KalkPro

GRIGORAS VALENTIN

1099561

Progetto del corso di Programmazione ad Oggetti a.a 2017/2018

SOMMARIO

PREFAZIONE

Lo scopo del progetto

Lo scopo del progetto e lo sviluppo in C++ di una calcolatrice estendibile basata sulle figure geometriche dotata di una interfaccia utente grafica (GUI) sviluppata in Qt e in grado di effettuare delle operazioni sulle figure implementate.

Strumenti utilizzati

La versione in java è utilizzabile solo da linea di comando.

La versione in c++ dispone anche di un'interfaccia grafica creata utilizzando la libreria Qt.

L'intero progetto è stato realizzato utilizzando Qt Creator 4.7.0 installato su una macchina avente Windows 10 Enterprise 64bit ed in fine testato sulla macchina virtuale con Ubuntu dotata di QtCreator 4.5 messa a disposizione dal professore. Il compilatore installato su QtCreator è il MinGW 4.9.2 32bit.

Sulla macchina virtuale è installata la versione 5.4.0 di Gcc.

Numero di ore richieste per la realizzazione del progetto

- 4 ore per l'analisi e la definizione dei requisiti del progetto
- 4 ore per la progettazione del modello
- 2 ore per la progettazione dell'interfaccia GUI
- 13 ore per l'apprendimento della libreria QT
- 5 ore per la codifica del modello
- 2 ore per la progettazione e codifica della classe Eccezioni
- 10 ore per la codifica dell'interfaccia GUI
- 10 ore per la codifica del controller
- 3 ore per il debugging e testing

KalkPro

Funzionalità principali

Kalk Pro è una calcolatrice che permette di fare alcune operazioni sulle figure geometriche.

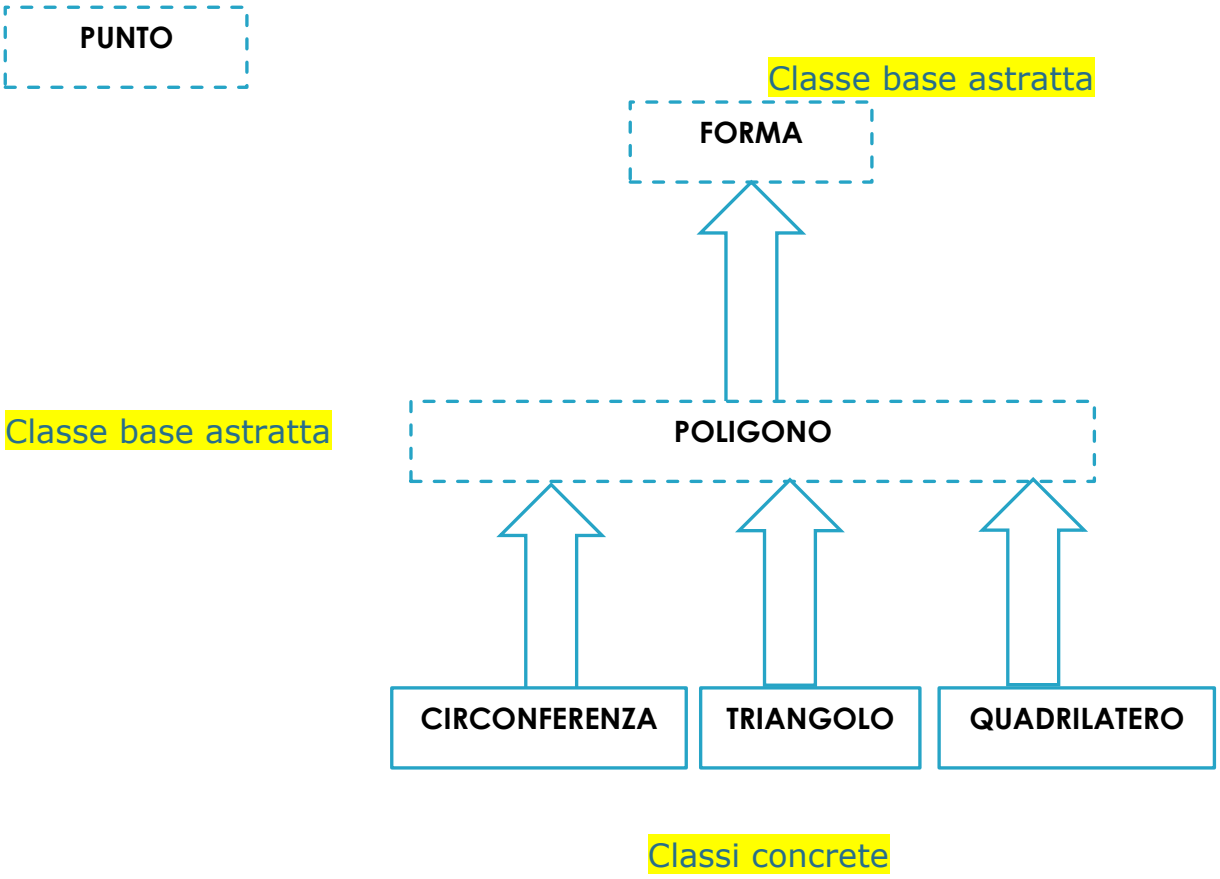
Le figure geometriche implementate in KalkPro sono:

- Circonferenza
- Triangolo
- Quadrilatero

Tutte le operazioni sono effettuabili sulle singole figure. Di seguito sono elencate tutte le operazioni:

- **CIRCONFERENZA**
  - AREA: che ritorna il risultato dell'operazione raggio * raggio * pigreco;
  - PERIMETRO: che ritorna il risultato dell'operazione 2 * raggio * pigreco;
  - BARICENTRO: viene disegnato sulla tavola da disegno un punto che rappresenta il baricentro della circonferenza corrente.
  - MODIFICA CIRCONFERENZA: è possibile inoltre modificare con il mouse il raggio della circonferenza corrente selezionando il bordo della circonferenza spostandolo verso interno o esterno.
- **TRIANGOLO**
  - AREA: che ritorna il risultato dell'operazione sqrt(semiPerimetro * (semiPerimetro-l[0]) * (semiPerimetro-l[1]) * (semiPerimetro-l[2])); dove l[n] rappresenta n-esimo lato del triangolo.
  - PERIMETRO: ovvero la somma di tutti i lati del triangolo.
  - LUNGHEZZA LATI: mediante l'invocazione della funzione FindLati() che si occupa di chiamare(poli)->getLati();. La funzione getLati() mi ritorna un vector di punti che attraverso la funzione di distanza mi calcolo la lunghezza di ogni lato.
- **QUADRILATERO**
  - AREA: il quadrilatero viene diviso in due triangoli e poi viene calcolata l'area di ciascun triangolo; la somma delle due aree rappresenta il valore ritornato dalla funzione Area.
  - PERIMETRO: ovvero la somma di tutti i lati del quadrilatero.
  - LUNGHEZZA LATI: mediante l'invocazione della funzione FindLati() che si occupa di chiamare(poli)->getLati();. La funzione getLati() mi ritorna un vector di punti che attraverso la funzione di distanza calcola la lunghezza di ogni lato.

Gerarchia



La classe Punto

Questa è una classe concreta che non deriva da nessuna classe.

La classe Punto ha due campi dati privati di tipo double che indicano le coordinate all'interno della tavola ed una serie di metodi pubblici che aggiungono funzionalità alla classe:

- costruttore a zero, uno o due parametri che crea di default un punto nell'origine: (0,0)
- costruttore di copia
- distruttore non virtuale che assume il comportamento di default definito dallo standard C++.

La classe Forma e le sue classi derivati

FORMA

Questa è una classe base astratta in quanto contiene tre metodi virtuali puri:

- Area(),
- Perimetro()
- Baricentro()

Contiene inoltre un distruttore virtuale che verrà ereditato dalle classi derivate.

POLIGONO

Questa è una classe derivata perché eredità tutti i metodi della classe FORMA. E' inoltre astratta in quanto ha il metodo virtuale puro Area() implementato nelle classi derivati. Ha inoltre uno scopo di estensibilità, in quanto in futuro si possono aggiungere nuove classi senza modificare la gerarchia.

La classe **poligono** ha come campo dati un vector di puntatori a Punti che contiene i puntatori dei vari punti dei vertici di un poligono. Contiene inoltre una serie di metodi pubblici che aggiungono funzionalità alla classe:

- Costruttore a zero o uno argomenti con valore di default.
- Un metodo virtuale puro: Area().
- Un metodo per il calcolo del perimento. Questo metodo è stato ereditato dalla classe forma e quindi tassativamente implementato in questa classe.
- Vari metodi getter/setter per la gestione del campo dati.
- Un distruttore virtuale
- Un metodo ContaVertici() per definire il tipo di forma: tre vertici per il triangolo e quattro vertici per il quadrilatero.
- Un metodo protetto AddVertice() che serve ad aggiungere un vertice nel vector di punti.
- Un metodo GetLato() che ritorna l-esimo lato del vector.

CIRCONFERENZA

Questa è una classe derivata CONCRETA perché eredità tutti i metodi della classe FORMA e inoltre implementa tutti i metodi virtuali puri della classe base.

La classe **circonferenza** ha come campi dati un'oggetto Punto e una variabile di tipo double che rappresenta il raggio della circonferenza. Contiene inoltre una serie di metodi pubblici che aggiungono funzionalità alla classe:

- Costruttore a due argomenti.
- Un costruttore di copia.
- I tre metodi ereditati dalla classe forma e quindi tassativamente implementati in questa classe.
- Vari metodi getter/setter per la gestione dei campi dati.
- Un distruttore virtuale

TRIANGOLO

Questa è una classe derivata CONCRETA perché eredità tutti i metodi della classe FORMA e inoltre implementa tutti i metodi virtuali puri della classe base.

La classe **triangolo** non ha campi dati propri. Contiene inoltre una serie di metodi pubblici che aggiungono funzionalità alla classe:

- Costruttore che dati 3 Punti costruisce un triangolo.
- I due metodi virtuali puri ereditati dalla classe forma e quindi tassativamente implementati in questa classe.
- Un metodo che calcola l'altezza del triangolo.

QUADRILATERO

Questa è una classe derivata CONCRETA perché eredità tutti i metodi della classe FORMA e inoltre implementa tutti i metodi virtuali puri della classe base.

La classe **quadrilatero** non ha campi dati propri. Contiene inoltre una serie di metodi pubblici che aggiungono funzionalità alla classe:

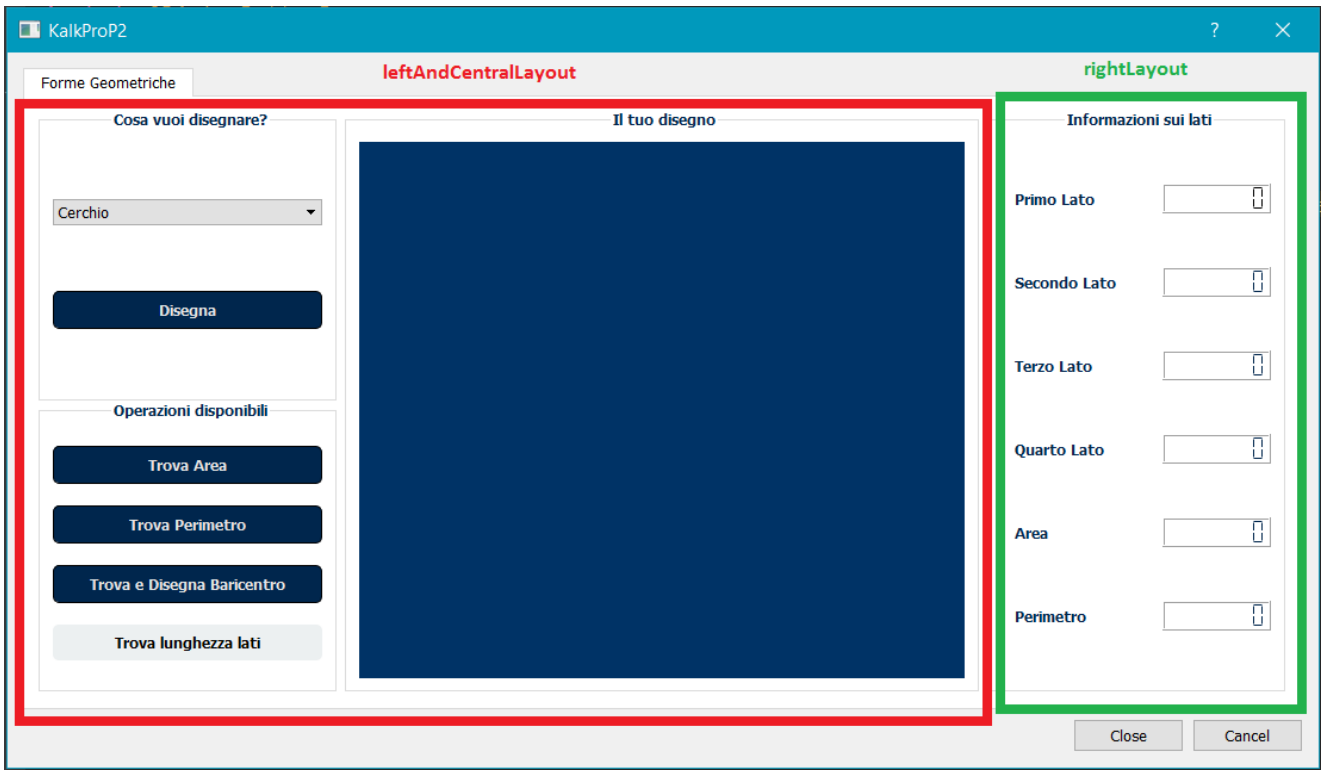
- Costruttore che dati 4 Punti mi costruisce un quadrilatero.
- Un metodo virtuale puro ereditato dalla classe forma e quindi tassativamente implementato in questa classe.
- Un distruttore virtuale.

La GUI

QTABDIALOG

La classe QTabDialog deriva dalla QDialog che è la classe madre delle finestre di dialogo.

Ho scelto questo modo di visualizzare i dati in quanto è più facile aggiungere in futuro gerarchie nuove senza stravolgere la view dell'applicazione.La QTab Dialog è divisa in 2 macro aree:



La **leftAndCentralLayout** contiene una comboBox per scegliere il tipo di forma che si vuole disegnare e quattro QPushButton che in base al tipo di forma scelto diventano abilitati oppure no. Al centro troviamo invece la tavola da disegno che visualizzerà il disegno scelto e su cui si possono apportare modifiche sulla forma.

La **rightLayout** contiene le informazioni delle varie componenti di una forma con le relative misure.

La **classe TabDialog** contiene un puntatore alla bottomBox contenente i pulsanti Close e Cancel che permettono all'utente di chiudere rapidamente l'applicazione e un puntatore alla tabForme che a sua volta contiene tutti i puntatori necessari alla costruzione della view.

Il polimorfismo

Il polimorfismo viene usato molto spesso, ad esempio per il recupero dell'area e del perimetro utilizzando un puntatore a Poligono.

Viene inoltre utilizzato nella funzione drawPolygon che permette accuratamente di scegliere la giusta figura da disegnare applicando uno static cast al puntatore poli di tipo Poligono.

La classe DataManager serve a gestire le informazioni tra il model e la view ed evitare problemi di conversioni a run-time.

Il materiale consegnato

La cartella principale contiene le cartelle Model, View, Controller, Eccezioni.

La cartella Model contiene:

```
forma.cpp
punto.cpp
poligono.cpp
circonferenza.cpp
triangolo.cpp
quadrilatero.cpp

forma.h
punto.h
poligono.h
circonferenza.h
triangolo.h
quadrilatero.h
```

La cartella View contiene:

```
tabdialog.cpp
formetab.cpp
drawbox.cpp

tabdialog.h
formetab.h
drawbox.h
```

La cartella Controller contiene:

```
datamanager.cpp

datamanager.h
```

La cartella Exception contiene:

```
eccezioni.cpp

eccezioni.h
```

NB: Per generare il MakeFile è indispensabile utilizzare il file KalkPro.pro

La cartella Java contiene tutti i file .java

Non ritengo necessaria un guida per l'utilizzo in quanto l'applicazione risulta molto intuitiva.