Progetto di Programmazione Ad Oggetti

Lisien Skenderi - 2023461

1 Introduzione

Questo e' una piccola descrizione del progetto per il corso di Programmazione ad Oggetti. Il progetto e' una semplice applicazione che fa possibile la visualizzazione dei 3 tipi di grafi, lineare, a barre e torta. La applicazione metto a servizio una tabella per inserire i valori di tipo Double dei punti, le barre, e le fette. Con l'abilita di inserire i dati;e possibile inserire anche i nomi dei questi punti, delle barre e delle fette. Solo i nomi dei punti non sono visivi nella dimostrazione del grafo.

L'applicazione ha l'abilita di salvare questi dati in una file tipo .json. Dal fatto che può salvare i dati può' anche aprire dati salvati nel formato .json.

E' possibile interagire con i dati nel seguente modo:

- Aggiungere una linea: Con il bottone *Add Cell* sarà possibile aggiungere una linea nella tabella dei dati. Viene aperto una nuovo finestra che dimostra il numero dove l'utente vuole inserire la nuova riga.
- Eliminare una linea: Con il bottone *Remove Cell* sarà possibile togliere una riga dalla tabella dei dati. Appena premuto sarà aperto una finestra per far scegliere la riga di togliere.
- Cambiare il nome della tabella dei dati: Dentro la menu sotto la voce di *Graph Name* sarà possibile cambiare il nome del grafo. Questo nome viene vista anche nella sezione del grafo.

Altri operazioni possibili sono i cambiamenti tra i grafi, il salvataggio dei dati scritti e l'apertura dei dati. Se i dati vengono messi nella tabella dei dati il grafo non e viene cambiato quindi e' necessario premere il bottone *Update Graph* che aggiorna la vista del grafo con i dati inseriti.

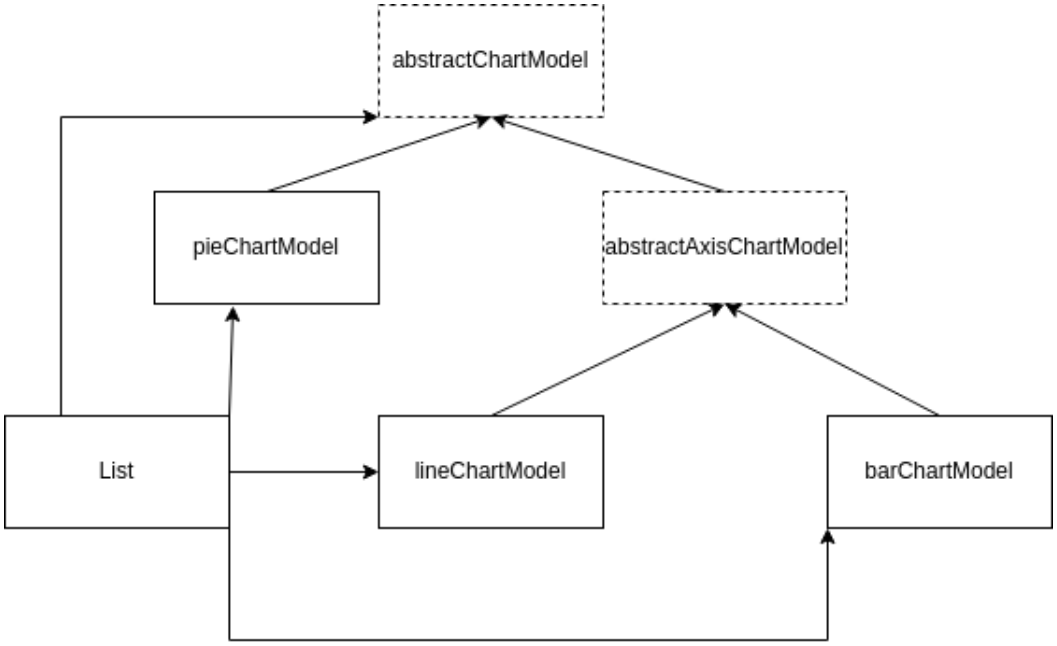
2 Gerarchia dei tipi

E' stato usato il pattern architetturale **Model-View-Controller** in modo per separare la vista e modello, e il controller che attua come un via di mezzo tra loro. Il progetto ha 2 gerarchie diverse, una per la parte di modello e l'altra per la parte grafiche che costruisce i grafi.

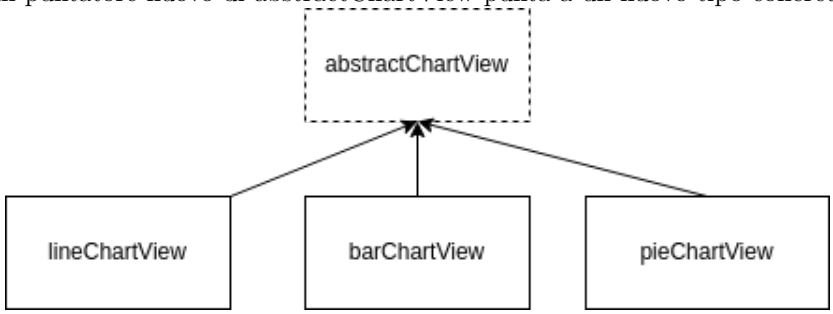
La parte di modello inizia con una classe base astratta con il nome abstractChartModel che ha le metodi virtuali puri come getData setData, removeData, insertData. Questo sono spiegato a i suoi nomi. I altri metodi virtuali puri sono getCounter che prende il numero di elementi che sono memorizzati, ho piu' chiaro le righe che sono rappresentate nella tabella. Poi e' updateCounter che cambia aggiorna la proprietà' delle classe derivate che conta il numero dei elementi inseriti. Poi e' l'ultimo metodo astrato che si chiama getDerivType che ritorna un intero dipendente dal tipo derivato. Viene usato nel posto di dynamic_cast$\leq\geq$.

L'altra classe astratta nella gerarchia e abstractAxisChartModel con i metodi astratte getMaxAxi e getMinAxi che ritornano una lista ciascuno. Se la classe concreta e lineChartModel ritorna una lista con 2 nodi il primo l'asse 0X e il secondo per l'asse 0Y. La getMinAxi funziona nel stesso modo. Se la classe concreta e un barChartModel le lista anno solo un nodo, solo per l'asse 0Y.

La classe lista e' una template di nodi doppio linkati. Ha le metodi per inserire elementi come push_back(), push_front(), insert_element(..). E altre metodi per togliere elementi come pop_front(), pop_back() e del_element(..). Gli uniche metodi const sono get_length() e get_element(..).



La gerarchia della vista e composta da una classe base astratta e 3 classe che lo ereditano. La classe base ha 2 metodi virtuali puri importanti. Questa gerarchia attua come una fattoria per i grafi. Le classe prendono i dati mandati dal controller di cui lui lo prenda dal model. L'altro metodo ritorna un puntatore nuovo di abstractChartView punta a un nuovo tipo concreto.



Poiché il controller contiene un puntatore alla vista principale MainWindow, un puntatore a abstractChartModel , un puntatore a abstractChartView, e l'attributo di tipo fileDataH che contiene i dati dei grafi e i metodi per scrivere e aprire file di tipo .json con la struttura apposta.

Altra classe nel progetto e la classe MainWindow che contiene tutta la parte che visiva che l'utente po interagire come i bottoni, menu e la tabella, di cui sono le widget della frameWork Qt.

3 Polimorfismo

3.1 Model

L'uso del polimorfismo nella parte del modello e' composta dal distruttore virtuale che e' banale e le altri 4 più importanti:

- getData(): Questo metodo ritorna un puntatore constante a una lista di double che mantiene tutti i dati inseriti nella seconda o la terza colona. Ha un comportamento speciale per il grafico di linea, dove i dati di x vengono salvati a i posti dispari e i dati di y vengono salvati a i posti pari della lista.
- setData(...): Questo metodo va a cambiare il valore del nodo x con il valore d. Se questo nodo non esiste allora lo inserisce. Vengono inseriti solo quelli dati che lo chiamano con un x maggiore del counter().

- insertData(...): Questo metodo inserisce nuovo nodo nella lista dei dati del modello. Se il modello e una lisa allora inserisce 2 nodi per simulare x e y inseriti.
- removeData(...): Questo metodo toglie dalla lista dei dati il nodo nella posizione x. Se il modello e' una lista allora toglie sia il X che il Y.

Le altre metodi virtuali erano spiegato sopra e molto semplice di capire dal i suoi nomi da solo. L' altri metodi virtuali residuano nella classe abstractAxisChart.

3.2 View

La gerarchia della vista e composta da una classe base astratta e 3 classe che lo ereditano. La classe base ha 2 metodi virtuali puri importanti:

- getChartView(..): Questo metodo prende i dati mandati dal controller e costruisce un grafo dipendente dal tipo concreto. Alla fine ritorna un puntatore di QChart.
- getRightType(): Questo metodo ritorna un puntatore di tipo abstractChartView che punta a un tipo concreto della gerarchia. E' usato per evitare segmentation fault.

4 Persistenza dei Data

Per la persistenza e' usato il formato JSON. Questa file contiene un Jobject per ogni grafo che l'applicazione può creare. Ogni oggetto che contiene i data contiene almeno 2 Json Array e un Jobject per il nome. Il primo array contiene i nomi dei ponti, delle barre e delle fette. Il secondo contiene i valori dei elementi, se Jobject e un grafo di linea ha 2 array, una per i valori di X e l'altra per i Y.

5 Funzionalità Implementate

La GUI e' molto semplice da gestire e usare. Le implementazione grafiche' sono un collezione di bottoni e voci di menu che modificano la tabella e' i dati che salva, la vista del grafo e il nome e posto dove il file di dati viene salvato.

6 Rendicontazione ore

Attività'	Ore Previste	Ore Effettive
Studio e progettazione	10	12
Sviluppo del codice del modello	10	11
Studio del framework Qt	10	14
Sviluppo del codice della GUI	10	13
Test e debug	5	6
Stesura della relazione	5	3
Totale	50	59

7 Compilazione

Per compilare il progetto via terminale e' necessaire avere installato i file header di Qt6 e poi dopo avere aperto il terminale nella cartella che contiene i file del progetto scrivere nel terminale:

- qmake -testing.pro
- make

Poiché sarà possibile aprire l' applicazione.