuno. Infine il side\_panel termina con il **panelAttendance**, che permette di visualizzare ed impostare le presenze per ogni lezione di un corso. Ogni volta che verrà selezionato un pannello da visualizzare, saranno inoltre richiamati dei metodi presenti in ogni JPanel (*showElements<NomePanel>*), dove all'interno si richiamano le procedure per poter caricare tutte le informazioni necessarie dal database. Inizialmente l'approccio è stato di caricare tutte le informazioni dal database all'avvio del MainFrame su tutti i panel. Con questo metodo però si riscontravano problemi di sincronizzazione con il DB, soprattutto riguardo le comboBox dei corsi, che non si sarebbero aggiornate nel caso in cui, ad esempio, ci fosse stato un inserimento o eliminazione di un corso. Per risolvere, sarebbe stato necessario implementare altre procedure di aggiornamento che avrebbero complicato il funzionamento dell'applicazione. Per questo, attraverso i metodi *showElements()* i dati saranno recuperati dal database singolarmente per ogni pannello, quando sarà selezionato. In aggiunta, sono richiamati da ogni pannello i metodi *clearFields()* (o *TextField*), che hanno il compito di ripulire tutte le componenti del JPanel visualizzato in precedenza. Ogni volta che si riaccederà ad un pannello, non lo si ritroverà "sporco", bensì allo stato di partenza.

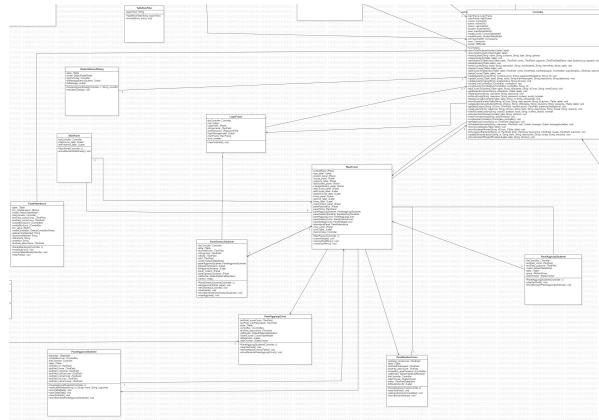


Figure 2.3: Controller e Gui

## 2.5 Classi CourseTableModel & StudenteTableModel

Le classi CourseTableModel e StudentTableModel sono rispettivamente associate a Corso,Studente ed alla classe Controller. Lo scopo di queste classi è definire gli elementi delle tabelle per i corsi e per gli studenti, come il numero di colonne, il tipo delle righe e delle colonne, ed i metodi ricavati facendo *override* dalla classe **AbstractTableModel**.

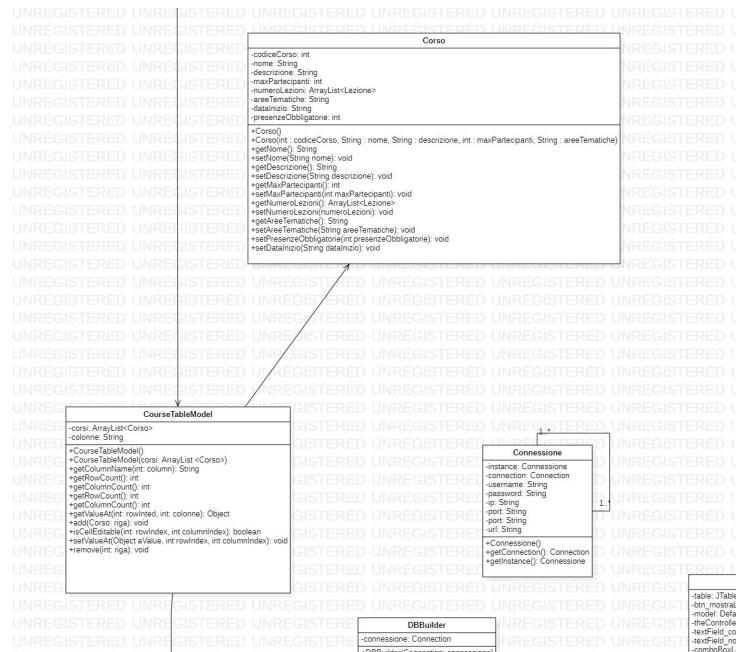


Figure 2.4: CourseTableModel

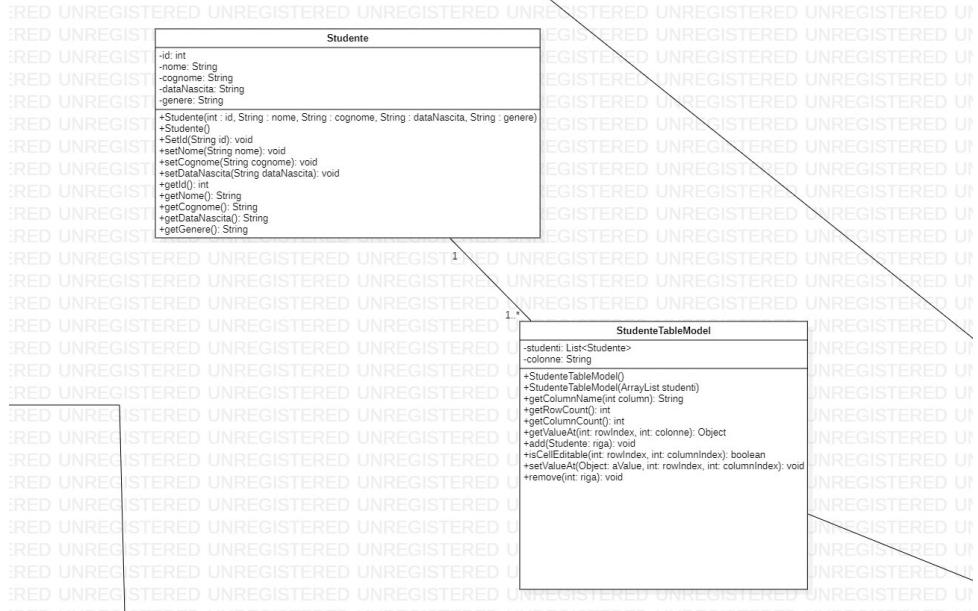


Figure 2.5: StudenteTableModel

## 2.6 Classi TestCellRenderer,PanelButtonMouseAdapter e TableRowFilter

Sono state create poi due classi per gestire alcune operazioni grafiche. La classe **TestCellRenderer** estende DefaultTableCellRenderer, per avere un comportamento specifico per ogni cella della JTable. JTable definisce per ogni cella della tabella un singolo renderer, e lo utilizza come un timbro per il rendering di tutte le celle della tabella. Attraverso il metodo `getTableCellRendererComponent` verrà restituito il renderer di celle di tabella predefinito. In questo modo in ogni cella che verrà attivata si scriverà il valore da assegnare, così da ottenere delle toolTipText per ogni elemento della tabella.

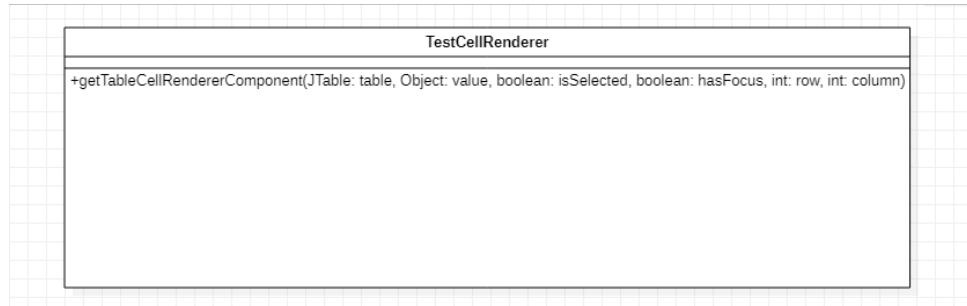


Figure 2.6: TestCellRenderer

La classe **PanelButtonMouseAdapter** estende MouseAdapter, che fornisce la ricezione di eventi del mouse. Gli eventi di cui è stato fatto l'override sono `mousePressed`, `mouseReleased`, `mouseEntered`, `mouseExited`. Ogni volta che viene attivato uno di questi, i JPanel che formano il menù cambieranno il colore in background a seconda dell'evento.

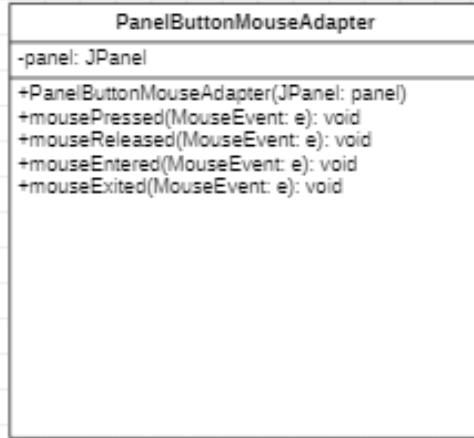


Figure 2.7: PanelButtonMouseListener

La classe **TableRowFilter** estende RowFilter, che è utilizzata per filtrare le entry dal model, in modo che non vengano visualizzate nella vista. Il metodo di cui è stato fatto override è *include()*, che restituisce **true** se la entry deve essere visualizzata, **false** altrimenti. In TableRowFilter c'è un costruttore che ha come parametro una stringa (**searchText**). Ogni volta che viene inserita una stringa per filtrare la JTable, il contenuto di ogni entry sarà confrontato con la stringa stessa, senza tenere in considerazione di eventuali differenze tra minuscole e maiuscole (ciò è ottenuto grazie alla combinazione dei metodi *toLowerCase()* ed *indexOf()*). Se vengono trovate delle similitudini, allora ci sarà il return del valore corrispondente che può essere visualizzato, altrimenti sarà ritornato false, ed il valore sarà nascosto.

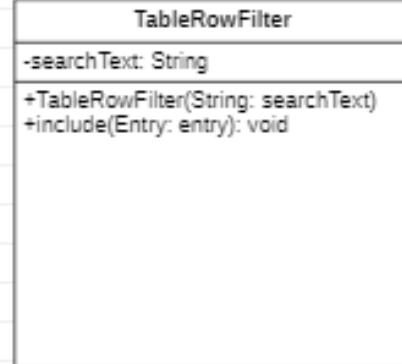


Figure 2.8: TableRowFilter

## 2.7 Overview

Questo è il Class Diagram completo con tutte le classi create nel sistema.

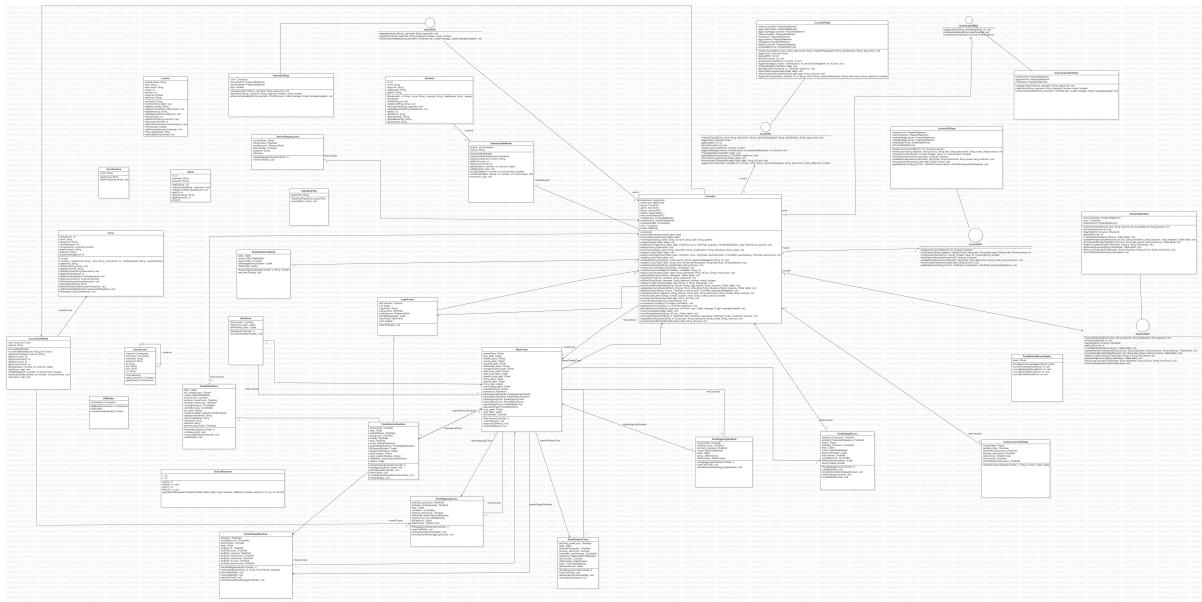


Figure 2.9: ClassDiagram

## 3 CRC Card

### 3.1 Introduzione

Dopo aver presentato il Class Diagram, un altro strumento che nelle fasi iniziali del progetto è risultato utile per la realizzazione della struttura di ogni classe sono le CRC Card. Una carta CRC rappresenta una sorta di scheda di una classe, che la descrive sommariamente, indicando

1. Nome della classe
2. Responsabilità
3. Collaboratori

Una responsabilità è tutto ciò che conosce o fa una classe. Spesso però una classe non ha abbastanza informazioni per adempiere alle sue responsabilità, ad esempio uno studente per poter essere iscritto ad un corso deve conoscere se esiste un posto a disposizione. Per farlo, deve collaborare con la classe Corso, che per questo motivo verrà inserita tra i collaboratori. Lo scopo è evitare di inserire troppe informazioni dettagliate, ed impedire che ad una classe vengano assegnate troppe responsabilità.

### 3.2 CRC Card

Di seguito sono elencate le CRC Card utilizzate per progettare il sistema

Table 1: PanelAggiungiStudente

PanelAggiungiStudente	
Responsability	Collaborator
Registrazione studenti DB	Controller
Eliminazione studenti DB	MainFrame StudenteDAOImpl StudentTableModel

Table 2: PanelGestisciStudente

PanelGestisciStudente	
Responsability	Collaborator
Accesso dettagli studente	MainFrame Controller StudenteDAOImpl StudentTableModel

Table 3: PanelDettagliStudente

PanelDettagliStudente	
Responsability	Collaborator
Iscrizione studente ai corsi	MainFrame
Eliminazione studente dai corsi	CorsoDAOImpl PanelGestisciStudente

Table 4: PanelAggiungiCorso

PanelAggiungiCorso	
Responsability	Collaborator
Aggiungi corso DB	MainFrame
Aggiungi area tematica	Controller CourseTableModel CorsoDAOImpl Corso AreaTematica

Table 5: PanelGestisciCorso

PanelGestisciCorso	
Responsability	Collaborator
Modifica corso	MainPanel
Elimina corso	Controller
Aggiungi area tematica	CourseTableModel
Modifica area tematica	CorsoDAOImpl Corso AreaTematica

Table 6: PanelDettagliCorso

PanelDettagliCorso					
Responsability	Collaborator				
Creazione lezioni	MainPanel				
Ricerca corsi per parole chiave	Controller				
Visualizzazione di statistiche	CorsoDAOImpl				
Visualizzazione studenti idonei	Corso	StudentiDAOImpl	Studenti	Lezioni	TableRowFilter

Table 7: Attendance

Attendance						
Responsability	Collaborator					
inserimento presenze	MainPanel					
visualizzazione studenti presenti	Controller	CorsoDAOImpl	Corso	Studenti	LezioneDAOImpl	Lezione

Table 8: MainFrame

MainFrame	
Responsability	Collaborator
gestione menu laterale	Controller
gestione visibilità panel	MainPanel
	PanelAggiungiStudente
	PanelGestisciStudenti
	PanelAggiungiCorso
	PanelDettagliCorso
	PanelGestisciCorso
	PanelAttendance

Table 9: MainPanel

MainPanel	
Responsability	Collaborator
mostra numero corsi creati	MainFrame
mostra numero studenti registrati	Controller

Table 10: LoginFrame

LoginFrame	
Responsability	Collaborator
Accesso al MainFrame	Controller
Registrazione admin	Admin
	AdminDAOImpl

Table 11: GestisciRegistrazione

GestisciRegistrazione	
Responsability	Collaborator
Registrazione admin	Controller LoginFrame Admin AdminDAOImpl

Table 12: GestisciLezioneJDialog

GestisciLezioneJDialog	
Responsability	Collaborator
Seleziona corso	Controller
Creazione lezioni	LezioneDAOImpl

Table 13: StudentiIdoneiJDialog

StudentiIdoneiJDialog	
Responsability	Collaborator
Mostra studenti idonei per corso selezionato	Controller Studente

Table 14: CourseTableModel

CourseTableModel	
Responsability	Collaborator
Costruzione tabella corsi	AbstractTableModel

Table 15: StudentTableModel

StudentTableModel	
Responsability	Collaborator
Costruzione tabella studenti	AbstractTableModel

Table 16: TableRowFilter

TableRowFilter	
<b>Responsability</b>	<b>Collaborator</b>
Filtraggio tabella in base a parole chiave	RowFilter

Table 17: TestCellRenderer

TestCellRenderer	
<b>Responsability</b>	<b>Collaborator</b>
Imposta Tooltip per JTable	DefaultTableCellRenderer

Table 18: PanelButtonMouseAdapter

PanelButtonMouseAdapter	
<b>Responsability</b>	<b>Collaborator</b>
Gestione dell'evento Mouse-Adapter	MouseAdapter

Table 19: Controller

Controller	
<b>Responsability</b>	<b>Collaborator</b>
Connessione DB	Connessione
Launcher finestre	CorsoDAO
Gestisce la corrispondenza tra GUI e DAO	StudenteDAO
	LezioneDAO
	AdminDAO
	LoginFrame
	MainFrame
	CourseTabelModel
	CourseTAbleModel

Table 20: Admin

Admin	
Responsability	Collaborator
Username	
Password	

Table 21: AreeTematiche

AreeTematiche	
Responsability	Collaborator
set e get nome aree tematiche	

Table 22: Corso

Corso	
Responsability	Collaborator
set e get attributi corso	

Table 23: Lezione

Lezione	
Responsability	Collaborator
set e get attributi lezione	

Table 24: Studente

Studente	
Responsability	Collaborator
set e get attributi studente	

Table 25: AdminDAOImpl

AdminDAOImpl	
Responsability	Collaborator
Registrazione admin DB	Connessione
Controllo disponibilità username	AdminDAO
Login	

Table 26: CorsoDAOImpl

CorsoDAOImpl	
Responsability	Collaborator
Inserire corso DB	Connessione
Mostra corsi in tabella	CorsoDAO
Mostra corso ComboBox	
Elimina corso DB	
Aggiorna corso	

Table 27: AreaTematicaDAOImpl

AreaTematicaDAOImpl	
Responsability	Collaborator
Aggiorna area tematica	Connessione
Crea area tematica	AreaTematicaDAO
Mostra area tematica	

Table 28: LezioneDAOImpl

LezioneDAOImpl	
Responsability	Collaborator
Mostra lezione ComboBox	Connessione
Inserisci lezione DB	Connessione
Controllo numero lezioni	
Mostra dettagli lezione	
Aggiorna dettagli lezione	

Table 29: StudenteDAOImpl

StudenteDAOImpl	
Responsability	Collaborator
Inserimento studente DB	Connessione
Eliminazione studente DB	StudenteDAO
Mostra studenti JTable	
Mostra studenti idonei	

Table 30: AdminDAO

AdminDAO	
Responsability	Collaborator
interfaccia dei metodi presenti in DAOImpl	

Table 31: LezioneDAO

LezioneDAO	
Responsability	Collaborator
interfaccia dei metodi presenti in DAOImpl	

Table 32: CorsoDAO

CorsoDAO	
Responsability	Collaborator
interfaccia dei metodi presenti in DAOImpl	

Table 33: AareaTematicaDAO

AareaTematicaDAO	
Responsability	Collaborator
interfaccia dei metodi presenti in DAOImpl	

Table 34: StudenteDAO

StudenteDAO	
<b>Responsability</b>	<b>Collaborator</b>
interfaccia dei metodi presenti in DAOImpl	

Table 35: Connessione

Connessione	
<b>Responsability</b>	<b>Collaborator</b>
Stabilisce connessione con il DB	Connessione

Table 36: DBBuilder

DBBuilder	
<b>Responsability</b>	<b>Collaborator</b>
Costruzione del database	Connessione

## 4 Sequence Diagram di due funzionalità

### 4.1 Introduzione

In questa sezione si vogliono presentare due Sequence Diagram che riguardano due metodi del sistema. Tramite i sequence si riesce a spiegare in dettaglio come vengono eseguite le operazioni. A differenza dei Class Diagram sono dinamici, perché si focalizzano sul tempo, seguendo l'ordine delle operazioni di un metodo.

### 4.2 Sequence Diagram

Il primo è *insertCourse(nome, descrizione, maxPartecipanti, areaTematica, dataSQL)*. Il suo scopo è quello di inserire un corso nel database, e si attiva quando l'utente clicca il pulsante **inserisci** in PanelAggiungiCorso.

Inizialmente, si stabilisce la connessione richiamando il metodo *getInstance* e *getConnection* con il database. Successivamente ci sono le varie operazioni di parse, che servono per inserire i dati correttamente nel DB. Successivamente vengono settati i valori convertiti alle rispettive colonne all'interno della tabella del database e si esegue la query.

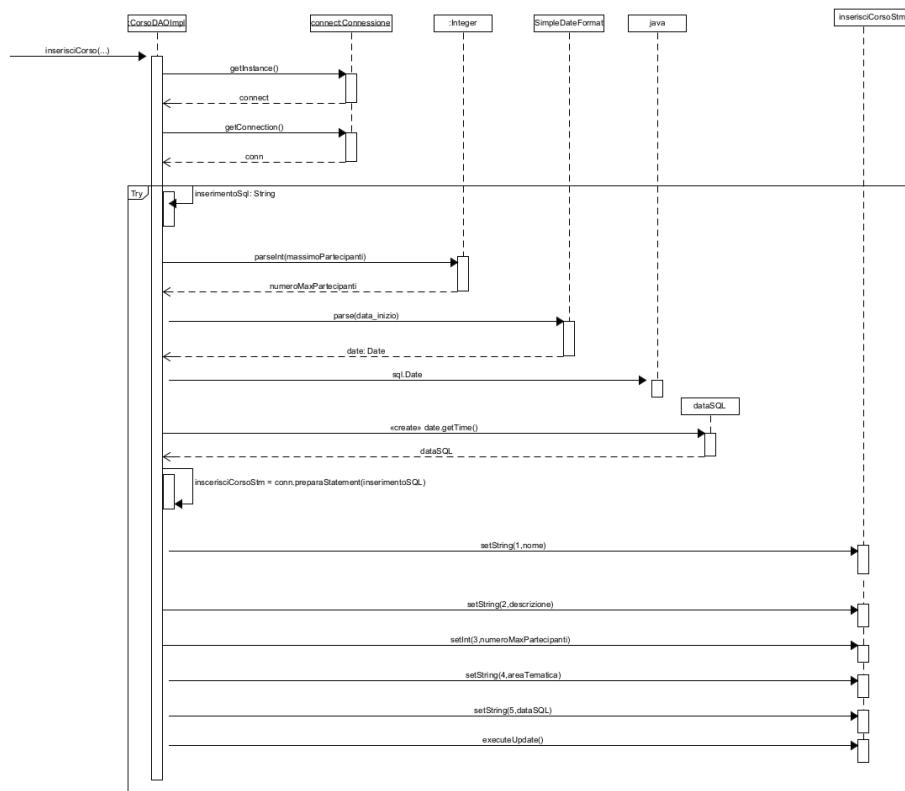


Figure 4.1: Sequence metodo inserisciCorso()

Il secondo Sequence, invece, fa riferimento al metodo *leggiStudenti()* presente in StudenteDAOImpl. A differenza di inserisciCorso, in questo caso si vuole fare il retrieving di tutti gli studenti presenti nel database. Quindi, dopo aver stabilito la connessione, viene creato un ArrayList di studenti al quale successivamente verranno aggiunti gli studenti

recuperati. All'interno del while si scorrono tutte le righe della tabella studente nel DB, e finché esistono studenti, si settano id, nome, cognome, dataNascita, genere per ogni studente e lo si adda alla lista da ritornare. Infine ci sarà il return dell'intera lista che poi potrà essere usata per riempire le JTable degli studenti.

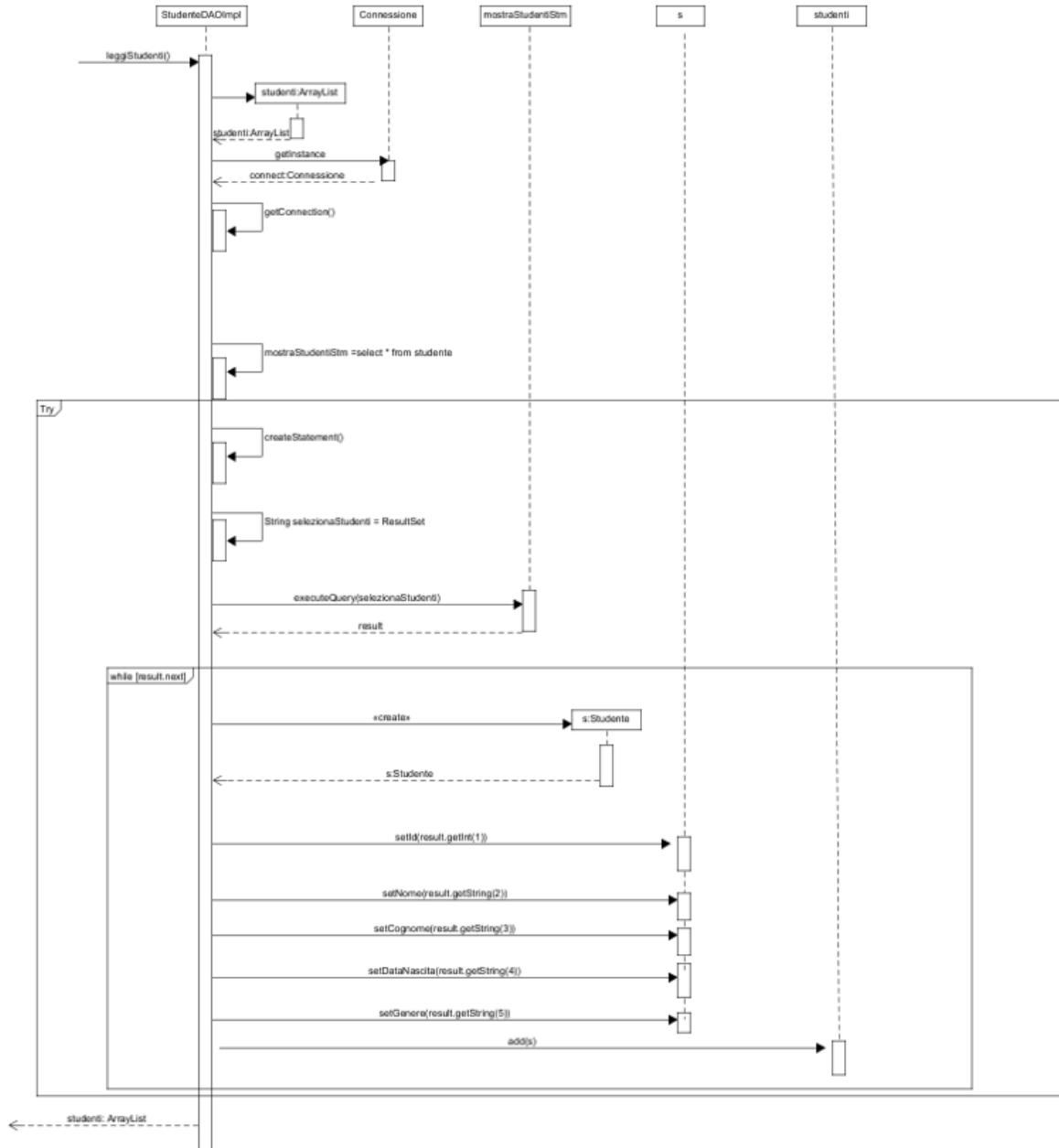
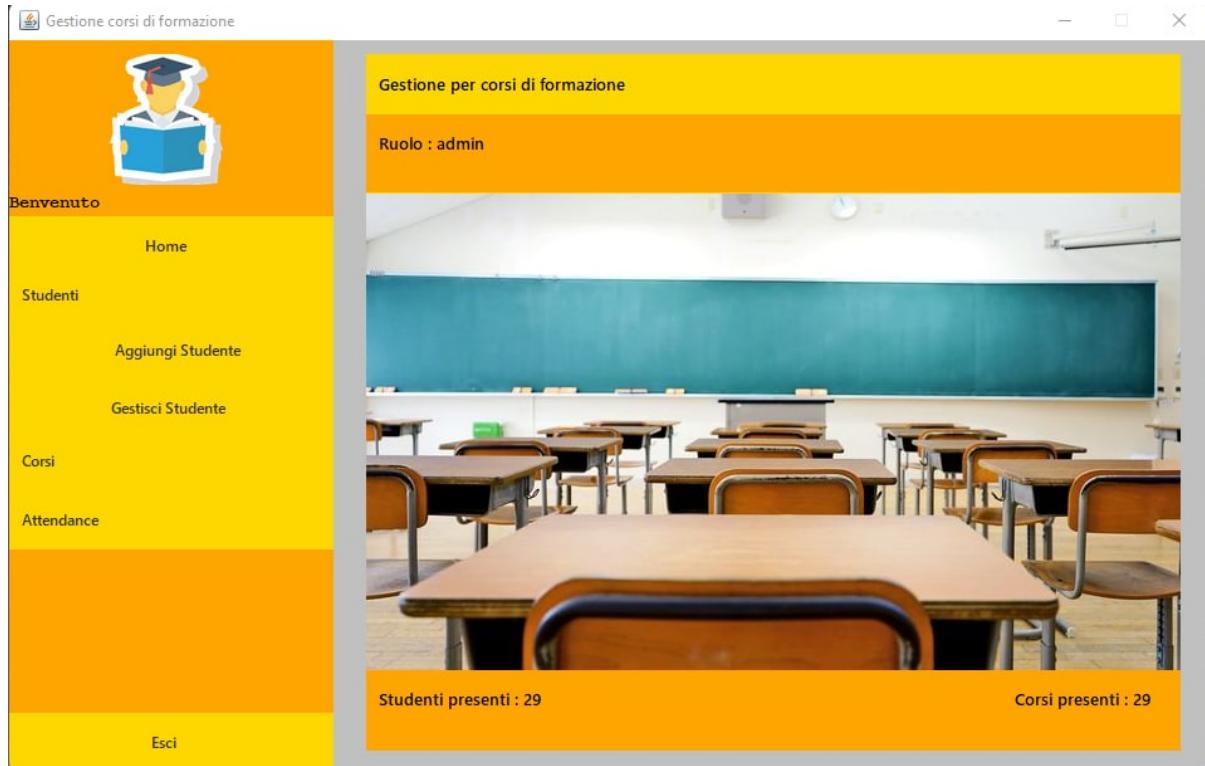


Figure 4.2: Sequence metodo leggiStudenti()

## 5 Esempi d'uso

### 5.1 Inserimento Studente

Dopo aver effettuato il login con il DataBase, selezionare **Studenti** dal side menù, e cliccare su **Aggiungi Studente**.



A questo punto inserire i dati dello studente rispettando opportunamente i campi

Gestione corsi di formazione



Benvenuto

Home

Studenti

Corsi

Attendance

Nome : Data Nascita :

Nome : Data Nascita :

Cognome : Genere :

Genere :

Inserisci

Esci

ID	Nome	Cognome	Data Nascita	Genere
1	Luigi	Ariola	2000-10-10	Uomo
3	Sergio	Mattarella	1941-07-23	Uomo
4	Omella	Muti	1955-03-09	Donna
5	Franz	Kafka	1883-07-03	Uomo
7	Gianluca	Vacchi	1967-08-05	Uomo
8	Muhammad	Ali	1941-01-17	Uomo
9	Mike	Tyson	1966-06-30	Uomo
12	Tiziano	Ferro	1980-02-21	Uomo
13	Lorenzo	Cherubini	1966-09-27	Uomo
14	Luigi	Pirandello	1867-06-28	Uomo
17	Pino	Mauro	1990-05-10	Uomo
18	Mimmo	Blanco	2022-02-04	Uomo
23	Giuseppe	Verdi	1923-02-09	Uomo

Infine cliccare su **Inserisci**. L'inserimento sarà visibile all'interno della tabella presente nel pannello stesso.

The screenshot shows a software interface for managing training courses. On the left, there's a sidebar with icons for a user profile, Home, Students, Courses, Attendance, and Logout. The main area has a title 'Registrazione' and a table with columns: ID, Nome, Cognome, Data Nascita, and Genere. A modal dialog box titled 'Conferma' displays the message 'Inserimento effettuato' (Insertion performed) with an 'OK' button. Below the table, there are input fields for Nome (Diego), Cognome (Maradona), and Genere (Uomo selected). A blue 'Inserisci' button is located at the bottom right. The table data is as follows:

ID	Nome	Cognome	Data Nascita	Genere
1	Luigi	Ariola	2000-10-10	Uomo
3	Sergio	Mattarella	1941-07-23	Uomo
4	Ornella	Muti	1955-03-09	Donna
5	Franz	Kafka	1883-07-03	Uomo
7	Gianluca	Vacchi	1967-08-05	Uomo
8	Muhammad	Ali	1941-01-17	Uomo
9	Mike	Tyson	1966-06-30	Uomo
12	Tiziano	Ferro	1980-02-21	Uomo
13			1966-09-27	Uomo
14			1867-06-28	Uomo
17			1990-05-10	Uomo
18			2022-02-04	Uomo
23			1923-02-09	Uomo