Alex Ruzzante Matricola: 1030089

qDB

Progetto di Programmazione ad Oggetti a.a.2013/14

	Materiale	Qualita'	Prezzo	Tipo	Misura	Solitario	Chiusura	Pendente
1	Oro	750	500,00	Anello	17,00	Assente		Assente
2	Argento	925	200,00	Orecchino	0,00		Monachella	Assente
3	Argento	925	300,00	Anello	14,00	Presente		Assente
4	Oro	950	2850,00	Orecchino	0,00		Clip	Presente
5	Argento	925	150,00	Anello	18,00	Assente		Assente
6	Oro	750	1980,00	Orecchino	0,00		Farfalla	Presente

Sistema operativo di sviluppo: Linux Mint 17 Qiana
Versione compilatore g++: 4.8.2
Versione Qt: 4.8.5

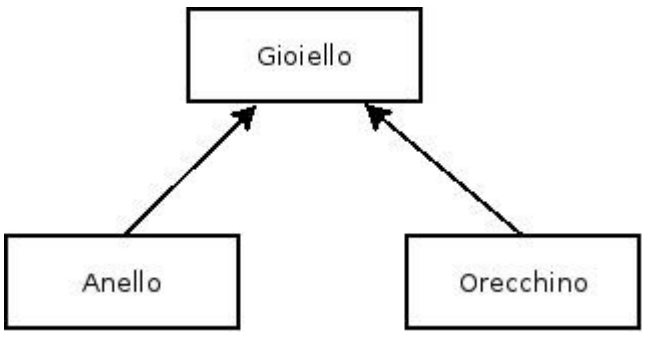
Introduzione

Il progetto è stato sviluppato in ambiente Linux Mint mediante l'editor Qt Creator, senza l'utilizzo di Qt Designer. Inoltre è stato testato nei computer del laboratorio Paolotti dove compila senza problemi. Tra i file consegnati è presente "*Prova.xml*" che consente di popolare rapidamente il contenitore.

Principali scelte progettuali

- Parte logica :

Il template di classe *Container<T>* fornisce funzionalità di inserimento, rimozione, modifica e ricerca di dati. Esso incorpora una classe *ContainerItem*(gli oggetti che sono elementi della lista) e una classe *Iteratore* che permette di iterare sugli elementi in esso contenuti. Inoltre sono stati ridefiniti costruttore di copia, assegnazione e distruttore in modo da renderli profondi.
La realtà modellata è quella di un'ipotetica gioielleria e la gerarchia di classi è la seguente:



Gioiello classe base polimorfa astratta, *Anello* e *Orecchino* sottoclassi concrete.
I campi dati propri di *Gioiello* sono: Materiale, Qualità, Prezzo.
I campi dati propri di *Anello* sono: Misura, Solitario.
I campi dati propri di *Orecchino* sono: Chiusura, Pendente.
Di conseguenza, il contenitore è istanziato con parametro T di tipo *Gioiello** e contiene oggetti che sono puntatori polimorfi alla base della gerarchia.
E' stato effettuato l'overloading dell'operatore ==, reso virtuale in *Gioiello* e opportunamente overridato nelle classi derivate.
Inoltre, il distruttore di *Gioiello* è stato marcato virtuale.
Per l'RTTI è stato utilizzato unicamente il costrutto *dynamic_cast<>()* al fine di individuare a runtime se il puntatore *Gioiello** punta effettivamente ad un oggetto di tipo *Anello* o *Orecchino,* poiché non possono esistere oggetti di tipo *Gioiello*.

- Parte grafica:

La finestra principale del programma è un oggetto di classe *MainWindow* che eredita da QM*ainWindow*. Essa consente all'utente di sfruttare le funzionalità rese disponibili dal contenitore, e di salvare/caricare i dati su/da file esterni in formato xml sfruttando le classi *QXmlStreamReader e QXmlStreamWriter* rese disponibili dalla libreria Qt.
L'inserimento dei dati nel contenitore, a meno che non si tratti di dati provenienti da un file xml(in questo caso è automatico), deve essere effettuato manualmente dopo averli aggiunti alla tabella principale che li rappresenta.
In ogni caso è sempre possibile conoscere quanti elementi sono presenti nel contenitore grazie alla classe *Aiuto,* la quale sfrutta il metodo *size()* del *Container*.
La classe *Table* si occupa della rappresentazione dei dati, è una tabella dinamica che permette di aggiungere(fino ad un massimo di 99), rimuovere e modificare le righe.
Ogni riga della tabella è un oggetto di *RowTable,* classe che collega la parte logica alla parte grafica in quanto gestisce il puntatore polimorfo a *Gioiello,* formato da diversi widget(uno per campo dati + uno per il tipo dinamico del puntatore) modificabili.
Allo scopo di non lasciare garbage è stato ridefinito il distruttore di *Table,* che si occupa di deallocare l'oggetto puntato dal campo dati di tipo *Gioiello** presente in *RowTable* mediante il distruttore virtuale.
Infine, la classe *FindWindow* presenta una finestra di dialogo(eredita da *QDialog)* all'utente in cui inserire un materiale(campo dati di *Gioiello*) per cercarne la presenza all'interno del contenitore.