Alberto Andeliero 1031800

Relazione progetto di Programmazione Ad Oggetti qDB

1.Introduzione

Lo scopo del progetto qDB è lo sviluppo in C++/Qt di un sistema minimale per la gestione di un (piccolo) database tramite una interfaccia utente grafica.
In questo progetto sono state implementate le funzionalità di inserimento, rimozione, ricerca, modifica, salvataggio su file XML e apertura da file XML.

2.Template di classe Container<K>

Nel file "deque.h" come template di classe contenitore è stata implementata una double-ended queue(deque<K>) per avere complessità O(1) per le operazioni sia in testa e sia in coda.
Per poter manipolare i nodi è stata implementata una classe annidata nodo dove verranno memorizzate i puntatori ai nodi precedenti, successivi e il campo dati K del template.
Sono state implementate le operazioni più importanti presenti anche nei contenitori della stl quali void push_back(const K&), void push_front(const K&), K pop_front(), K pop_back() , la compilazione è stata fatta per inclusione per ragioni di compatibilità.
Sono rese disponibili in public una classe base const_iterator che permette le più importanti operazioni sul contenitore senza modifiche ai campi dati, da questa classe deriva iterator che non aggiunge altri metodi ma permette la modifica. Questo iteratore itera dal nodo* inizio(incluso) a nodo* 0(non incluso) e quindi i metodi begin() e end() restituiscono tali iteratori come i contenitori della stl.

Per eliminare un elemento dal conteiner sono resi disponibili i metodi void erase(iterator), void deleteAt(int) oppure per deallocare ogni elemento void erase().

3.Classi Logiche

Nei file "guitar.h/cpp", "acoustic.h/cpp", "electric.h/cpp" è stata creata una piccola gerarchia di classi che rappresentano le chitarre presenti nel negozio. La classe base è guitar da cui derivano le classi acoustic ed electric, nei suoi campi dati troviamo una stringa pel nome del modello, un intero per le corde e il prezzo in double. I costruttori sono protetti perchè non vengano allocati oggetti di questa classe e siano utilizzati soltanto dalle classi derivate.
I medoti più importanti da elencare sono:

virtual bool operator==(const guitar&)const;
metodo per l'operatore di uguaglianza che verrà reimplementato anche nelle sottoclassi electric e acoustic.

virtual string get_ponte()const =0;
virtual void set_ponte(string) =0;
questi metodi verranno implementati nelle sottoclassi perchè i vari tipi di chitarra montano modelli di ponte completamente diversi.

virtual bool amplf() const =0;
metodo che ritorna true se la chitarra è amplificabile, false altrimenti, utile nel caso una chitarra acustica è provvista di impianto di amplificazione, una chitarra elettrica ritorna sempre true perchè è sempre amplificabile.

La classe electric nei suoi campi dati specifica il modello del ponte e anche il modello dei pickup montati per amplificare il segnale mentre nella classe acoustic viene indicato se è amplificabile con un booleano e il modello del ponte con una stringa.
Nel file "essemu.h/cpp" la classe responsabile della manipolazione del database è essemu, e viene indicato il nome del file associato in 'file' e se è ci sono state modifiche con un booleano 'changed'.

La gestione dell'apertura e il salvataggio in XML è implementata nei metodi di void openXml(string) e void saveXml(string) usando le classi QXmlStreamReader e QxmlStreamWriter del modulo <QtCore>, questi metodi li ho implementati in questa classe perchè li considero metodi logici.
I metodi per la gestione dei dati sono 'void insertGtr(const guitar&)', 'void removeGtr(int i)' e 'void replaceGtr(guitar*,guitar*)', quest'ultimo utile per campiare il tipo di un'oggetto.
La ricerca viene implementata nel metodo,
'deque<int> searchgtrs(int cl,int cu,double pl,double pu, bool amp,string mo="",string ponte="",string typ="",string pickup="")const' che restituisce una lista di indici del magazzino che soddisfa la ricerca, per la ricerca ho utilizzato un metodo messo a disposizione nella classe Qstring bool 'contains (const QString & str, Qt::CaseSensitivity cs = Qt::CaseSensitive) const'.

4.Interfaccia Grafica

La classe grafica che gestisce la finestra principale è mainwindow "mainwindow.h" che deriva da QmainWindow che mette a disposizione un menù a tre tendine (File,Tabella,Ricerca).
Le funzioni implementate in "File" sono quelle necessarie per gestire un file, mentre in "Tabella" abbiamo 2 funzioni che permettono di eliminare gli elementi selezionati nelle righe della tabella e di aggiungere una nuova chitarra invocando la classe grafica elementdialog "elementdialog.h" , questa classe deriva Qdialog e utilizza come form newelement "newelement.h"
In "Ricerca" abbiamo la funzione di ricerca e per chiuderla, invocando la classe searchdialog nel "searchdialog.h".
In mainwindow viene installata come Widget centrale table che deriva da QtableWidget, questa tabella è responsabile della rimozione, inserimento e cambio oggetto nei record memorizzati in essemu *shop, e in ogni riga associata ad un element ("element.h") vengono installate nelle celle i Widget dei suoi campi dati. Ogni element è associato ad un guitar* ed è responsabile della manipolazione dei suoi campi dati, nel caso debba campiare tipo emette un SIGNAL "void changeElement(int index);" dove index riferisce la posizione in essemu* shop e verrà incocata il metodo "void replaceGtr (guitar* , guitar*); " dalla classe table.