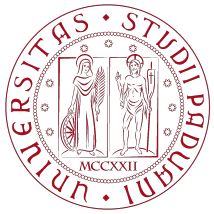


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova

Anno Accademico 2024/2025

Progetto di Programmazione ad oggetti

BibQT

Autore:
Michele Dioli 2077629

GitHub Repository:
`https://github.com/MicheleDioli/UniPd-PaO-2025`

1 Introduzione

BibQT è un'interfaccia grafica che permette la gestione di tre diversi tipi di articoli comunemente associati a una libreria. La gestione include operazioni come l'inserimento, la modifica, il salvataggio e la cancellazione. È possibile gestire i seguenti articoli: film, libri e riviste, ognuno dei quali presenta caratteristiche specifiche in base al proprio formato. In questa applicazione è stata prestata particolare attenzione all'interfaccia grafica, cercando di renderla il più intuitiva e comprensibile possibile.

2 Descrizione del modello

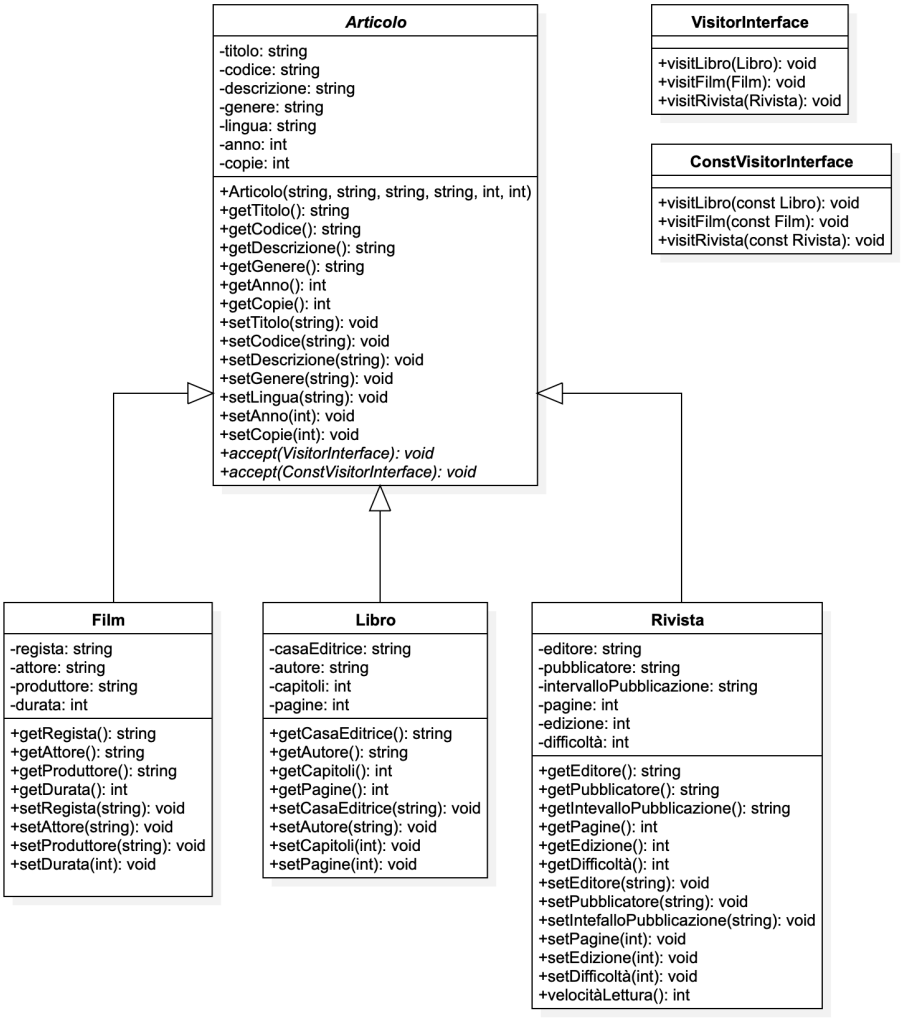
Il modello logico si occupa della rappresentazione dei possibili articoli inseribili. Viene definito il design pattern Visitor, implementato come VisitorInterface e nella sua versione costante ConstVisitorInterface, introdotti per un'ottimale comunicazione tra il modello logico e quello grafico.

La gerarchia ha come classe astratta Articolo, che rappresenta un generico articolo senza particolari caratteristiche. Ogni articolo ha un codice univoco, che può essere alfanumerico, e un titolo, che insieme a genere, lingua e anno crea un vincolo di chiave nel modello implementato. Altri attributi della classe Articolo sono autore, descrizione e copie.

L'applicazione gestisce tre diverse classi che ereditano da Articolo:

- **Film**: classe implementata per rappresentare un film tramite attributi distintivi quali regista, attore (principale), produttore e durata.
- **Libro**: classe che contiene attributi comuni nella rappresentazione di un libro, quali autore, casa editrice, pagine e capitoli.
- **Rivista**: la classe Rivista rappresenta una generica rivista e cerca di distinguerla il più possibile. A tale scopo, sono stati utilizzati gli attributi editore, pubblicatore, intervallo di pubblicazione, pagine, edizione e difficoltà.

L'attributo difficoltà è completamente arbitrario e viene utilizzato per stimare la velocità di lettura, oltre che per evidenziare una differenza rispetto alla classe Libro.



3 Polimorfismo

L'utilizzo principale del polimorfismo è dovuto al design pattern Visitor. Sono state costruite due classi astratte per implementare tale design pattern, necessarie per gestire anche oggetti costanti. Una volta scritta la classe astratta VisitorInterface, è possibile implementarla in più classi e sfruttarla in molte occasioni. L'implementazione di questo pattern nel progetto è stata adottata ogni volta che si è presentato un utilizzo pratico.

La classe astratta è stata implementata dalle seguenti classi concrete: JsonVisitor, MostraVisitor e ListaVisitor.

- `JsonVisitor`: utilizzato per la creazione del file JSON per il salvataggio.

- `MostraVisitor`: utilizzato per la creazione della finestra di visualizzazione di un singolo articolo.
- `ListaVisitor`: utilizzato per la creazione del singolo elemento presente nella finestra di visualizzazione della lista completa degli articoli.

4 Persistenza dei dati

La persistenza dei dati è implementata attraverso l'uso di un formato strutturato JSON, sfruttando la libreria Qt JsonObject per una gestione più veloce e semplice. Nel programma è possibile salvare direttamente l'intera lista di articoli tramite il pulsante "Salva" presente nella toolbar oppure utilizzando la scorciatoia da tastiera `Ctrl + S`, creando così un file JSON contenente tutte le informazioni sugli articoli creati. Ogni articolo può essere salvato singolarmente tramite il menu a comparsa, cliccando con il tasto destro del mouse sull'articolo desiderato. Inoltre, è stato implementato un sistema per importare file JSON e visualizzarli nella finestra principale. A tale scopo è stato preparato un file di esempio, `lista.json`, che può essere importato per avere subito a disposizione alcuni articoli.

5 Funzionalità implementate

Di seguito una lista delle principali funzionalità implementate:

- Inserimento di articoli tramite GUI
- Minimo algoritmo di parsing sull'inserimento
- Importazioni di lista e articoli singoli da file strutturato JSON
- Salvataggio di lista e articoli singoli in file strutturato JSON
- Modifica di articoli tramite GUI
- Eliminazione di articoli dalla lista visualizzata
- Menù a comparsa per gestire singolarmente un articolo
- Sistema di ricerca parziale
- Sistema di filtri per diversa visualizzazione della lista
- Possibilità di scorrere verticalmente la lista
- Scorciatoie da tastiera
- Utilizzo di stile qss inline
- Toolbar statica in alto

- Minima status bar
- Utilizzo di pulsanti nella visualizzazione degli oggetti

5.1 Compilazione aiutata

La compilazione è facilitata dalla presenza di uno script `make.sh` nella cartella `src`. Lo script è stato testato su Debian 12 e macOS Sequoia 15.3.2.

6 Rendicontazione ore

Il principale motivo dello sforamento delle ore è stato dovuto ai numerosi tentativi di implementare il menu a comparsa tramite clic destro, una feature inizialmente sottovalutata. Un altro imprevisto è stato la gestione dei file importati che, se modificati, creavano un nuovo file JSON invece di aggiornare quello esistente. Inoltre, il debugging per la risoluzione dei segmentation fault a runtime si è rivelato un incarico più complesso del previsto. Uno strumento utile per affrontare questo problema è stato il debugger lldb, che ha facilitato l'individuazione dei metodi problematici.

Attività	**Ore previste**	**Ore effettive**
Studio e progettazione	10	10
Sviluppo del codice del modello	10	12
Studio del framework Qt	8	8
Sviluppo del codice della GUI	4	8
Test e debug	5	7
Stesura della relazione	3	3
Totale	**40**	**48**

7 Possibile warning

L'uso di `globalPos()` è attualmente deprecato. Se si verificano problemi, modificare il file `cliccabile.cpp` situato in:

`src/view/sensoriqt/cliccabile.cpp`

Alla riga 8, individuare la seguente riga:

```
MostraMenu(event->globalPos());
```

E sostituirla con:

```
// MostraMenu(event->globalPos());  // Versione deprecata
MostraMenu(event->globalPosition().toPoint());  // Versione aggiornata
```