UNIVERSITÀ DI PISA

A.A. 2019/2020

Large-Scale and Multi-Structured Databases

Task 0

Davide Coccomini - Luigi Gjoni
Marilisa Lippini - Lorenzo Nelli

Indice

1 Introduction

The application is a Java program for the management of a book shop. In this situation, the scope is to improve the efficiency of the workers but also grant a complete and efficient track of the books that are stored in the book shop.

For every book the following information are needed: title, author, publisher, price, number of pages, category, publication year and quantity. The application also maintains some information regarding the author (first name, last name and a brief biography) and the publisher (name and location) in order to allow the salesman to query books by author and/or publisher. Each entity is uniquely identified by an id automatically assigned by the system.

2 Specifications

## 2.1 Actors and functionalities

The application is mainly thought to be used by the workers of the book shop to manage the books flow. They will be able to:

- Read the list of the books in the book shop;

- Read the list of the authors that wrote the books in the book shop;

- Read the list of the publishers that published the books in the book shop;

- Insert a new book;

- Sell a stored book to a customer;

- Update the quantity of a book;

- Remove a book from the book shop.

2.2 Use cases

1. IF user clicks the button "Books" on the top of the page:

   (a) The system select all the books from the table BOOK;

   (b) The table BOOK, the form to insert a new book and the buttons for editing a book are shown;

2. IF user clicks the button "Authors" on the top of the page:

   (a) The system select all the books from the table AUTHOR;

   (b) The table AUTHOR is shown;

3. IF user clicks the button "Publishers" on the top of the page:

   (a) The system select all the books from the table Publisher;

   (b) The table PUBLISHER is shown;

4. IF user compiles the form to insert a new book and press the button to submit:

   (a) The system adds the new book to the table BOOK;

   (b) The list of books showed to the user is updated.

5. IF user presses on the button "-" on a book's row:

   (a) The system decreases the quantity of the book in the book shop;

   (b) The table BOOK is updated considering the new quantity of the book.

6. IF user presses on the button ”+” on a book’s row:

   (a) The system increments the quantity of the book in the book shop;

   (b) The table BOOK is updated considering the new quantity of the book.

7. IF user compiles the text field on a book’s row, inserts a value and leaves the focus on the object:

   (a) The system update the quantity value for the book;

   (b) The table BOOK is updated considering the new quantity of the book.
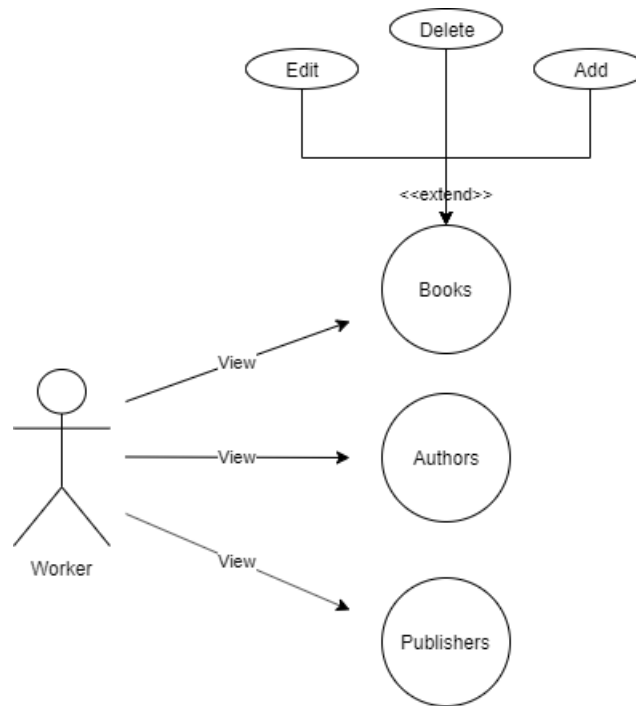
3 Data storage and flow

3.1 Implementation

The database was designed using draw.io and realized using MySQL Workbench and PHPMyAdmin.The tables have been filled using real books, publishers and authors from the international literature. The connection to the database is managed by the class ConnectionManager, that contains the credentials for the connection and a function worker() that executes the queries on the database. This function is differentiated to work both with or without query parameters, compiling the query accordingly to the given arguments. The possible queries are contained in the class DatabaseManager, that receives the commands from the interface and sends the query to be executed to the ConnectionManager.

3.2 ER-Diagram
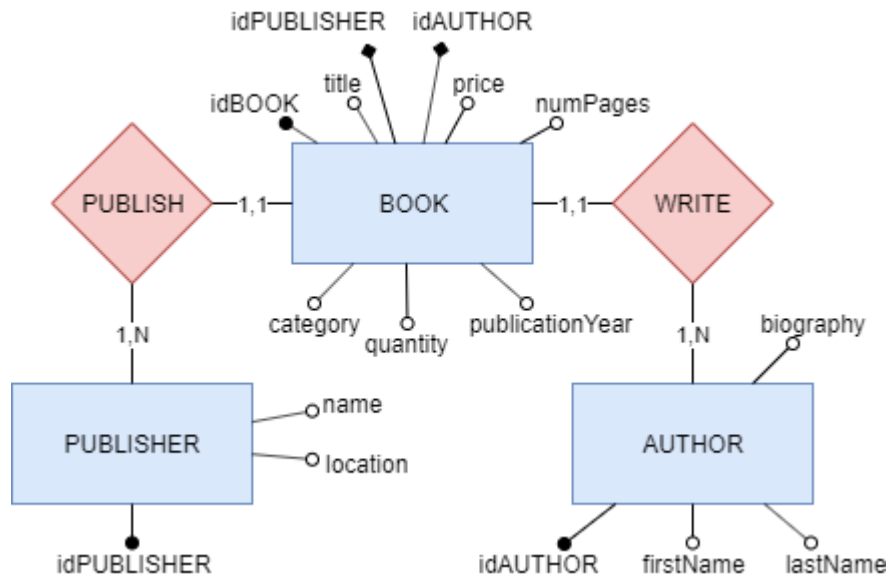
4 Software Architecture

4.1 User Interface

The UI is built using JavaFX, a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms. When the application starts, the worker will see an empty table and three buttons. Clicking on one of them, a list of books, authors or publishers is shown. The interface's heart is the table that shows to the user all the information about books, authors and publishers. The majority of the features have been implemented for the books table. Indeed, in the books section, there are buttons to decrease, increase and edit the quantity of a book (useful for the workers when a book is replenished/returned or a stored one is sold to a customer). There is also a form that can be compiled by the user to add a new book to the bookshop.

Interface showing books list:

The table columns can be used to reorder the elements (in this case, accordingly to quantity decreasing):

Book Title	Book Price	Book Pages	Publication Date	Book Category	Copies In Stock ▼
pperfield	3.95	768		Biography	3
ne Comedy	14.29	928		Classics	1
Thousand Leagues Under the Seas	13.43	518		Science fiction	0
e Earth to the Moon	11.95	338		Science fiction	0
and Juliet	4.79	336		Tragedy	0
	5.76	342		Tragedy	0
Peace	15.71	928		Historical	0
nd Punishment	14.29	565		Psychological	0
e in Stone: A Novel	12.1	322		Historical	0
erables	17.93	1264		Historical	0
	20.0	496		Tragedy	0
on In The Trees	11.64	288		Fiction	0
	12.73	384		Romance	0
Furioso	17.95	656		Classics	0
ce	5.69	88		Political treatise	0
n: The Tigers of Mompracem	16.95	272		Adventure	0
nt of Carmagnola and Adelchis	11.55	360		Tragedy	0
othed	18.74	275		Classics	0

Task0 Books Authors Publishers

Title
Pages
Quantity
Category
Price
Publication Year (YYYY)
Author ID
Publisher ID
Add Book

4.1 User Interface

The buttons can be used to increase/decrease the quantity or delete a row. The textfield can be used to update the quantity of a book by typing the value:

And leave the focus from the textfield:

The form can be used to add a new book:

Pressing the button the form is resetted and the table is updated with the new book:

Interface showing authors list:

Interface showing publishers list:

Publisher Name	Publisher Location	
Hachette Livre	Paris [FR]	
HarperCollins	New York [US]	
Macmillan Publishers	London [UK]	
Simon & Schuster	New York [US]	
Springer	Berlin [GER]	
Bonnier Books	Stockholm [SWE]	

Task0

Books   Authors   Publishers

4.2   Java Structure

Frontend

- Task0Controller: This class manages the interface accordingly with the actions of the user. It contains functions fill/update/initialize the table, submit forms and react to button clicks;

- EditButton: This class is an extension of the class "Button" and it is used to add functions to get the row's id and the arguments according to the table's row which the button is associated to;

- MainApp: This class contains the stage shown to the user with all the elements that are manipulated by the controller.

Middleware:

- Book: Bean class for the database table "Book". It contains all the fields and the methods to get/set them in order to be stored for a book instance;

- Author: Bean class for the database table "Author". It contains all the fields and the methods to get/set them in order to be stored for a author instance;

- Publisher: Bean class for the database table "Publisher". It contains all the fields and the methods to get/set them in order to be stored for a publisher instance;

- Util: Utility class that contains useful functions to interact with bean classes.

Backend:

- DatabaseManager: This class stores all the queries and contains the functions to interpret the commands received and translate them to queries. It also extract the results from the received ResultSet transforming it into a more readable format;

- ConnectionManager: It contains the database credentials and manage the connection to it. It also receives the queries from the DatabaseManager and executes them to return the ResultSet.

4.3   UML Diagram
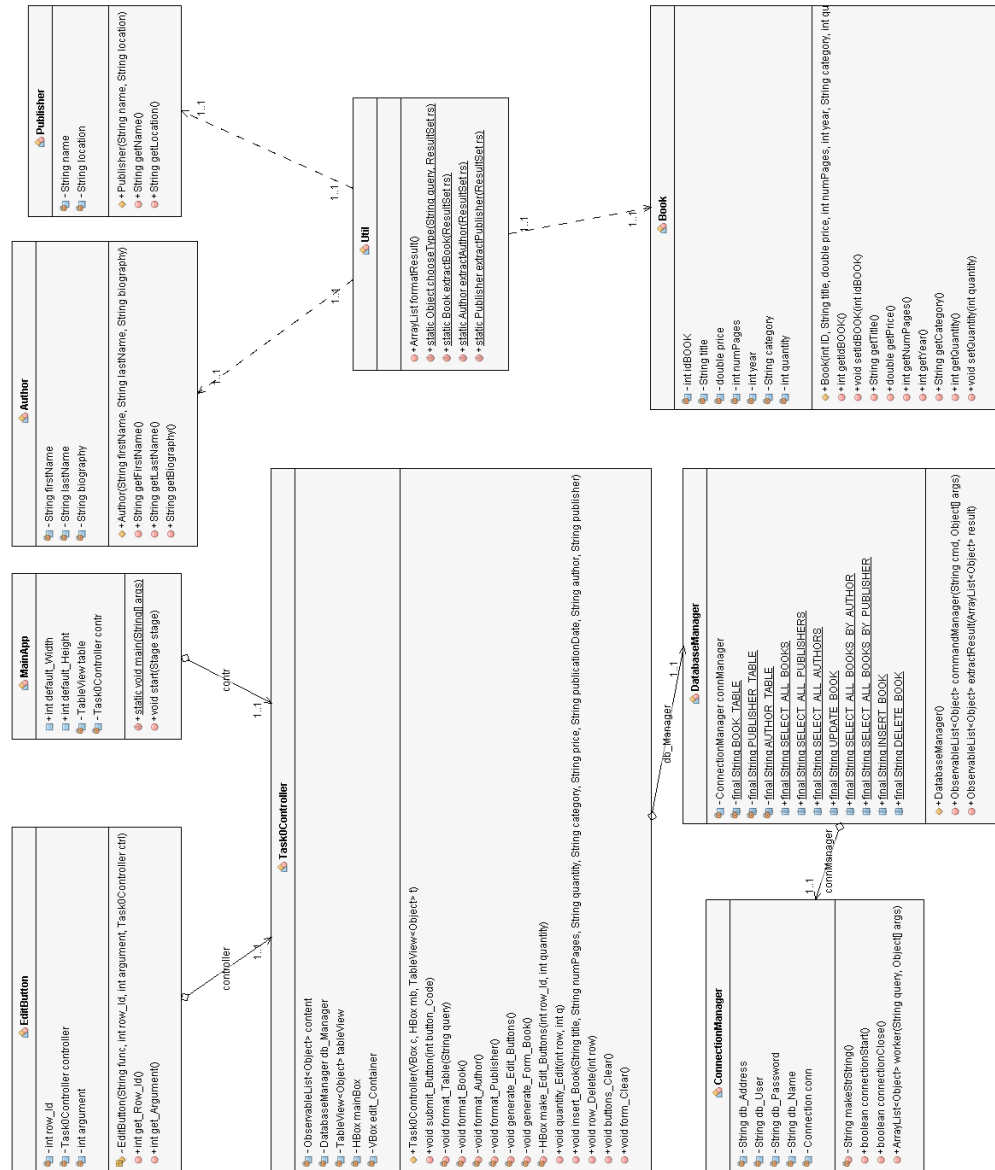
5   Testing

The first part of the test was carried out in the same way for all the functions: testing and verifying the correct functionality first through the interface, then via PHPMyAdmin. The rest of the testing phase was the debugging part related to various errors mainly due to JavaFX and FXML, but not to features, such as version compatibility, dependencies to optimize and non-functioning tags.

5.1   Special cases

In particular two special cases were tested:

- The user tries to decrease the quantity of a book with quantity zero. The system doesn't change the quantity;

- The user adds a book without inserting all needed attributes: the database handles this case automatically thanks to the NOT NULL attribute;

5.2 Workflow

The following scheme summarize the applied workflow highlighting the test phases:
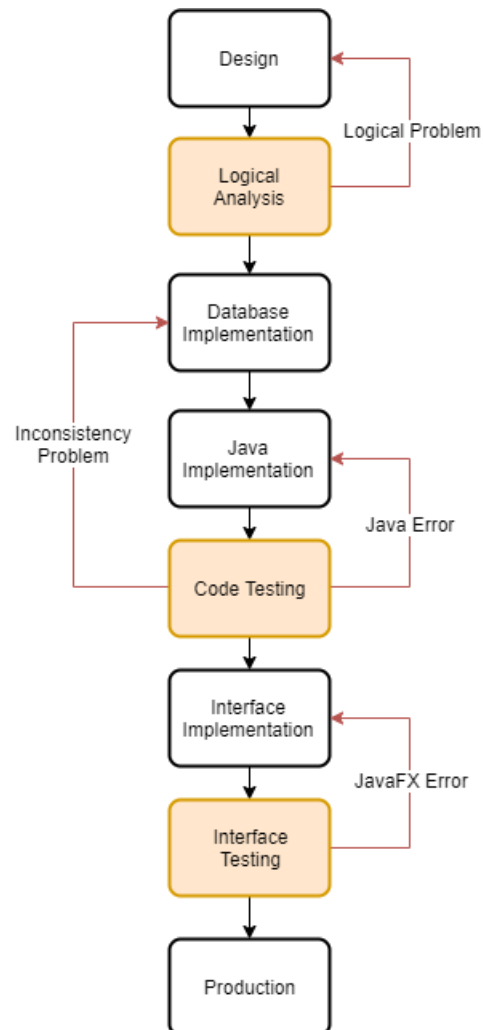
6 Conclusions

The main goal of the program have been achieved: using it, the bookshop will be able to increase its efficiency obtaining the total control to the book's flow. This software is already usable by a bookshop but it is also suitable for expansion with a lot of useful features in the future tasks.