RELAZIONE PROGRAMMAZIONE AD OGGETTI

Nome e Matricola:

PEGUY NYA SADJO , 1126295

February 16, 2019

Titolo: University Faculties

Contents

1 Introduzione

1.1 Abstract

L applicazione (University Faculties) sviluppata da me è centrata su un Dipartimento in particolare quello di INGENIERIA che rappresenta uno tra tutti i Dipartimenti che possono esistere in un Università.Nonostante che esistano numerosi Facoltà in quel dipartimento,ho scelto di sviluppare una piccola applicazione basata su tre facoltà(Informatica,Meccanica,Aeronautica) precise cercando di rispettare i requisiti del progetto.

1.2 Ore Impiegate Per il Progetto: 50

- **Template di Container:** 3 Ore(1 Ora di Progettazione 2 Ore di Codifica).
- **Modello:** 7 Ore(4 Ore di Progettazione e 3 Ore di Codifica
- **Imparare Qt:** 20 Ore(20 Ore di impredimento di Qt che era un linguaggio nuovo per me).
- **Controller:** 7 Ore(4 Ore di Progettazione e 3 Ore di Codifica).
- **View** 10 Ore(7 Ore di Progettazione e 3 Ore di Codifica).
- **Debug:** 3 Ore(incluse ore per memory leak analyser).

2 Presentazione del progetto

2.1 Organizzazione dell'informazione

Questo progetto o applicazione sviluppata da me ha lo scopo di aiutare il settore di segretaria studente di un Università a poter Inserire o Registrare ,Rimuovere,Cercare uno studente appartenente ad un dipartimento preciso (Ingenieria nel mio caso) rispettando i requisiti d ingresso alla facoltà(nel caso dell inserimento). Inoltre,dall applicazione ho offerto la possibilita all utente che l utilizza di poter avere una lista degli studenti appartente ad ogni facoltà precisa.
Nella realizzazione dell applicazione, ho tenuto ad separare all massimo tutte le parte(presentazione,contenuto,comportamneto) per permettere una manutenzione facile e anche per la robustezza dell applicazione che si puo facilmente estendere.

L' applicazione si vede come presente nella Figura 1.

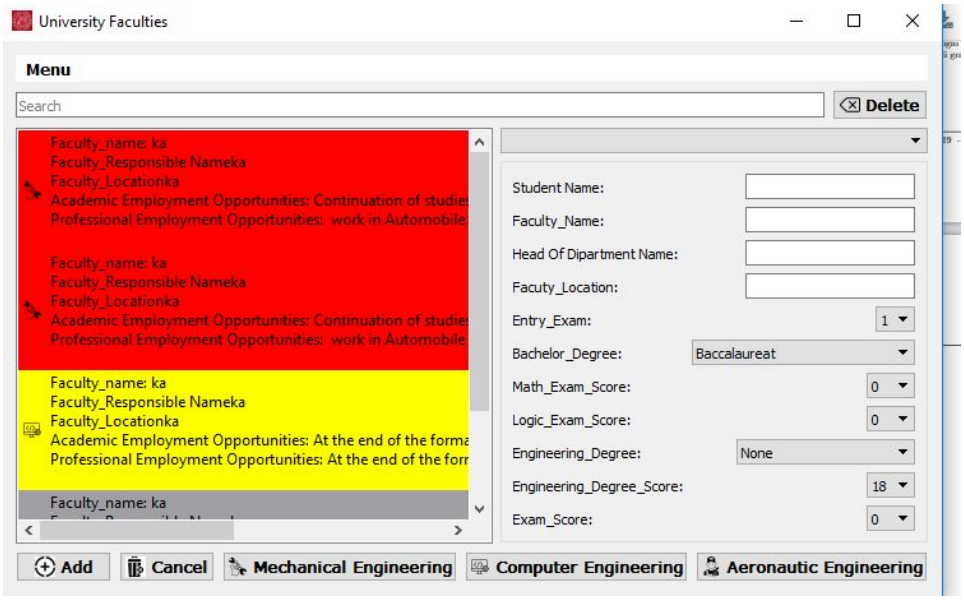

Figure 1: Presentazione dell'applicazione.

Come si puo vedere, l applicazione comporta piu bottoni,una vista e una dialog in cui si puo fornire informazioni.
Piu in alto dell applicazione ,c'è il titolo dell applicazione University Faculties,con un logo scelto apposto. nella barra di navigazione, ho incluso un bottone Menu cliccando su menu,si vede due altri bottoni(save,close).
- Save permette di salvare i dati su un file Xml.
- Close chiude l applicazione.
Il sito presenta ancora altri bottoni come:
- Add: permette di aggiungere un nuovo elemento nella vista.
- Cancel: permette di cancellare l'ultimo elemento della vista.
- Mechanical Engineering: che ci apre una nuova finestra e stampa la lista degli studenti di questa facoltà.
- Computer Engineering: chi stampa la lista degli studenti appartenete alla facolta.
- Aeronautic: chi stampa la lista degli studenti della facoltà di Aeronautic.
Oltre ai bottoni, abbiamo una vista che ci stampa le informazioni sulla facoltà in cui viene iscritto lo studente ogni volta che si clicca sulla Add.
Ho anche fatto uso dei colori e delle icone alla sinistra di ogni informazione per differenziare le diverse facoltà di interesse.
Il colore Rosso per rappresentare la facoltà d ingenieria Mecanica, Giallo per ingenieria informatica e Griggio per Aeronautica.

2.1.1 Database

Tutto il database è stato gestito dalla classe XMLIO.Questa classe è stata utilizzata per permettere scritture e letture sul e dal file XML rispettando le norme di scrittura e letura imposte dallo standard di XML come ci presenta La Figura 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--File di salvataggio dell'applicazione. Non modificare a mano.-->
- <root>
  - <Engineering type="Aeronautic">
      <s_name>ma</s_name>
      <fac_name>Faculty_name: ka</fac_name>
      <head_name>Faculty_Responsible
          Nameka</head_name>
      <location>Faculty_Locationka</location>
      <degree>Baccalaureat</degree>
      <entry_exam>1</entry_exam>
      <math_score>15</math_score>
      <logic_score>12</logic_score>
      <bachelor_degree>Physics</bachelor_degree>
      <degree_score>25</degree_score>
      <entry_exam_score>14</entry_exam_score>
    </Engineering>
  + <Engineering type="Mechanics">
  + <Engineering type="Informatic">
  + <Engineering type="Informatic">
</root>
```

Figure 2: File Xml dell applicazione.

3 Contenitore

Tra le classi presente nel modello, c' è un Template di classe. chiamato Container che è stato creato come richiesto dai requisiti del progetto per contenere una lista di facoltà(nel mio caso).dentro il template Container,ho sviluppato 2 altre classi iteratori una costante e l'altra no per scorrere la lista ed effettuare vari operazione sugli elementi della lista. Ho fatto la scelta di sviluppare una lista doppiamente linkata per motivi di efficienza sia nell inserzione(che ho scelto di fare sempre in testa alla lista) che la rimozione che,se si fa in testa o sull' ultimo elemento della lista si farà sempre in un tempo costante O(1).

4 Gerarchia delle classi del modello

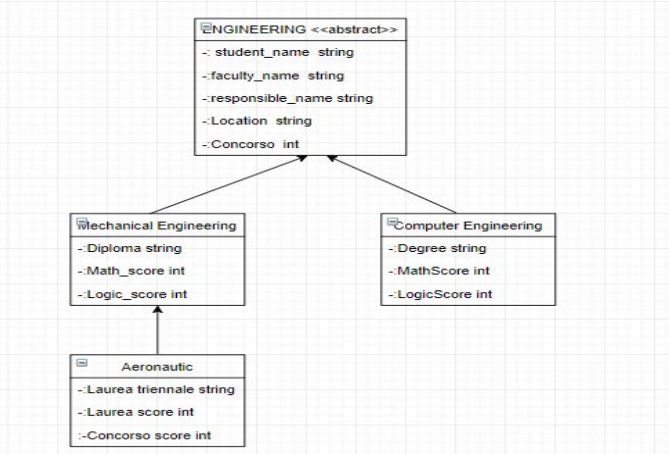

Figure 3: Gerarchia delle classi.

4.1 Engineering:

E' la classe base astratta ed è rapprensentata dai seguenti metodi:
- virtual string $Academic_Employment_Opportunities() const = 0$;
– $virtual\ string\ Professional_Employment_Opportunities() const = 0$;
– $virtual\ string\ Educational_Pathway() const$;
– $virtual\ string\ getResponsibleName() const$;
– $virtual\ string\ getLocation() const$;
– $virtual\ string\ getFacultyName() const$;
– $virtual\ string\ getStudentName() const$;
– $virtual\ int\ has_Concorso() const$;
– $virtual\ stringgetType() const$;

4.2 Mechanical Engineering

E' la sottoclasse diretta concreta di Engineering ed è rapprensentata dai seguenti metodi:
- virtual bool access()const override;
- virtual string $Academic_Employment_Opportunities() constoverride$;
– $virtual\ string\ Professional_Employment_Opportunities() constoverride$;
– $virtual\ string\ Educational_Pathway() const$;
– $string\ getDiploma() const$;
– $int\ getMathScore() const$;
– $int\ getLogicScore() const$;
– $virtual\ string\ getType() constoverride$;

4.3 Computer Engineering:

E' la sottoclasse diretta concreta di Engineering ed è rapprensentata dai seguenti metodi:
- virtual string Academic$_E$mployment$_O$pportunities()constoverride;
– *virtual string Professional$_E$mployment$_O$pportunities()constoverride*;
– *virtual string Educational$_P$athway()const*;
– *string getDiploma()const*;
– *int getMathScore()const*;
– *int getLogicScore()const*;
– *virtual string getType()constoverride*;

4.4 Aeronautic:

E' la sottoclasse diretta di Mechanical Engineering ed è rapprensentata dai seguenti metodi:
- virtual bool access()const override;
- virtual string Academic$_E$mployment$_O$pportunities()constoverride;
– *virtual string Professional$_E$mployment$_O$pportunities()constoverride*;
– *virtual string Educational$_P$athway()const*;
– *virtual string getFacultyName()const*;
– *virtual string getStudentName()const*;
– *string getDiploma()const*;
– *int getMathScore()const*;
– *int getLogicScore()const*;
– *virtual string getType()constoverride*;

5 Descrizione del codice polimorfo del modello

Nella classe base Engineering,sono stati definiti metodi sia puri che non puri(che sono stati implementati nel Engineering.cpp).Tutti questi metodi saranno implementati o ovveridati nelle classi derivate assicurando che le operazioni vengono fatte in base ai dati ricevuti in ingresso dall applicazione.Esempio ogni volta che viene chiamato il metodo Professional Employment Opportunities(),essendo ovveridato,la scelta dal metodo da eseguire si fà secondo il tipo dinamico del puntatore ad Engineering che lo invoca e viene ritonato il metodo adeguato che appartiene ad una delle classi derivate di Engineering.Insoma tutti i metodi virtuali usati nel codice permettono al controller di eseguire delle chiamate polimorfe in base ai dati forniti in ingresso per soddisfare determinate richieste.

6 Gerarchia delle classi del Controller

6.1 ModelAdapter

Per la parte Controller che si occupa della communicazione tra modello e view, ho scelto di essere il piu semplice posibile nella sua rappresentazione come si vede sulla figura 4.

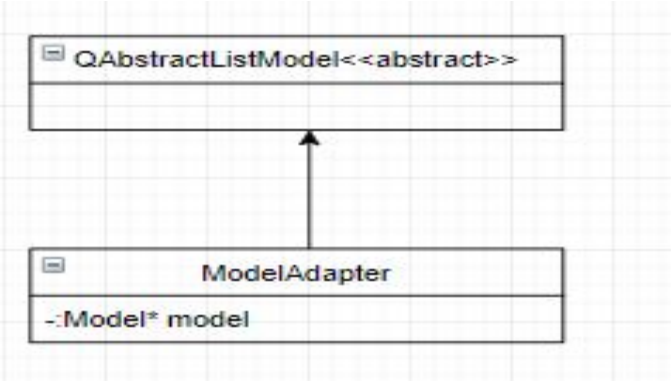

Figure 4: Risultato del test visivo sui colori

La classe ModelAdapter che si occupa del flusso di dati scambiati tra il modello e la view comporta un riferimento al modello nella sua parte privata.Come metodi, ho implementato qualche metodo che si comporta da adattatore tra i dati che il modello passa alla view ogni volta che si vuole effetuare operazioni o scambio di dati tra loro.Imetodi implementati dalla classe ModelAdapter sono i seguenti: - int rowCount(const QModelIndex = QModelIndex()) const override;
- Model* getModel()const; ritorna un riferimento al modello
- QVariant data(const QModelIndex, int role = Qt::DisplayRole) const override;
- void saveToFile()const;
- void loadFromFile();
- void $Read_Info()const$;
$-QStringmodifica()$;
$-voidinsertrow(intindice, conststring, conststring, conststring, conststring, int, conststring, int, i$
$-boolremoveRows(int, int = 1, constQModelIndex = QModelIndex())override$;

7 Gerarchia delle classi della View

7.1 Mainwindow

Essendo la finestra principale,rappresenta la classe base delle altre view presente sull apllicazione(View1,Dialog,Dialog2).La mainwindow comporta nei suoi campi dati privati un riferimento al controller(Figura 5) che permette alla Mainwindow di communicare o mandare segnali al controller che saranno successivamente mandati al modello,un riferimento alla View1 in cui vengono stampati i dati ricevuti dal modello.

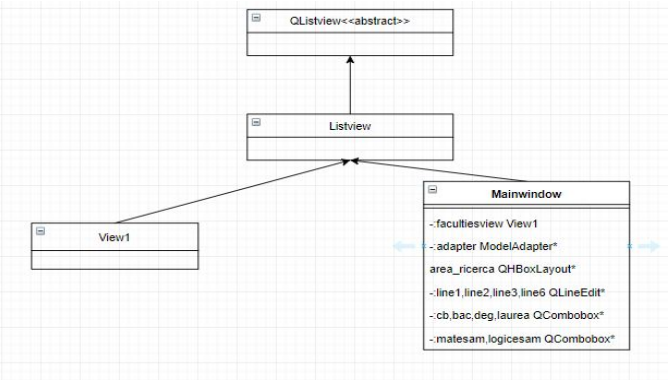

Figure 5: Gerarchia delle classi della View

In piu, della mainwindow e della View1, ho usato due Dialog per stampare informazioni specifiche come i nomi degli studenti di ogni facotà con Dialog e una descrizione della facoltà con Dialog2 tutti i due sono derivate della QDialog e si presentano anche come figli della Mainwindow. queste finestre sono invocate solo una volta che l utente abbia mandato un segnale cliccando sul buttone adeguato(Mechanical Engineering,Computer Engineering ecc..) come si presenta la Figura(6)

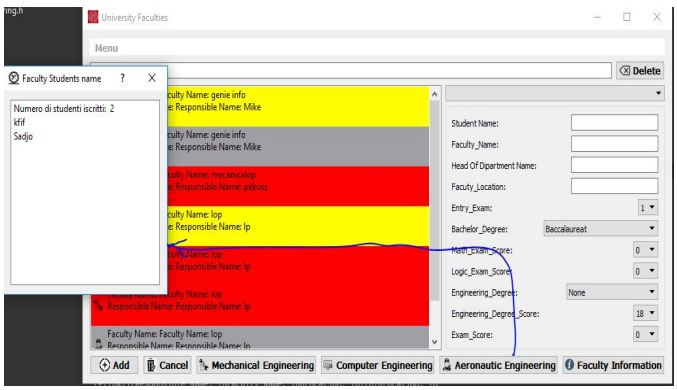

Figure 6: Gerarchia delle classi della View