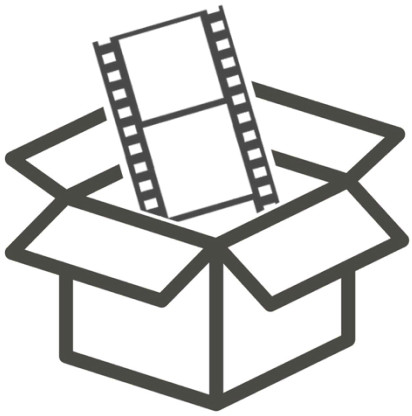


moviecontainer

Progetto di Programmazione ad Oggetti
A.A. 2018/2019

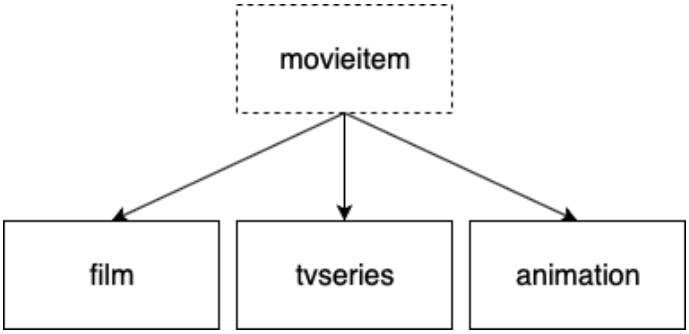
Francesco Battistella
1102842

1. Descrizione generale

MovieQontainer è un contenitore che si occupa di gestire una gerarchia di item che riguardano il mondo dei film, delle serie tv e degli anime/cartoni animati. Tale contenitore vuole fornire un modo pratico e veloce per tenere traccia di questi item visti o in progetto di vedere, salvandone diverse informazioni come il titolo, la durata, l'anno di uscita, il genere e la trama e altre informazioni che variano a seconda del tipo inserito.

2. Descrizione delle classi

La gerarchia di classi usata per modellare la realtà in questione è la seguente:



MovieItem

Ereditarietà

È la classe base della gerarchia. Per modellare la realtà in questione è stato opportuno rendere tale classe astratta in quanto le classi derivate presentato diversi attributi in comune. Il distruttore è virtuale.

Attributi

Title: tipo string che rappresenta il titolo dell'item.
Duration: tipo int che rappresenta la durata in minuti.
Genre: tipo string che rappresenta il genere.
Year: tipo int che rappresenta l'anno di pubblicazione.
Plot: tipo string che rappresenta la trama.
Type: tipo int che rappresenta il tipo dell'item in forma numerica ('1' per Film, '2' per Serie Tv, '3' per Anime/cartoni).

Metodi polimorfi

Il metodo dichiarato *virtual* in *movieitem* è l'operatore di confronto ==.
Il metodo che rende astratta la classe dichiarato *virtual* puro (=0) è il metodo Type() che restituisce uno string adeguato per ogni sottoclasse in cui è ridefinito ("Film", "Serie", "Anime").

Metodi importanti

I metodi dichiarati in *movieitem* sono i vari metodi per accedere ai diversi attributi della classe o modificarli: getTitle() const, setTitle(const string&), getDuration() const, setDuration(const int&), getGenre() const, setGenre(const string&), getYear() const, setYear(const int&), getPlot() const, setPlot(const string&), getType() const, setType(const int&).

Film

Ereditarietà

Deriva pubblicamente da *movieitem* ed è una classe concreta perché va a definire il metodo Type() virtuale puro. Rappresenta un film visto o da vedere. La classe ridefinisce il costruttore, il distruttore (che è uguale a quello di default) e il costruttore di copia aderendo alla *Rule of three*.

Attributi

Director: tipo string che rappresenta il regista dell'item.
Saga: tipo string che rappresenta la saga di cui fa parte il film in caso ne facesse parte, *null* altrimenti.
Country: tipo string che rappresenta i Paesi dove è stato girato il film.

Metodi polimorfi

La classe ridefinisce l'operatore di confronto == dichiarato virtuale in *movieitem* e definisce il metodo Type() virtuale puro restituendo il tipo in formato string: "Film".

Metodi importanti

I metodi dichiarati in *film* sono i diversi metodi per accedere ai vari attributi della classe o modificarli: getDirector() const, setDirector(const string&), getSaga() const, setSaga(const string&), getCountry() const, setCountry(const string&).

TvSeries

Ereditarietà

Deriva pubblicamente da *movieitem* ed è una classe concreta perché va ad definire il metodo Type() virtuale puro. Rappresenta una serie tv vista o da vedere. La classe ridefinisce il costruttore, il distruttore (che è uguale a quello di default) e il costruttore di copia aderendo alla *Rule of three*.

Attributi

Ideator: tipo string che rappresenta l'ideatore dell'item.
Season: tipo int che rappresenta il numero della stagione se la serie è divisa in stagioni, 0 se è un'unica stagione.
Episodes: tipo int che rappresenta il numero di episodi che fanno parte della stagione considerata.

Metodi polimorfi

La classe ridefinisce l'operatore di confronto == dichiarato virtuale in *movieitem* e definisce il metodo Type() virtuale puro restituendo il tipo in formato string: "Serie".

Metodi importanti

I metodi dichiarati in *tvseries* sono i diversi metodi per accedere ai vari attributi della classe o modificarli: getIdeator() const, setideator(const string&), getSeason() const, setSeason(const int&), getEpisodes() const, setEpisodes(const int&).

Animation

Ereditarietà

Deriva pubblicamente da *movieitem* ed è una classe concreta perché va ad definire il metodo Type() virtuale puro. Rappresenta un film d'animazione tipo Anime (Giapponese) o Cartone animato (Americano) visto o da vedere. La classe ridefinisce il costruttore, il distruttore (che è uguale a quello di default) e il costruttore di copia aderendo alla *Rule of three*.

Attributi

Country: tipo string che rappresenta il paese di produzione dell'item.
Season: tipo int che rappresenta il numero della stagione se l'opera è divisa in stagioni, 0 se è unica.
Episodes: tipo int che rappresenta il numero di episodi che fanno parte della stagione considerata.

Metodi polimorfi

La classe ridefinisce l'operatore di confronto == dichiarato virtuale in *movieitem* e definisce il metodo Type() virtuale puro restituendo il tipo in formato string: "Anime".

Metodi importanti

I metodi dichiarati in *animation* sono i diversi metodi per accedere ai vari attributi della classe o modificarli: getCountry() const, setCountry(const string&), getSeason() const, setSeason(const int&), getEpisodes() const, setEpisodes(const int&).

Qontainer<T>

Qontainer<T> è stato implementato sotto forma di un array ridimensionabile in stile std::vector prendendone spunto per la realizzazione dei vari metodi. Il Qontainer fa uso di altre due classi annidate:

1. iterator: iteratore usato per scorrere l'array in lettura e in scrittura, implementato seguendo i principi visti in aula;
2. const_iterator: iteratore costante usato per scorrere l'array in lettura, implementato seguendo i principi visti in aula.

Vista l'assenza di operazioni che copiano interamente il qontainer è stato deciso di non implementare un eventuale DeepPtr<T>.

3. Uso del polimorfismo

Il polimorfismo è stato applicato nella gerarchia con i vari metodi virtuali come il distruttore virtuale, l'operatore di uguaglianza == virtuale e il metodo virtuale puro nella classe base Type() che, come descritto in precedenza, restituisce il tipo della classe in formato string.
Nell'interfaccia grafica, quando si clicca sul bottone "Elimina" presente nel dialog di visione degli item o nel dialog dove verranno visualizzati i risultati di una ricerca, viene invocato il metodo void remove(T) definito in Qontainer<T>, il quale richiama l'operatore di confronto ridefinito virtuale bool operator==(const T&) già visto precedentemente.

4. Compilazione ed esecuzione

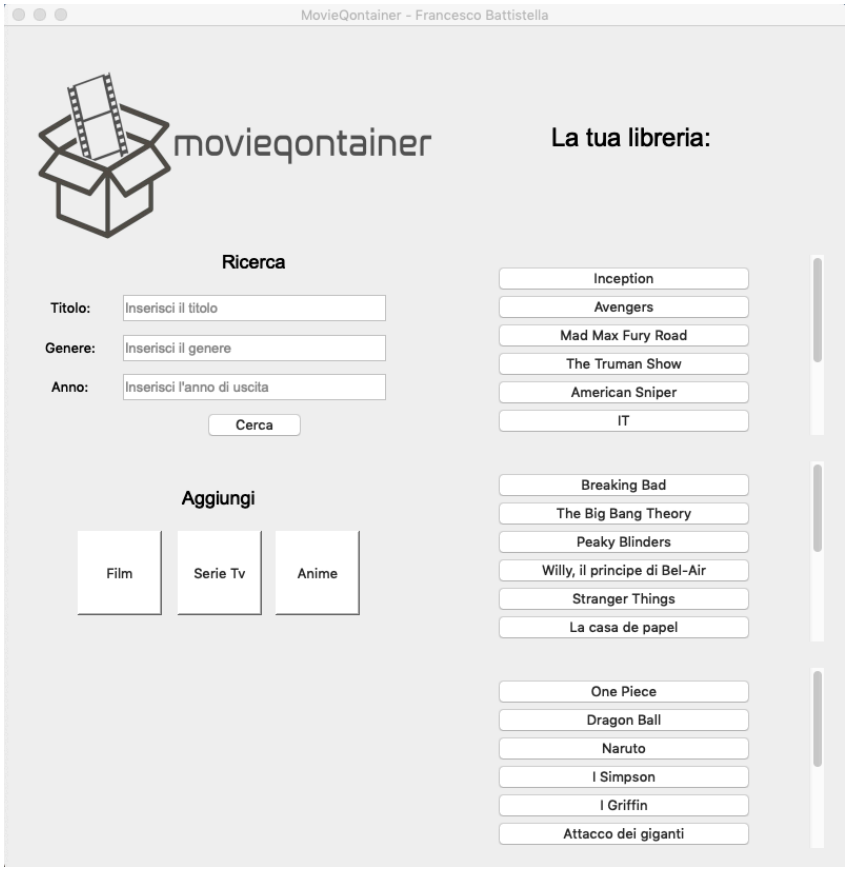
Il file MovieQontainer.pro che è stato consegnato è stato modificato aggiungendo Qt += widgets, Qt += core, Qt += gui, QMAKE_CXXFLAGS += -std=c++11, CONFIG += sdk_no_version_check per la corretta compilazione del progetto.
La compilazione si esegue tramite i comandi *qmake* e *make.*

5. Caricamento/salvataggio dei dati

All'apertura della GUI, all'utente viene chiesto di scegliere il file da caricare di tipo xml (che si trova nella root folder del progetto, MovieQontainer, sotto il nome di data.xml).
È stato scelto di definire una classe dedicata managefile la quale, mediante i metodi load(Qontainer<movieitem*>&) e save(Qontainer<movieitem*>&), si occupa di caricare dati da un file di tipo .xml e di salvarli sul medesimo file. Il metodo load è invocato nella GUI all'apertura (invocazione del costruttore della main window), mentre il metodo save viene invocato dopo l'aggiunta di un elemento o alla chiusura della main window (mediante l'overriding del metodo QCloseEvent, come da documentazione Qt) per salvare eventuali modifiche o eliminazioni.

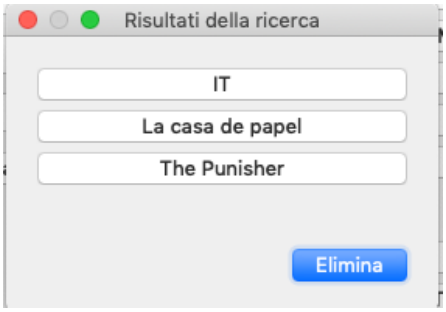
6. Interfaccia grafica

L'interfaccia grafica vuole essere il più essenziale possibile. Per questo è stato deciso di implementare tutte le azioni possibili nella main window.
Nella parte sinistra sono presenti il logo, i campi per effettuare una ricerca o i 3 pulsanti per aggiungere rispettivamente un Film, una Serie o un Anime. Nella parte destra invece sono visibili 3 aree scrollabili dove sono presenti: nella prima area i film inseriti, nella seconda area le serie tv e nella terza area i film d'animazione.



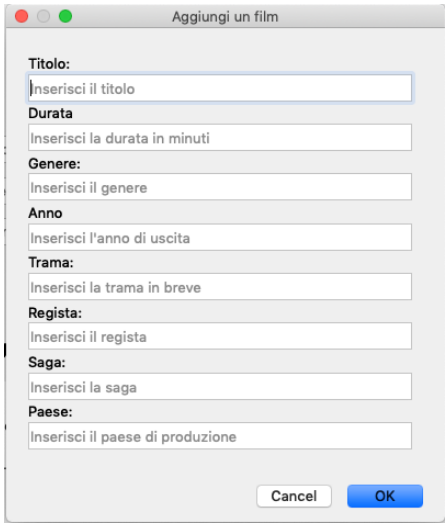
6.1 Ricerca

La GUI mette a disposizione diversi tipi di ricerca: è possibile ricercare un elemento attraverso solo il titolo, solo il genere o solo l'anno di pubblicazione. È anche possibile effettuare una ricerca attraverso titolo e genere, titolo e anno o genere e anno, o inserendo tutti i campi richiesti.
Al click del pulsante "Cerca" se non ci sono risultati si otterrà un messaggio di errore, altrimenti si aprirà una QDialog contenente tutti i risultati ottenuti. Da questa QDialog sarà possibile, cliccando sul titolo, vedere tutti gli attributi dell'elemento selezionato oppure eliminare dal qontainer tutti i risultati ottenuti attraverso il pulsante "Elimina".



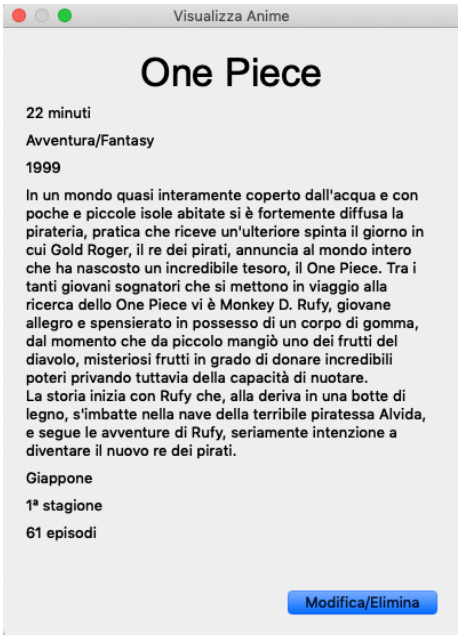
6.2 Aggiungi

Cliccando uno dei pulsanti di aggiungi si aprirà una QDialog contenente tutti i campi necessari ad aggiungere un nuovo Film, Serie Tv o Anime che saranno diversi tra loro. All'omissione o all'inserimento di una stringa al posto di un intero il programma considererà "null" o '0' l'inserimento non andando in errore. Al click del pulsante "Ok" l'elemento sarà aggiunto al qontainer e sarà visualizzato nella lista di film, serie o anime nella HomePage.



6.3 Visualizza Elemento

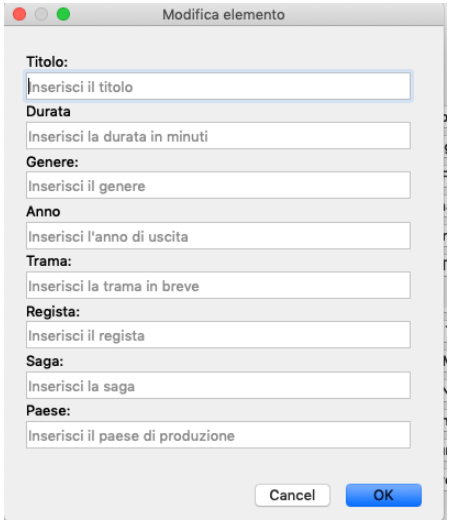
Cliccando sul titolo nella lista di film, serie o anime o nel QDialog contenente i risultati della ricerca l'utente sarà rimandato in una nuova pagina contenente tutti gli attributi dell'elemento selezionato. Da questa pagina è inoltre possibile richiamare la pagina di modifica o di elimina cliccando sull'apposito pulsante "Modifca/Elimina".



6.4 Modifica/Elimina

In questo QMessageBox l'utente sarà invitato a scegliere se vuole tornare indietro, cliccando sul pulsante "Indietro", modificare l'elemento attraverso il pulsante "Modifca" o eliminare l'elemento dal qontainer attraverso il pulsante "Elimina" ma solo dopo aver confermato la volontà di cancellare l'elemento.

Il pulsante "Modifica" farà apparire un altro QDialog dove saranno presenti diversi campi, a seconda se si tratta di un film, una serie o un anime, dove sarà possibile immettere il nuovo valore da dare all'item.



7. Tempo richiesto

Analisi preliminare del problema:	2h
Progettazione del modello:	1h
Progettazione della GUI:	2h
Apprendimento della libreria Qt:	10h
Codifica modello e GUI:	27h
Debugging:	4h
Testing:	4h
Totale:	50h

8. Specifiche

Sistema Operativo di sviluppo: Mac OS Mojave 10.14.6
Compilatore: Apple LLVM version 10.0.1 (clang-1001.0.46.4)
Versione Qt: Qt 5.11.2
Versione Qt Creator: 4.7.2