Media QIlection

Progetto di
Programmazione ad Oggetti
2018/2019

Luca Marcon
1070602

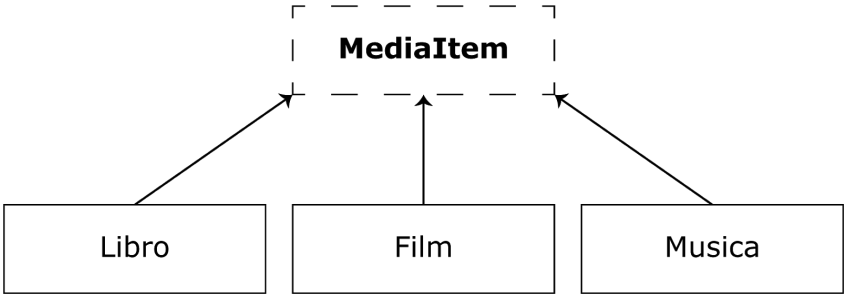
Descrizione

Media Qllection è un'applicazione con lo scopo di gestire un contenitore di media - musica, film e libri - che ha la principale utilità di archiviare i media di cui si è in possesso e di cui si ha usufruito.
L'applicazione, oltre al salvataggio e al caricamento automatico della lista, permette di inserire, modificare, rimuovere e cercare media secondo determinati parametri. Sono presenti infine delle funzioni di ordinamento alfabetico per titolo e di calcolo dell'intrattenimento totale, secondo una formula che somma la durata di tutti i film, di tutta la musica e il totale delle pagine (considerando tre minuti per la lettura di una pagina).

Progettazione

Il design pattern a cui si è deciso di aderire è il Model-View. Gli strumenti forniti da Qt, in relazione alla flessibilità di questo pattern, hanno permesso facilità nel mantenere modello e vista separati e uno sviluppo del progetto con tempistiche adeguate a quelle richieste. Pur consapevole della possibilità di utilizzare il pattern Model-View-Controller, è stata fatta a priori la scelta del Model-View per poter completare il progetto soddisfando le richieste della consegna nel modo più aderente possibile.

Gerarchia di tipi



La classe base astratta polimorfa MediaItem è la classe base della gerarchia e rappresenta un media generico che non ha una tipologia, ma è dotato di attributi generici comuni a tutti i media: le *stringhe* titolo, genere e formato, gli *unsigned int* annoDistribuzione e recensione - rilasciata dall'utente (espressa in numero di cuori da 1 a 5). Si tratta di una classe base astratta perché contiene il metodo virtuale puro `durataTOT()`, il quale verrà implementato dalle classi discendenti (come già accennato

nella descrizione, questo metodo virtuale puro permetterà, tramite chiamate polimorfe, di calcolare la durata totale dei media). Oltre a ciò, contiene dei semplici metodi di accesso e inizializzazione dei campi dati e un metodo virtuale **`info()`** che stampa nello stream di output una stringa con i campi dati. Ovviamente di quest'ultimo metodo, essendo virtuale, verrà fatto l'override nelle classi figlie, dove manterrà l'output dei campi dati di MediaItem, aggiungendoci inoltre i campi dati della classe che lo ha implementato. Infine sono presenti anche dei metodi per il comportamento degli operatori di uguaglianza, disuguaglianza e confronto.

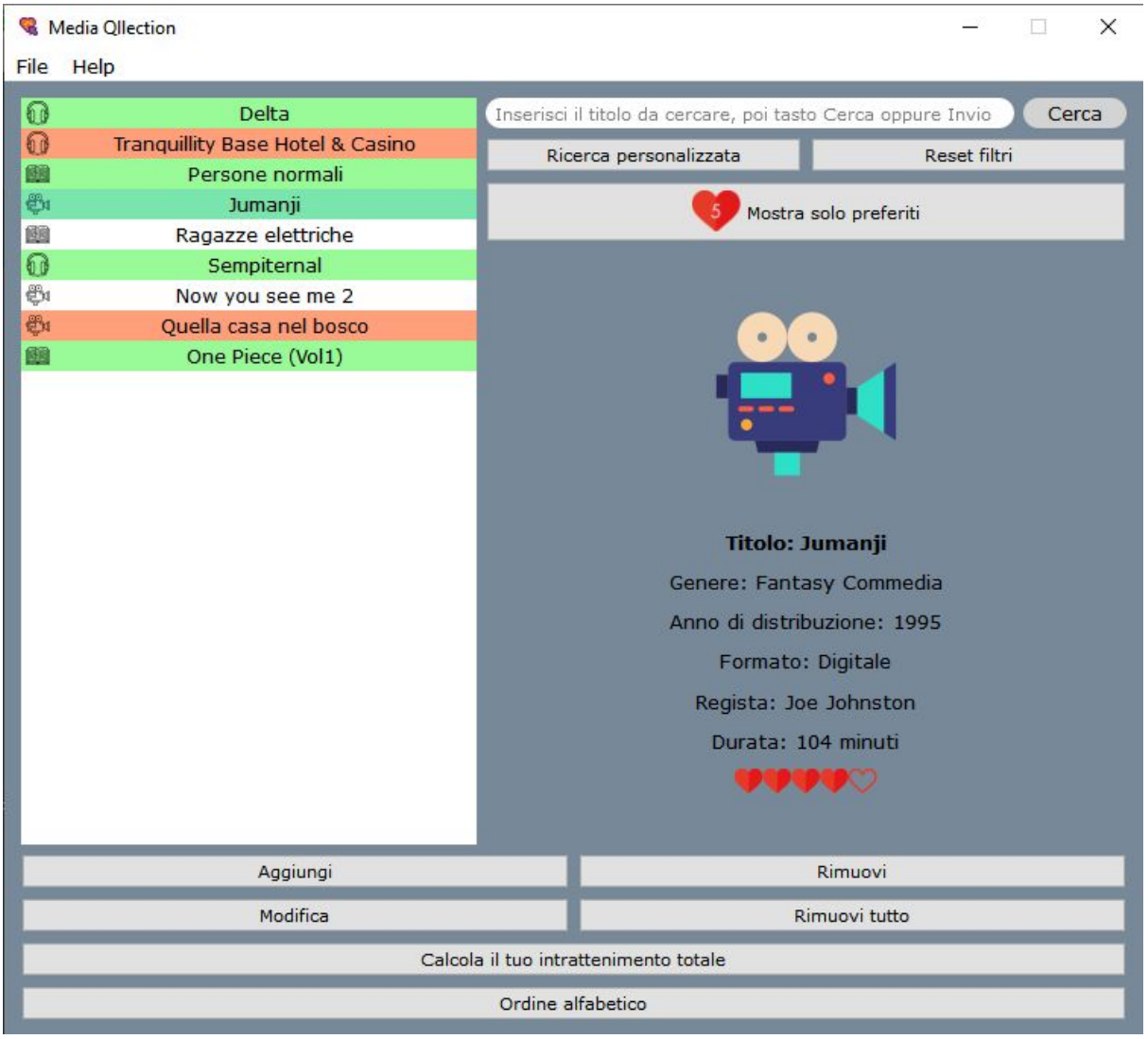
I sottotipi diretti della classe MediaItem sono: **Libro**, **Film** e **Musica**

- La classe **Libro** è caratterizzata dai campi dati autore (di tipo *string*), in cui si riportano nome e cognome dell'autore del libro, casaEditrice (di tipo *string*) che identifica con un'altra stringa il nome della casa editrice e numeroPagine (di tipo *unsigned int*) che memorizza il numero di pagine del libro. Libro contiene inoltre l'override dei metodi virtuali **`info()`** e operatori di uguaglianza e disugualianza. Ha dei metodi per accedere e modificare i campi dati. Infine implementa il metodo virtuale **`durataTOT()`**, restituendo il numero delle pagine moltiplicato per 3 (3 minuti per la lettura di una pagina).
- **Film** contiene due campi dati oltre a quelli ereditati da MediaItem: il nome del regista di tipo *string* e la durata del film di tipo *unsigned int*. Oltre ai classici metodi per accedere ("*get*") e per modificare ("*set*") i campi dati, anche Film ridefinisce **`info()`** e gli operatori == e !=. Qui l'implementazione di **`durataTOT()`** semplicemente restituisce la durata del film ottenuta tramite **`getDurata()`**.
- **Musica**: in modo similare alle altre classi dello stesso livello di gerarchia, Musica fa l'overriding di **`info()`**, == e != ed implementa in modo personalizzato **`durataTOT()`**. Poichè i campi dati di musica sono artista (di tipo *string*), numeroTracce(che tiene nota del numero di canzoni dell'album) e durata (quest'ultimi due di tipo *unsigned int*), allora **`durataTOT()`** sfrutta il metodo **`getDurata()`** per ottenere la durata totale dell'album. Questa classe lascia libera interpretazione all'utilizzatore dell'applicazione, in quanto può tenere nota sia di album musicali, che di raccolte, che di singole canzoni. In quest'ultimo caso (pur poco verosimile visto lo scopo primario dell'applicazione di tenere nota di grandi quantità di dati) sarà sufficiente impostare il numero di tracce a 1.

Risulterà quindi chiaro che i metodi polimorfi utilizzati sono gli operatori di uguaglianza, di disuguaglianza, della stampa di informazioni (metodo **`info()`**) e di calcolo della durata totale (metodo **`durataTOT()`**) di cui si sono ampiamente spiegate le funzionalità in

precedenza. Le chiamate polimorfe su questi metodi fanno sì che abbiano una funzionalità personalizzata in base al tipo dinamico dell'oggetto di invocazione.

GUI ed Utilizzo del software



La GUI presenta tre layout principali:

- il primo, a sinistra, contiene una lista di tutti gli oggetti finora inseriti denominati per titolo e contraddistinti da un simbolo personale: Libro, Film e Musica sono ben distinguibili grazie a tre diverse icone. Inoltre, per una più semplice visualizzazione "a colpo d'occhio", si è deciso di impostare un colore di sfondo per ogni elemento in base alla recensione ottenuta al momento dell'inserimento: i media che hanno ricevuto una recensione bassa (1 cuore) hanno lo sfondo rosso, quelli che invece hanno ottenuto una valutazione ottima o eccellente(4 e 5 cuori) hanno lo sfondo verde

- il secondo layout, a destra, contiene le principali funzionalità di ricerca e di filtro. Oltre alla ricerca testuale per titolo è possibile impostare un filtro personalizzato che mostra solo i media di tipo Libro, o quelli di tipo Film, o quelli di tipo Musica. Altra funzionalità è il filtro per "preferiti": permette di visualizzare solo i media a cui è stata conferita una valutazione di 5 cuori su 5. Infine, il tasto "Reset filtri" permette di resettare tutte le ricerche precedentemente descritte, reimpostando la lista completa.

  Sotto a questi pulsanti, una volta selezionato un media, vengono visualizzate tutte le informazioni relative ad esse. Ogni media mostra quindi tutti i suoi campi dati inizializzati nel momento dell'inserimento nella lista (o dopo essere stati modificati)
- il terzo layout è collocato sotto ai due precedenti e presenta le funzionalità di inserimento, modifica e rimozione. Sono inoltre presenti i pulsanti per ordinare la lista secondo l'ordine alfabetico e per mostrare "l'intrattenimento totale" che fa affidamento sul metodo virtuale **`durataTOT()`** e attraverso le chiamate polimorfe somma la durata di tutti i film, la durata di tutta la musica e il tempo necessario per la lettura di tutti i libri (3 minuti per pagina, quantità considerata generalmente ragionevole).

Container

Il contenitore di oggetti che viene utilizzato dal modello è stato implementato tramite una double linked list (lista doppiamente linkata). Esso permette le operazioni di inserimento e rimozione di elementi in qualsiasi posizione in maniera efficiente, ovvero consentendo tali operazioni in tempo O(1) costante.

Estensibilità

Per la migliore chiarezza e manutenibilità del codice, si è deciso di separare il più possibile tutte le varie sezioni, sia nella parte del modello che nella parte della vista. Il progetto, visto il monte ore limitato, presenta una gerarchia alquanto basilare e per ciò questa gerarchia risulta facilmente sviluppabile in "verticale": Libro, Film e Musica potrebbero a loro volta avere delle classi derivate come ad esempio E-book e Cartaceo per Libro. L'implementazione risulterebbe semplice: sarebbe sufficiente inserire i nuovi file relativi alle nuove classi, modificare parzialmente il file xml per lettura e scrittura, e

implementare le nuove funzionalità. Le modifiche al codice preesistente sarebbero dunque molto limitate.

Caricamento e salvataggio del contenitore

Il caricamento e il salvataggio del contenitore avvengono attraverso l'utilizzo di un file XML e l'implementazione di alcuni metodi di lettura e scrittura nei file xml.h e xml.cpp.
Il caricamento del contenitore avviene in automatico quando l'applicazione viene lanciata, mentre per accedere alla funzione di salvataggio sarà sufficiente fare click su File nel menù principale e conseguentemente salvare (verrà mostrata una finestra per informare dell'avvenuto salvataggio).
E' stato scelto il formato XML in quanto facilmente fruibile da tutti gli utenti, anche quelli meno esperti: i tag sono di facile interpretazione e rappresentano direttamente i campi dati della gerarchia.
In scrittura, la funzione `void xml::write(Container<MediaItem *> & lista) const` verifica il risultato del downcasting sull'oggetto MediaItem e scrive i campi dati relativi alla sottoclasse in questione (libro, Film, Musica).
In lettura invece, `Container<MediaItem *> xml::read() const` tramite un ciclo while legge tutti i tag e li inserisce nei relativi campi.

Tempo impiegato

Circa 50 ore, di cui:

- Analisi preliminare del problema: 3 ore
- Progettazione modello e GUI: 6 ore
- Apprendimento libreria Qt e partecipazione al tutorato: 15 ore
- Codifica modello: 7 ore
- Codifica GUI: 14 ore
- Debugging: 3 ore
- Testing: 2 ore

Note

Il progetto prevede l'utilizzo di un file .pro diverso da quello ottenibile con l'esecuzione del comando qmake-project, per la presenza di alcuni flag relativi alla versione c++11, come ad esempio l'istruzione `QMAKE_CXXFLAGS += -std=c++11`. In particolare, nel

progetto sono stati utilizzati alcune keywords come *default* e *override*, specifiche di c++11.

Utilizzare i comandi **`qmake; make;`**

E' richiesto inoltre che la cartella in cui risiedono tutti i file e tutto il codice si chiami **ProgettoOOP** (nomenclatura che aveva durante lo sviluppo, abbreviazione di Progetto Object Oriented Programming) in quanto sia lettura e scrittura attraverso file xml, che fogli di stile e risorse (con immagini annesse), fanno riferimento a una cartella denominata in quel modo.

L'intero progetto è stato sviluppato con sistema operativo Windows 10, Libreria Qt in versione 5.9.5, compilatore MinGW 5.3.0 32 bit ed editor Qt Creator in versione 4.9.2 (Community)