ANALYSIS 3: OBJECT-ORIENTED MODELING (INFANL03-3 | INFANL23-3)

Educational Period 3 [2020-21]

Public Library System

This assignment consists of the design and implementation of a Public Library System (PLS) with which the information can be managed that is needed to run a public library, lending out paper books to customers. It goes without saying that the functionality required is only minimal and comes nowhere near a real PLS. Moreover, normally such a system would involve a database, but here you can limit yourself to storing data in RAM and in files. The system will consist of a backend (data processing and storage) and a frontend. The frontend can consist of a console based (textual) interface that gives access to all the functions of the system. The whole issue of security and access rights is left out of scope of this assignment.

The library has paper books, a catalog of books, a loan administration, customers, and a librarian (bibliothecaris). New customers can be added to the administration. New books (that means: the identification details of paper book items) can be added to the catalog. Customers can search for books and loan books. The library can be filled with books and customers from data files. All the data in the PLS can be backed up on file, and restored from file.

The Assignment

The assignment consists of the following parts:

1. A Requirements specification
2. A system design consisting of
   - ➔ a Use Case diagram (preferably implemented in Astah)
   - ➔ a Class Diagram (preferably implemented in Astah)
     - ◆ Astah is recommended for the diagrams, but if you prefer other design tools, you need to properly include the diagrams in your documents. See the deliverables.
   - ➔ an optional explanation of the design in natural language
3. An implemented and operational system
4. A user manual for the PLS

Requirements Specification

The **Requirements** document should cover at least the following stakeholders:

1. Librarian
2. Subscriber
3. Publishing Company

The latter only needs to have access to the PLS in a way similar to a Subscriber.

System Design

The design must consist of a **UML Class Diagram** and an optional textual explanation for this diagram. The Class Diagram should at least cover the following classes:

1. PublicLibrary
2. Book, BookItem (a paper copy of a book)
3. Subscriber
4. Librarian
5. Person
6. Catalog
7. LoanAdministration
8. LoanItem

An author does not need to be a Person. You may limit Author to just a name (a string). Especially the relationships between the classes should be shown in the diagram. Only the most essential attributes and methods per class have to be expressed.

The **Use Case diagram** should cover all stakeholders that have direct access to the system, and all the main functions of the system.

Implemented and Operational System

The implementation should consist of a **Python** program which is clearly an implementation of the design. It should consist of two main parts: the backend and the frontend. The backend should implement all data processing and data storage and retrieval. The frontend should be a user interface, which may be a console based (textual) interface, giving access to all the functions of the system. Additional classes, attributes and methods (not made explicit in the design) are allowed. **The system must be built only in Python 3 using only modules in the standard library of Python 3**. If you know about databases and SQL you may implement the data storage with the module sqlite3.

The system must feature at least the following functions:

- Searching for a book. This shows if there are book items in the library present for this book and if they are available. The search criteria must cover various keys, such as title, author, ISBN, etc, and combinations of keys.
- Making a book loan for an available book item.
- Adding new customers.
- Adding new book items.
- Making a backup of the data in the system (catalog of books and book items, data in the loan administration, list of customers, etc.)
- Restoring the library from a backup.

User Manual

A User Manual should be provided to explain how to use the system, for both the librarian and for subscribers. Any information needed for users of the system should be clearly provided in this document, including commands, shortcuts, usernames and passwords (if any), etc. This manual may contain screenshots, examples, diagrams. This information may also be used by your teacher to run, test, and assess the system.
Do not suppose that the teacher will explore your code to find out how to run the system.

Submission

Deliverable

The delivery to be handed in must consist of **one zip-file**, named as below:

studentnumber1_studentnumber2_studentnumber3.zip

The zip-file must contain:

1. A **pdf document**, called **PLS-Documentation.pdf**, containing:
   a. **Names** and **student numbers** of the team (maximum **3** students per team),
   b. The **Requirements Specification**,
   c. The **Class Diagram** and, if needed, textual explanations,
   d. The **Use Case diagram**,
   e. A **User Manual** explaining, with examples, how to use the system, for both the librarian and for subscribers. **This is needed for your teacher to test and assess your code.**
2. A directory called **PLS-SourceFiles**, containing all the **code files** and the **data files**, including one main file **PLS.py**. Starting the system should be done by running **PLS.py**.
3. **[Optional]** **Astah files** (*.asta) for the Class Diagram and Use Case Diagram. If you do not use Astah, you may just have the diagrams only in the pdf file explained in the No. 1, above.

⚠ Important Notes

1. **Do not** include any **bulky** Python system files in the delivery.
2. The code must **only** use **standard library modules**.
3. The code must run **error-free**.
4. The code should **only** write to **temporary storage** directories on the local machine, meaning on the current (running) folder or a subfolder of it.
5. We encourage you to work in a **team of 2 or 3 persons**. However, individual work is also acceptable, if you prefer to do it individually, or you are not able to make a team (specially retakers).
6. When working in a team, **only one team member (the team leader)** submits the assignment, and all group members submit a group-info message with names and student numbers of the entire team, clearly indicating who is the team leader. **Feedback will be given to the team leader only**, who will then communicate it to the other team members.
7. You are **not allowed** to work together with someone who has a different teacher! Here we mean the teacher in your schedule. These combinations are allowed:
   - INF1F
   - INF1B + INF1E
   - INF1C + INF1G
   - INF1D + INF1J
   - INF1H + INF1I

How to submit?

- **Regular Students** can only submit it via the appropriate channels in MS Teams.
- **Retake students** of the last year can directly send their assignments by email to bashb@hr.nl.

Deadline

~~Submission deadline is **25 APRIL 2021** and will be also announced in MS Teams.~~

Retake: Submission deadline is **2 July 2021** and will be also announced in MS Teams.

Grading

For assessment of the delivery, the documentation file must comply with the requirements mentioned above. The implementation will be tested for the following functions:

- filling the system with books from a file of books in JSON format (file with books). You may add a fake ISBN number to book items and generate the number of book items (copies) per book that the library has.
- filling the system with customers from a file of people (file).
- adding a book item.
- adding a customer.
- searching for a book.
- making a book item loan.
- making a backup of the system in JSON format.
- restoring the system from a backup.

Minimal requirements for being graded as sufficient

- There must be a sensible **Requirements Specification**, **Use Case Diagram** and **UML Class diagram**. The latter must express all of the classes mentioned in the assignment text, with sensible relationships between classes and with the correct graphical symbols (inheritance should be expressed with the inheritance arrow, association with the association line or arrow, etc.). The associations must have a name. The class boxes must contain the class name but don't need to contain the complete list of attributes and methods used in the code. The Class diagram should first of all express the relationships between classes.
- A visual inspection of the code must show that the classes in the UML diagram have been implemented as classes in the code (Python3) and that the functions of the system have been assigned to the proper class or are implemented as a separate function. For instance, searching for a book should be a function of class Catalog, not of class Book (other sensible solutions are possible). In general everything must be defined at the proper level and the proper place in the code.
- Among the **8 functions required for the system** (filling with books, filling with customers, adding a book, adding a customer, searching a book, making a book item loan, making a backup, restoring from a backup), **the book item loan plus at least 4 other functions must be clearly implemented in the code**. The loan of a book item must be administered such that the book item cannot be borrowed twice at the same time and that the customer borrowing the book item is known to the system as the borrower.
- The required user manual must give descriptions and examples for all the implemented functions of the system.
- If the visual inspection leaves serious doubt about the proper functioning of the system, **the student team may be asked to demonstrate the system to the teacher**, which should then function properly during the demo.

Request for Regrade

If student(s) have concern about their grade, they can directly contact their teacher via email, within 5 working days after feedback is provided in the MS Teams.