Book Store

Student: Deac Dan

**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 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Design and implement an application for the employees of a book store using Java/C# API. 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.

# Functional Requirements

The regular user can perform the following operations:

* Search books by genre, title, author.
* Sell books.

The administrator can perform the following operations:

* CRUD on books (book information: title, author, genre, quantity, and price).
* CRUD on regular users’ information.
* Generate two types of reports files, one in pdf format and one in csv format, with the books out of stock.

# Non-functional Requirements

One of the non-functional requirements is the data security: in order to use the application the users must to provide their usernames and passwords. The portability and the reusability are sustained by the implementation of the application in Java, a well-known programming language. The maintainability is provided by the good organization of the classes, so that if we need to change a part of the program, only few parts will be affected.

2. Use-Case Model

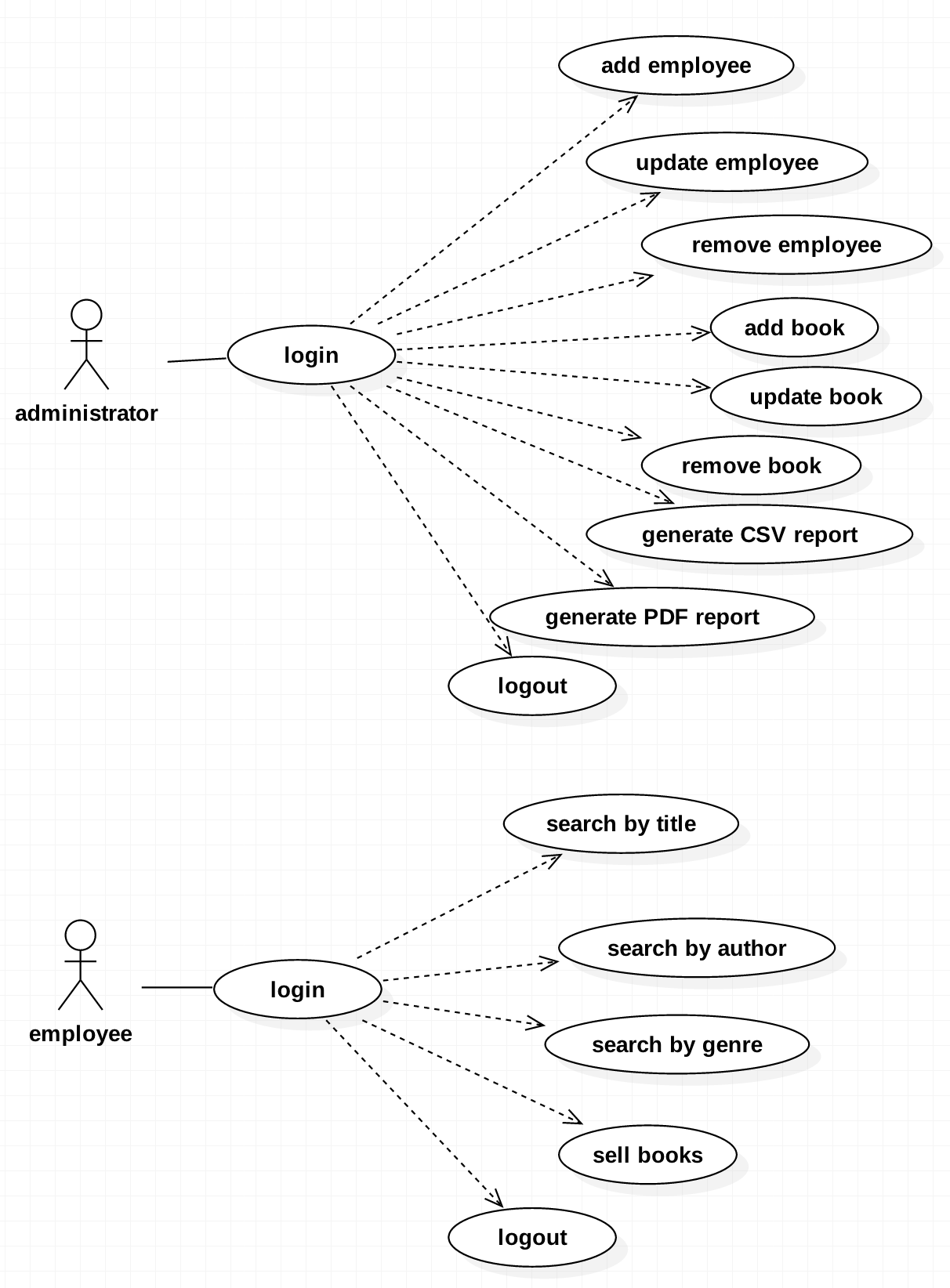
Use case: search books by title

Level: user-goal level

Primary actor: the employee

Main success scenario: login, type the name of the book, a table that will contain all the books with the given name will pop-up

Extensions: there are no books with the given title, a message dialog will be showed



3. System Architectural Design

**3.1 Architectural Pattern Description**

The architectural pattern that I've used is MVC (Model View Controller). The classes are split into three main modules: model part that contains the logic of the application, view part that contains the user interface and the controller part that ensures the communication between all the modules. This pattern provides a good separation of the data.

**3.2 Diagrams**

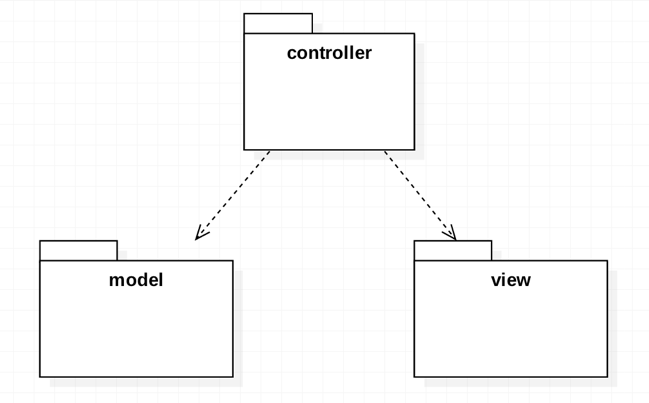
The MVC architectural pattern relationships are showed in the picture below:



The implementation of the MVC in my application is described below:

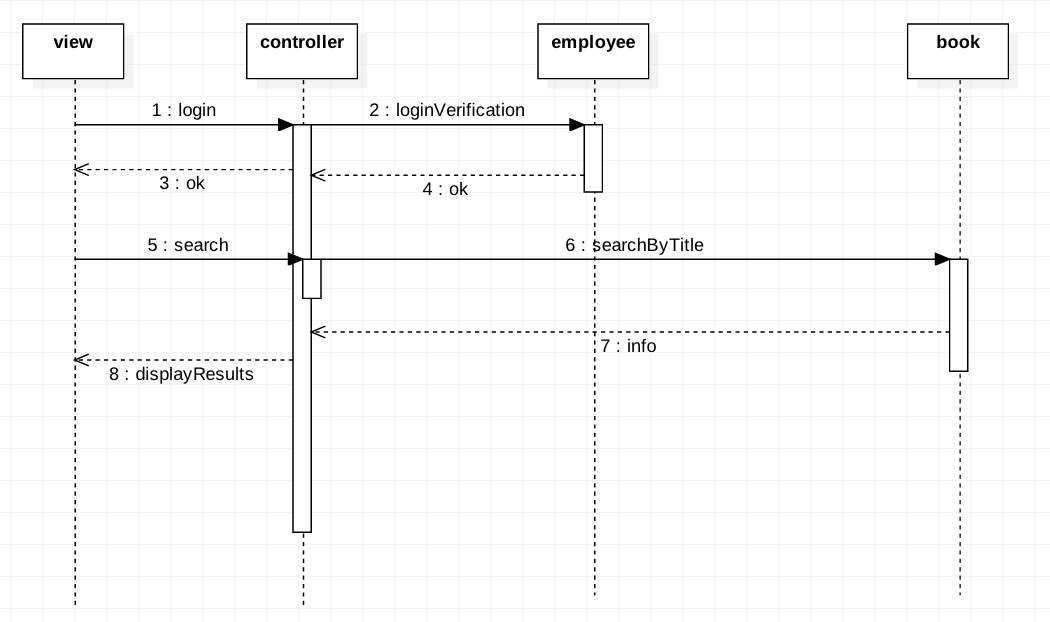
* I have created 3 packages, one for each module of the MVC pattern
* Model package contains the classes that describes the logic of the application, the classes responsible for reading/writing from/into xml files
* View package contains the classes which make up the user interface: a class for each frame (login page, admin page and employee page)
* Controller package contains three classes that are associated to the classes from the View package. Each class is responsible for controlling a frame

The package diagram is showed in the picture below:



4. UML Sequence Diagrams

The sequence diagram for the "search by title" scenario is showed in the picture below:

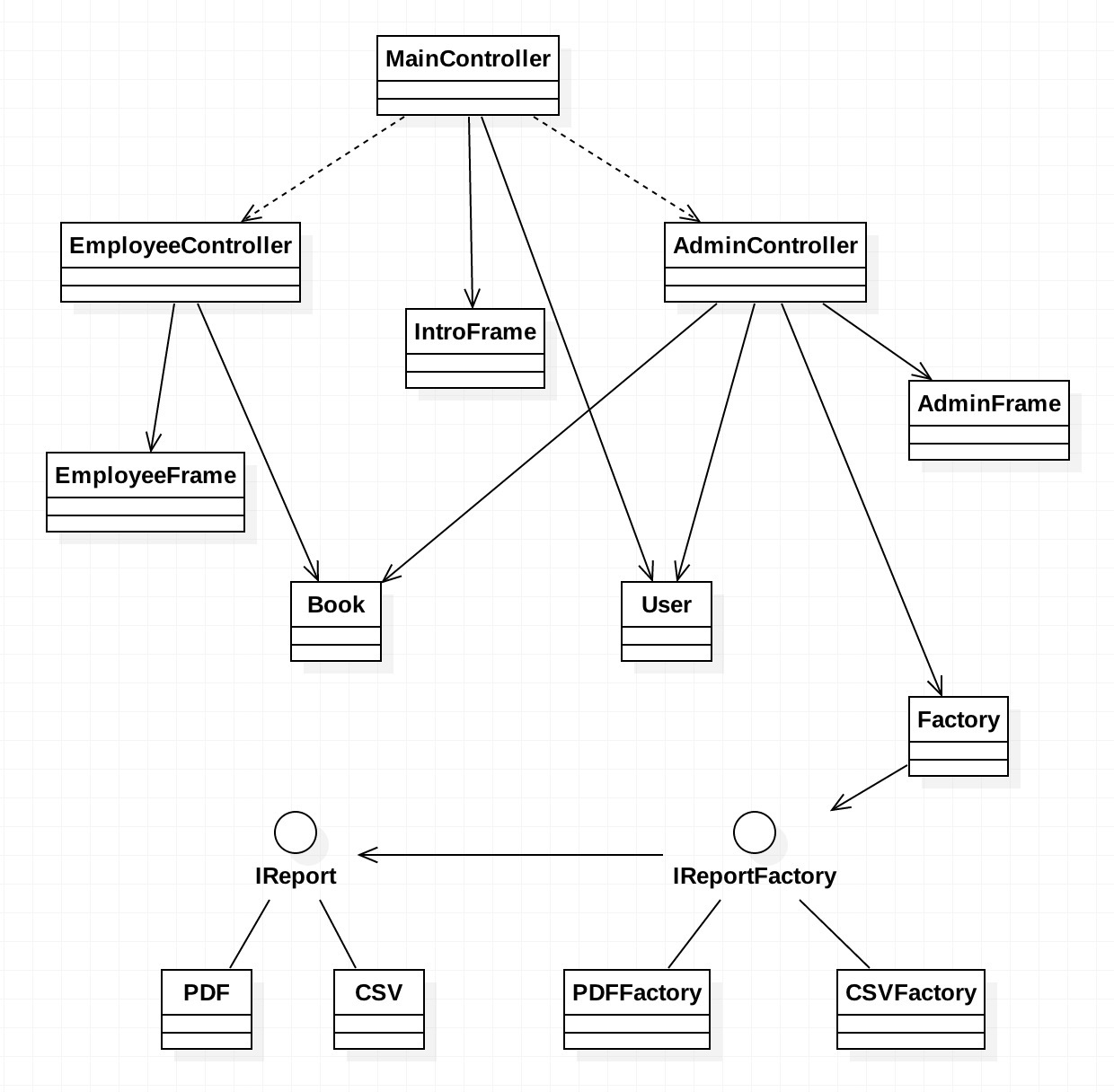


5. Class Design

**5.1 Design Patterns Description**

As a design pattern I have used Factory Pattern. Factory method pattern is a creational pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created. It defines an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses.

**5.2 UML Class Diagram**



6. Data Model

The persistency of the data is ensured by storing it in .xml files. I've used three xml files: one for books, one for users and one for the selling operations. The data are stored and read to/from xml files using XML DOM parser.

Book.xml stores information like title, author, genre, quantity and price:

<book>

<title>Fratii Jderi</title>

<author>Sadoveanu</author>

<genre>epic</genre>

<quantity>15</quantity>

<price>22</price>

</book>

User.xml stores information like username, password and type:

<user>

<username>admin</username>

<password>admin</password>

<type>admin</type>

</user>

Sell.xml stores information like title, author, genre, quantity, price and total price:

<sell>

<title>Fratii Jderi</title>

<author>Sadoveanu</author>

<genre>epic</genre>

<quantity>123</quantity>

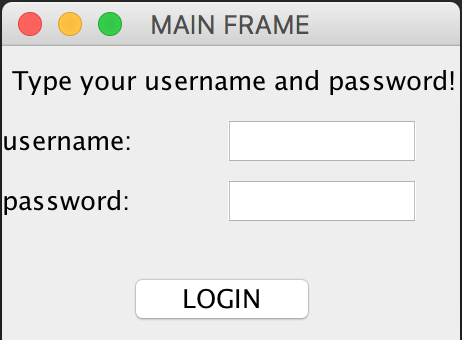
<price>22</price>

<totalPrice>2706</totalPrice>

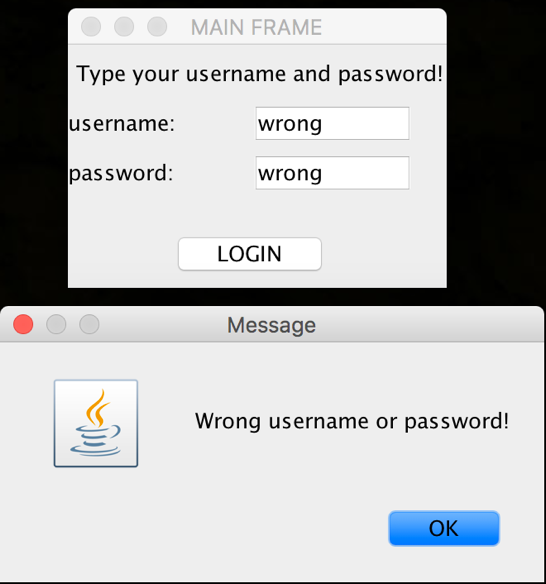
</sell>

7. System Testing

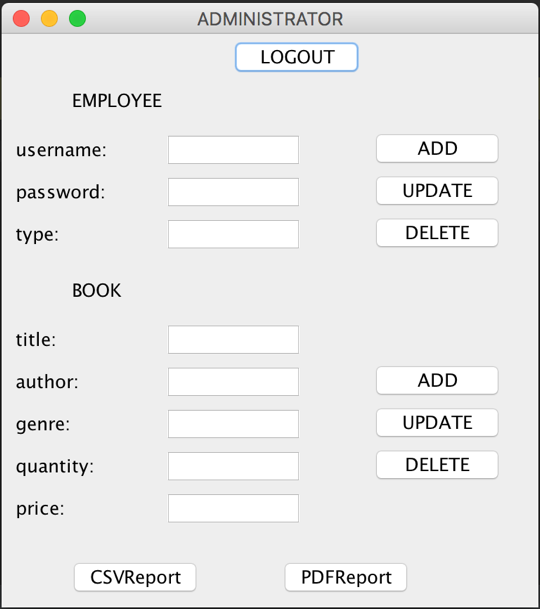
After we run the application the login page will pop-up:



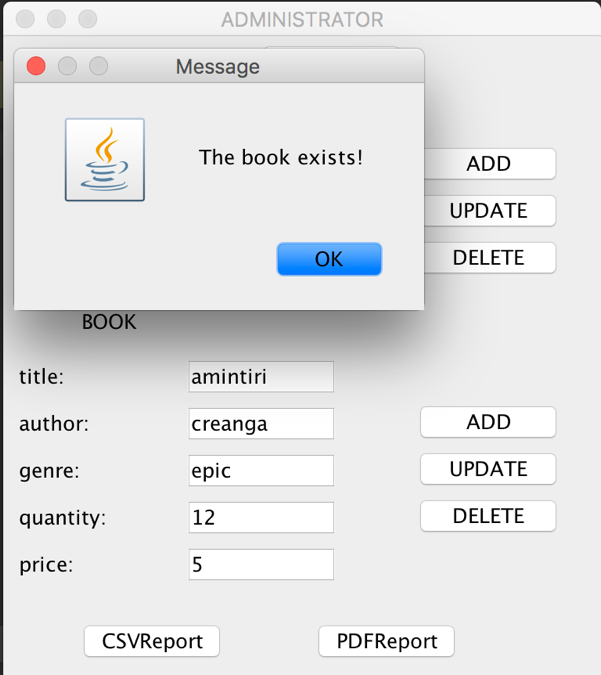
The user types his credentials and depending on his type the admin page or the employee page will pop-up. If the username or the password are wrong a message dialog will be showed.



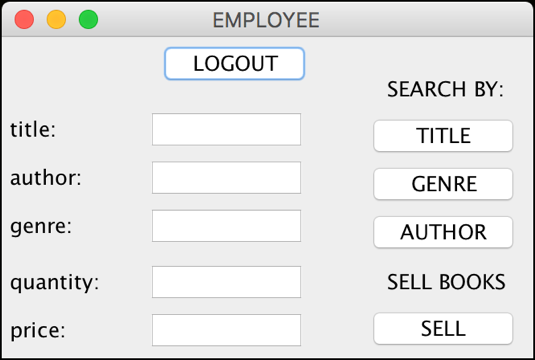
The administrator could do CRUD operations on employees and books and generate two types of reports.



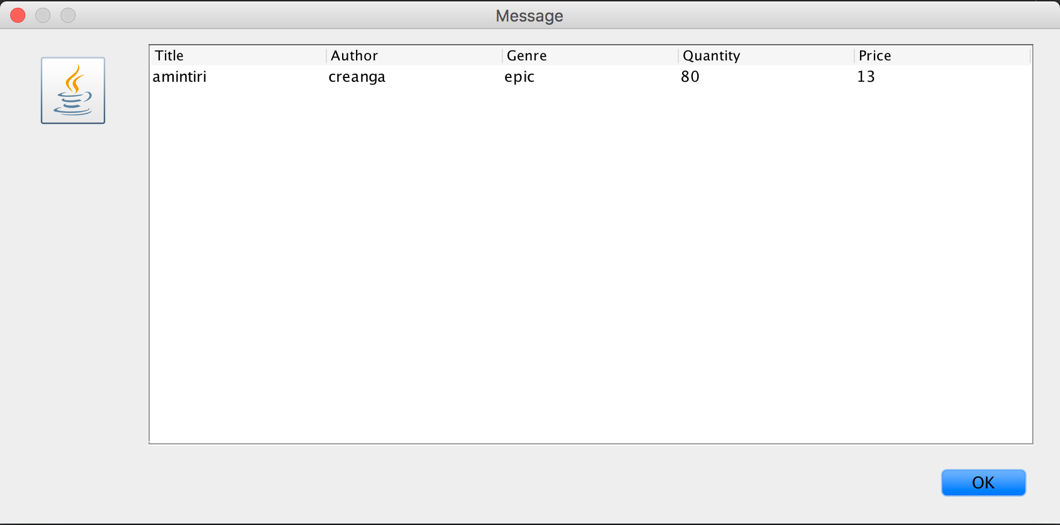
When adding a new book an error message will be showed if the administrator tries to add a book that already is in the xml file (same title and author). When adding an employee if there is an employee with that name in the xml file an error message will be showed as well.



For the employees the page will look like in the picture below, and they can perform searches by title, author and genre as well as selling books.



When searching a book a table will pop-up if there are books that are matching the search or a message dialog will be showed if there are no books that are entailing the search.



8. Bibliography

<https://www.w3schools.com/xml/xml_dom.asp>

<https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html>

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>

<https://sourcemaking.com/design_patterns/factory_method>

<http://stackoverflow.com/questions/14566596/loop-through-all-elements-in-xml-using-nodelist>

<https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilderFactory.html>

<https://www.tutorialspoint.com/java/xml/javax_xml_parsers_documentbuilderfactory_newinstance_classloader.htm>