

Assignment 2

Analysis and Design Document

Student: Matei Cristina-Bianca
Group: 30233

Table of Contents

1. Requirements Analysis	3
1.1 Assignment Specification	3
1.2 Functional Requirements	3
1.3 Non-functional Requirements	3
2. Use-Case Model	3
3. System Architectural Design	3
4. UML Sequence Diagrams	6
5. Class Design	6
6. Data Model	7
7. System Testing	8
8. Bibliography	13

1. Requirements Analysis

1.1 Assignment Specification

The assignment requires the implementation of an application for the employees of a book store. The application should have two types of users (a regular user represented by the book store employee and an administrator user) which have to provide a username and a password in order to use the application.

1.2 Functional Requirements

The system should provide a user interface in order to be used by the users. These should have the possibility to choose from logging in as a regular employee or as an administrator. After logging in, they should choose from various operations.

The regular user can perform the following operations:

- Search books by genre, title, author
- Sell books

The administrator can choose from the following:

- Add/Delete/View/Update books' information
- Add/Delete/View/Update regular users' information.
- Generate two types of reports files, one in pdf format and one in csv format, with the books out of stock.

The users should provide the corresponding inputs for each operation, especially when they have to provide some integers or reals for the quantity or the price of the books. The system should provide input dialogs for every attribute it needs to perform the operations. Also, the system should notify users if an illegal input has been provided, should perform the chosen operation, inform the users if the operation has finished with success or not and generate the outputs for the “view” operations.

1.3 Non-functional Requirements

The system should have some important properties like maintainability, reusability and portability. The maintainability of this application is generated by a good organization of classes and methods that provide the functionality of the application. The reusability and portability are provided by the implementation of this application in Java, a common programming language on all platforms. It should also store the data in multiple XML files, use the Model View Controller architectural pattern and use the Factory design pattern for generating the reports.

2. Use-Case Model

I will present the use-case description for selling books.

Use case: Sell books

Level: User-goal level

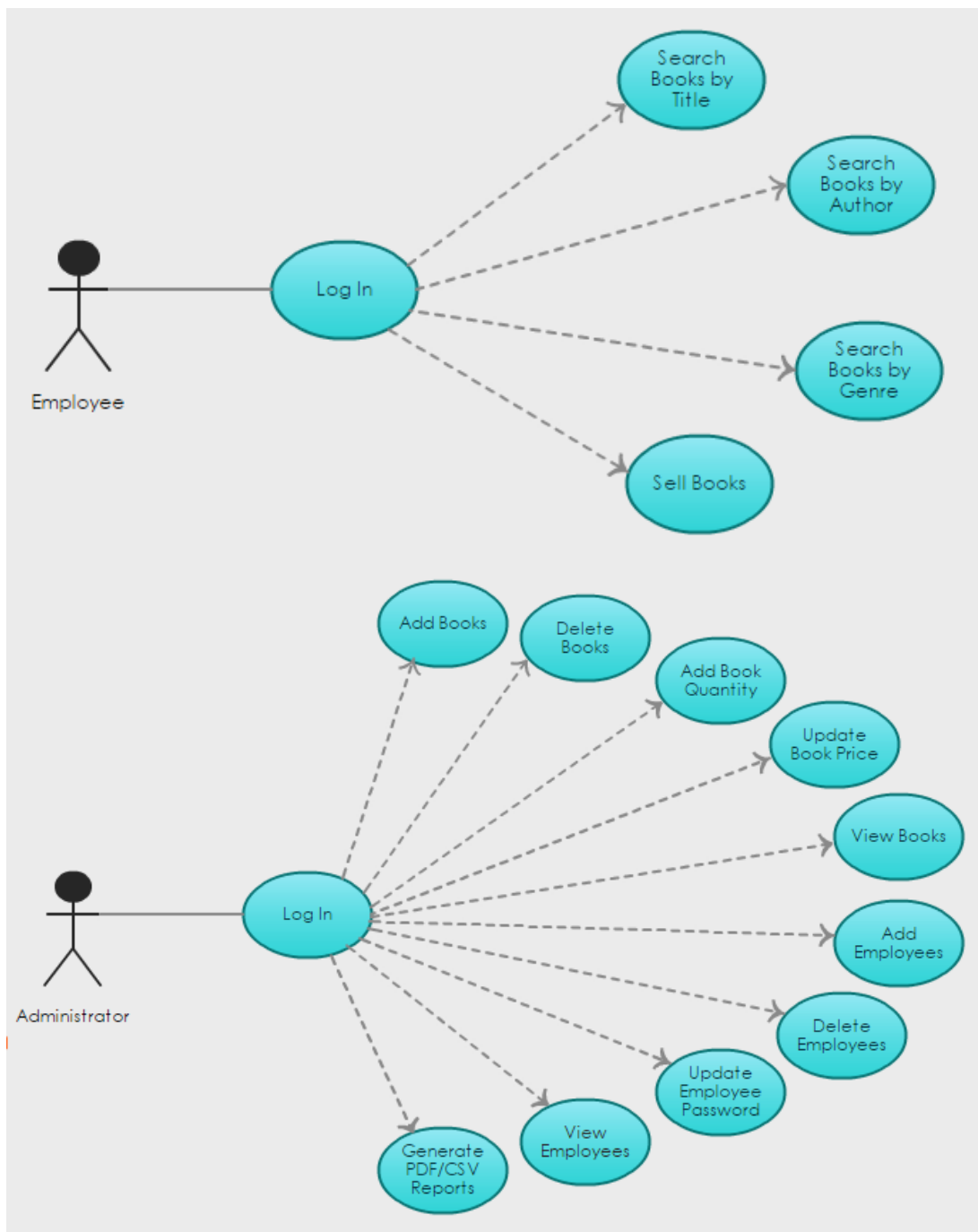
Primary actor: Regular user (employee)

Main success scenario: log in, press the “Sell books” button, introduce the title of the

book and the quantity, the book exists in the store in the specified quantity, update the quantity of the book, success message

Extensions: introduce the title of the book and the quantity, the book does not exist in the store or the quantity existing in the store is not enough, the selling can't be done, warning message

The use-case diagrams for the application include the actors Employee and Administrator and they are described below:



3. System Architectural Design

3.1 Architectural Pattern Description

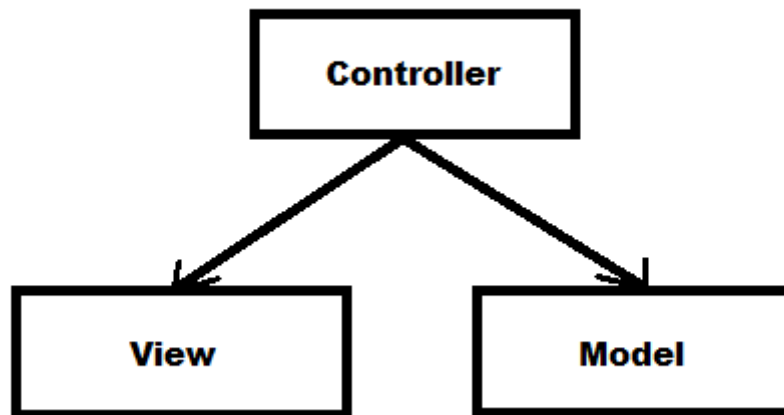
In this application I used the Model–View–Controller (MVC) pattern. This is a software architectural pattern that divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user.

The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic and rules of the application.

The view represents the user interface through which the system communicates with the user by input and output data. The controller accepts input and converts it to commands for the model or view.

3.2 Diagrams

The system's MVC architecture is described in the picture below.



In this application, the MVC architecture pattern is implemented as follows:

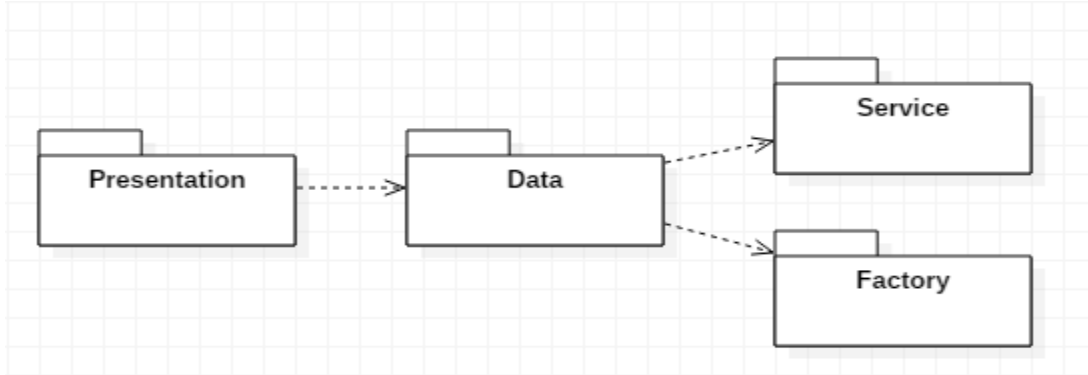
- The View component is represented by the View class that describes the user interface
- The Model component is represented by all the classes that describe the data used and the ones that provide data access from XML files
- The Controller is the class Controller that have a View object through which it gets input data and provides some operation by using objects described in the model classes

I have separated the classes of this application into four packages:

- Presentation package contains the classes View and Controller
- Data package contains the classes Book, Employee, Admin and the abstract class User. This classes represent the data used in this application.
- Service package contains the classes BookService, EmployeeService, AdminService that provide the data access to the XML files

- Factory package contains the classes PDFFile, CSVFile, PDFFileFactory, CSVFileFactory, Factory and the abstract classes File and FileFactory that implement the Factory design pattern

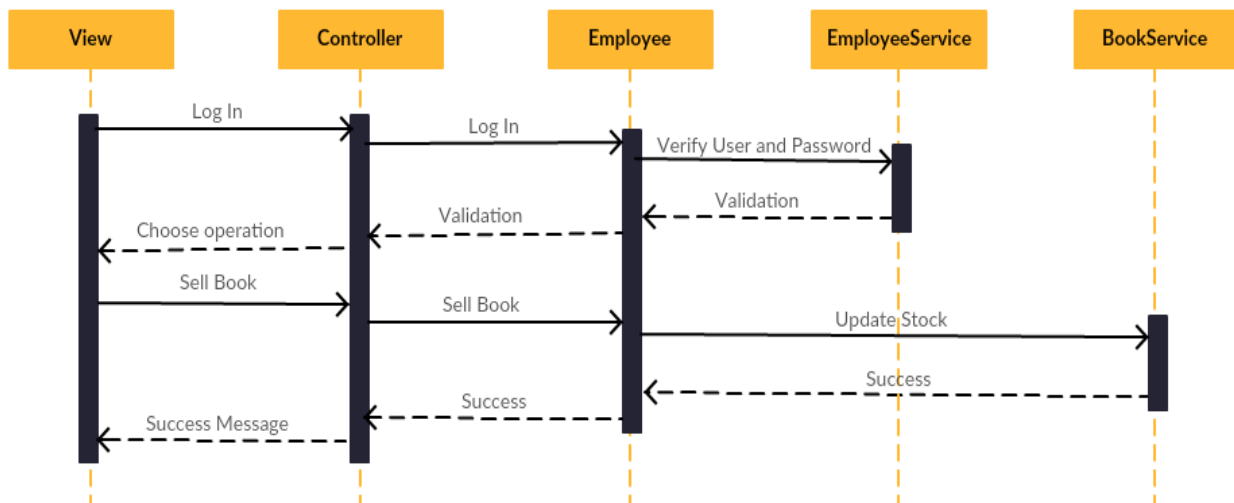
The package diagram of this application is described in the following picture.



4. UML Sequence Diagrams

A relevant scenario for this application is “Sell books” that can be performed by the employee only. The employee has to provide his username and password and then to provide the title and the quantity of the book to be sold.

The sequence diagram for this scenario is described by the picture below.



5. Class Design

5.1 Design Patterns Description

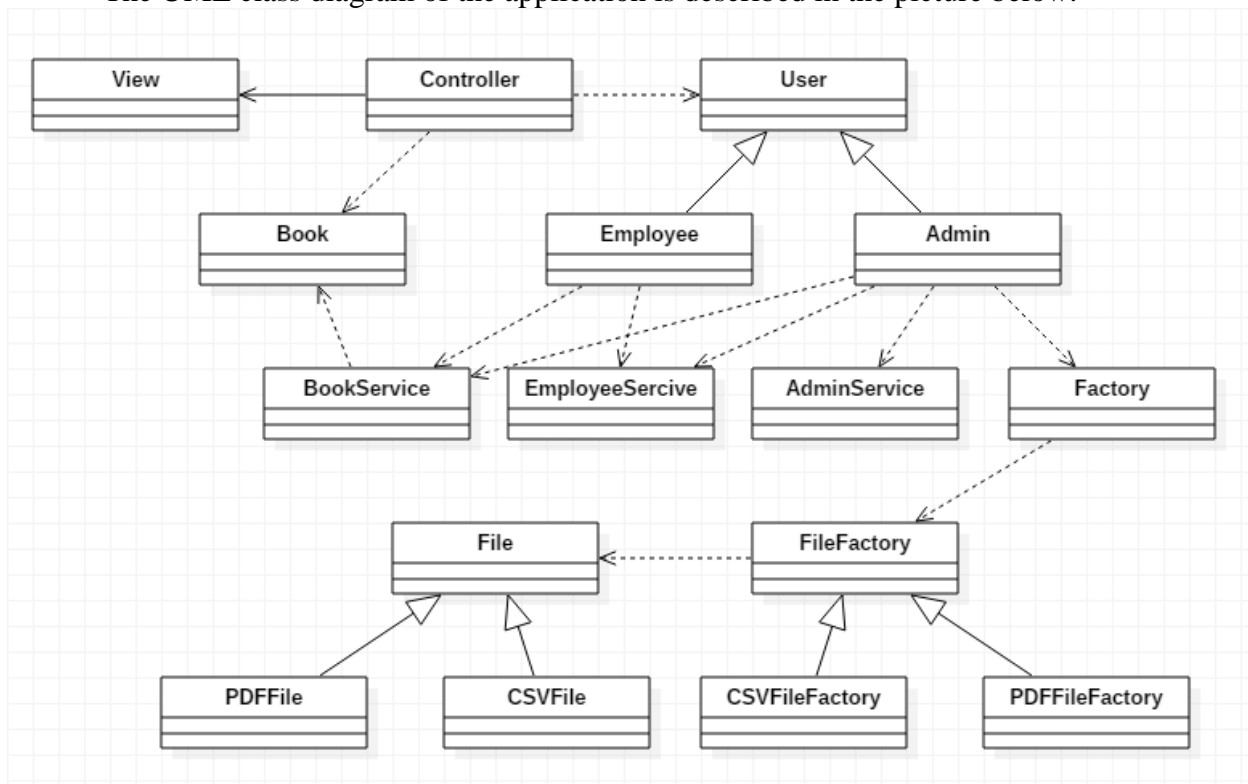
In this application I used the Factory design pattern. This is a creational pattern that help in object creation using indirect creation levels (subclasses or other objects). It defines an

interface for creating an object but let subclasses to decide which class to instantiate and it allows a class to defer instantiation process to subclasses. So, the Factory pattern provides an application-independent object with an application-specific object to which it can delegate the creation of other application-specific objects.

The Factory pattern is used in this application for creating the reports in the pdf and csv formats. In this way, I created an abstract class *File* that declares the abstract method *saveFile* that is implemented by the subclasses *PDFFile* and *CSVFile* in different ways so that it writes and saves a file in the specific format. The abstract class *FileFactory* declares the abstract method *buildFile* implemented in the subclasses *PDFFileFactory* and *CSVFileFactory* and it creates a specific file (*PDFFile* or *CSVFile*). Also, the class *Factory* implements the method *getFile* that creates a specific file using *PDFFileFactory* or *CSVFileFactory* based on a String that it has as a parameter.

5.2 UML Class Diagram

The UML class diagram of the application is described in the picture below.



6. Data Model

The data used in this application is stored in XML files. There are three XML files storing the information about the book, employees and administrators.

- The books.xml file contains information about the books: title, author, genre, quantity and price. A book node from the XML file looks like the description below:

```
<Book>
  <title>On war</title>
  <author>Carl von Clausewitz</author>
  <genre>War</genre>
  <quantity>0</quantity>
  <price>35.9</price>
</Book>
```

- The employees.xml file contains information about the employees: name, username and password. An employee node from the XML file looks like the description below:

```
<Employee>
  <name>Matei Cristina</name>
  <username>kitty_matei</username>
  <password>0000</password>
</Employee>
```

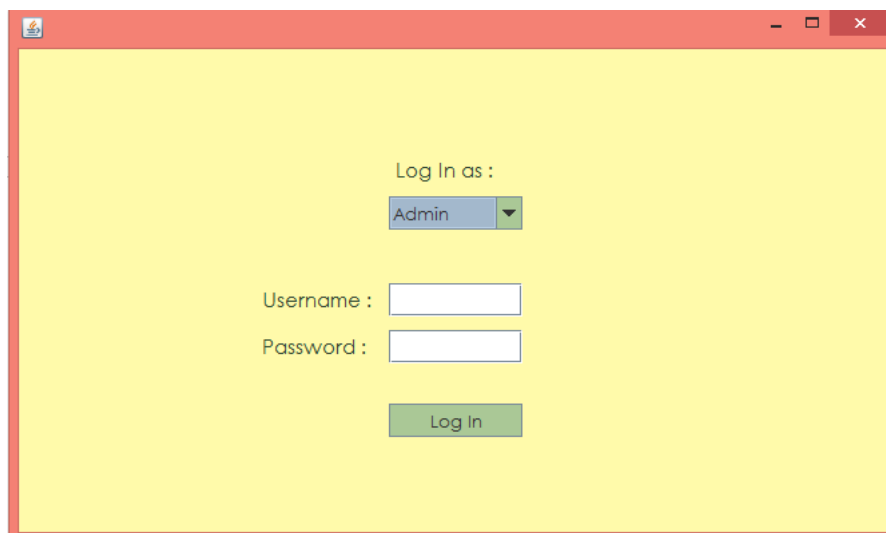
- The admins.xml file contains information about the administrators: username and password. An administrator node from the XML file looks like the description below:

```
<Admin>
  <username>admin1</username>
  <password>1111</password>
</Admin>
```

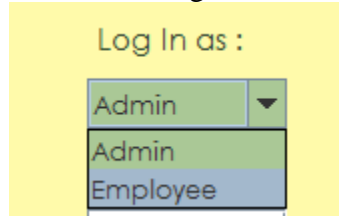
All the queries, insertions, deletes and updates on these XML files are made through XPath object and the changes made are then saved through Transformer objects.

7. System Testing

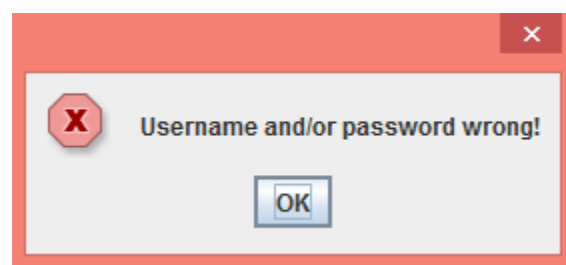
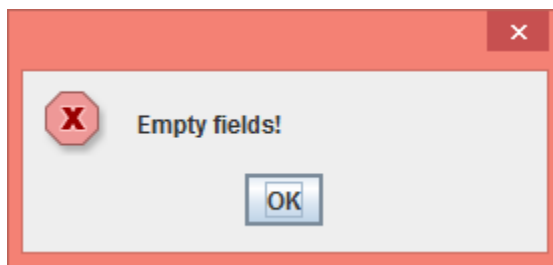
When we open the application it will look like in the following image.



You can choose from the combo box to log in as an admin or as an employee.



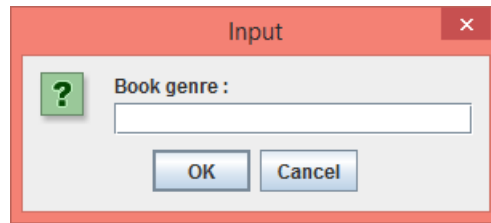
The admin that we will work with has the username “**admin1**” and the password “**1111**” and the employee has the username “**kitty_matei**” and the password “**0000**”. If you press the button before completing the fields or if the username or the password are not correct the will appear error messages like in the pictures below.



Firstly, we will log in as an employee and there will appear corresponding buttons for each operation that an employee can choose. The application will look like in the following image.



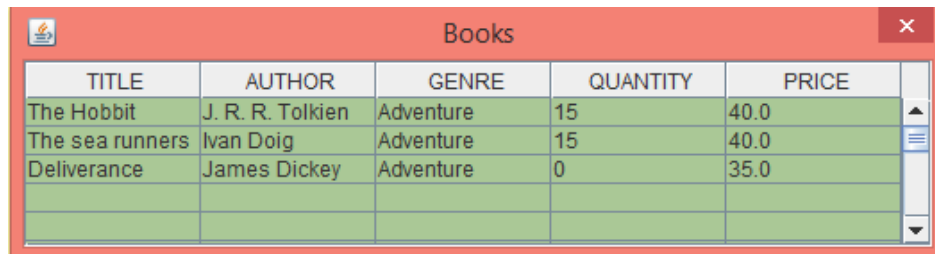
For the “search” operations there will appear dialog frames asking for book title/author/genre and then there will appear a table with the corresponding results of the search operation. For the “Search books by genre” operation if we introduce the book genre “Adventure” in the dialog frame the table will look like below.



Input

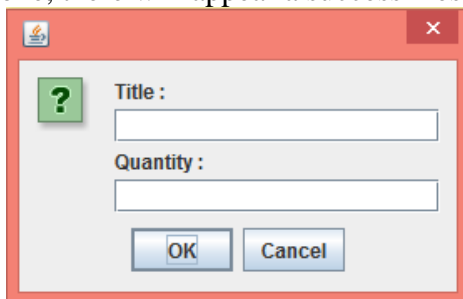
Book genre :

OK Cancel



TITLE	AUTHOR	GENRE	QUANTITY	PRICE
The Hobbit	J. R. R. Tolkien	Adventure	15	40.0
The sea runners	Ivan Doig	Adventure	15	40.0
Deliverance	James Dickey	Adventure	0	35.0

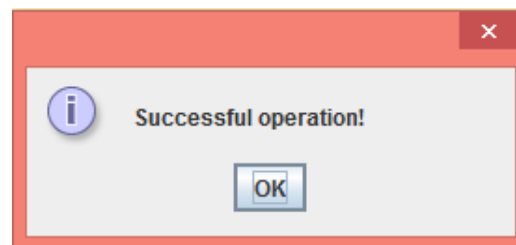
For the “Sell books” operations there will appear a dialog frame asking for the book title and quantity. If the quantity introduced is not an integer, is not greater than zero or if the book stock is not enough for the transaction there will appear error messages. In case the transaction can be done, there will appear a success message.



Title :

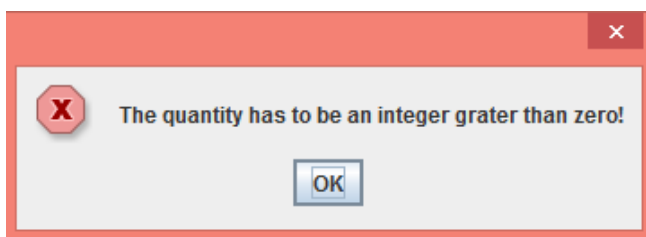
Quantity :

OK Cancel



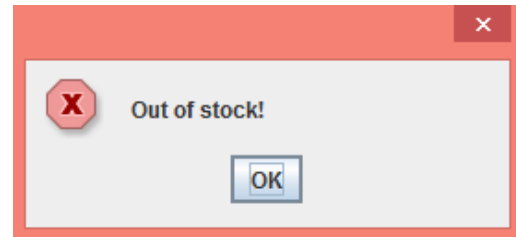
Successful operation!

OK



The quantity has to be an integer greater than zero!

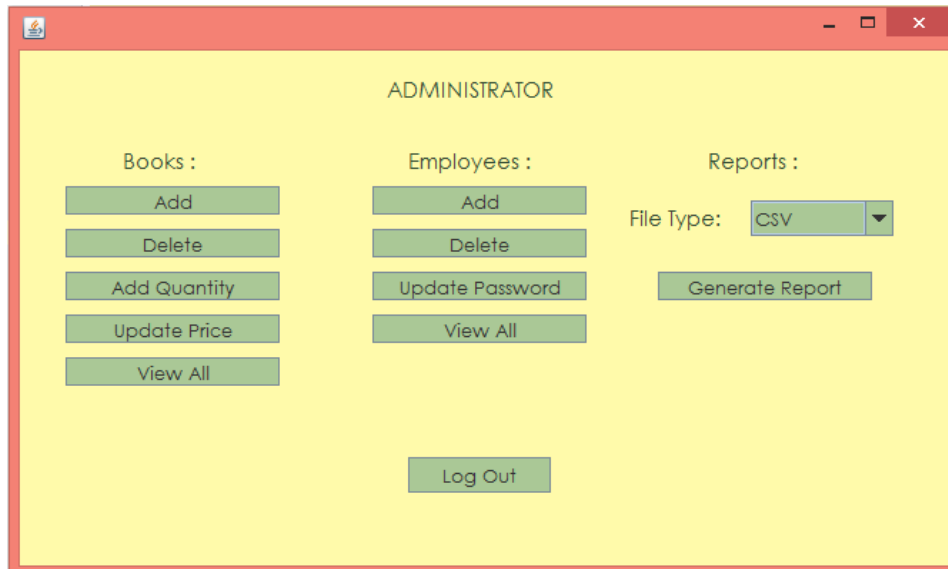
OK



Out of stock!

OK

The “Log Out” button takes us back at the beginning. Now, if we log in as an administrator there will appear corresponding buttons for each operation that an admin can choose. The application will look like in the following image.



The “View All” operations generate tables with all books and employees existing.

Employees		
NAME	USERNAME	PASSWORD
Matei Cristina	kitty_matei	0000
Radu Paul	radu_paul	0000
Matei Diana	dyamatei	0000

Books				
TITLE	AUTHOR	GENRE	QUANTITY	PRICE
On war	Carl von Clause...	War	0	35.9
The art of war	Sun Tzu	War	15	25.0
Maitreyi	Mircea Eliade	Romance	5	20.0
The Hobbit	J. R. R. Tolkien	Adventure	15	40.0
Evermore	Alyson Noel	Romance	24	40.0
Blue Moon	Alyson Noel	Romance	25	50.0
The sea runners	Ivan Doig	Adventure	15	40.0
Deliverance	James Dickey	Adventure	0	35.0

For the “Add” operations there will appear dialog frames asking for all the information about book and employees.

The image shows two separate dialog boxes. The left dialog box is titled with a green question mark icon and contains five text input fields labeled 'Title:', 'Author:', 'Genre:', 'Quantity:', and 'Price:'. At the bottom are 'OK' and 'Cancel' buttons. The right dialog box is also titled with a green question mark icon and contains three text input fields labeled 'Name:', 'Username:', and 'Password:'. It also has 'OK' and 'Cancel' buttons at the bottom.

If the quantity and price of the books are not integers/real greater than zero, if the title of the book already exists or if the username of the employee already exists there will appear error messages.

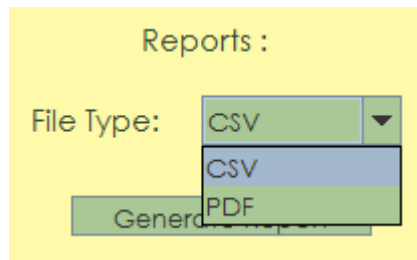
The image shows three error dialog boxes, each with a red 'X' icon. The top dialog box contains the message 'The quantity has to be an integer and the price has to be a real greater than zero!' and an 'OK' button. The bottom-left dialog box contains the message 'The username already exists!' and an 'OK' button. The bottom-right dialog box contains the message 'The book already exists!' and an 'OK' button.

For the “Update Price”, “Add Quantity” and “Update Password” operations we have to provide the title of the book and the new quantity/price or the name of the employee and the new password.

The image shows a single dialog box with a green question mark icon. It contains two text input fields labeled 'Name:' and 'New Password:'. At the bottom are 'OK' and 'Cancel' buttons.

For the “Delete” operations we have to provide the title of the book or the name of the employee we want to delete.

For the “Generate Report” operation we have to choose from the combo box the type of the file we want to generate, CSV or PDF. This operation generates a report in the specified format containing all the books out of stock.



8. Bibliography

<http://developers.itextpdf.com/content/itext-7-jump-start-tutorial/examples>
<http://howtodoinjava.com/3rd-party/parse-read-write-csv-files-opencsv-tutorial/>
<http://stackoverflow.com/questions/1431238/append-node-to-an-existing-xml-java>
<http://stackoverflow.com/questions/33759441/delete-a-node-and-its-elements-from-an-xml-file-in-java>
<http://viralpatel.net/blogs/java-xml-xpath-tutorial-parse-xml/>
<http://howtodoinjava.com/xml/how-to-work-with-xpaths-in-java-with-examples/>
<https://developer.ibm.com/urbandcode/docs/using-update-xml-xpath-step/>