Innopolis University
Information Technologies

INTRODUCTION TO PROGRAMMING II

Course Project

Implementation of the Library Management System "LSD"

REPORT FOR DELIVERY III

Professor:

Eng. Daniel de Carvalho

Members:

Vladimir Scherba

Godfred Asamoah

Roberto Enrique Chavez Rodriguez

Semester:

Spring - 2018

CONTENT TABLE

1 Objectives.

1.1 General Objective.

- To successfully implement the LSD library management system by means of applying all the concepts and theory learned during class.

1.2 Specific Objectives.

- To comply with the requirements established by the Project description in deliveries I, II and III.
- To apply the concepts learned in theory class and lab sessions.
- To develop a successful Project.
- To comply with the observations made on the project presentation for delivery II

2 Background information of deliveries.

2.1 Project delivery I

In project delivery I, the following features of the project were implemented:

- Initial construction of Classes: implementation of data Structures representing the material of the library and user.
- Implementation of the feature Checkout: With this feature
- Implementation of the test cases: successful implementation of all cases for delivery I yielded results as expected.

2.2 Project delivery II

For delivery II, a restructuration of the project was implemented in order to enhance the performance of the system.

The refactoring was carried out in the different modules of the project.

According to the specifications of the project description in delivery II, the following features were also added:

- System-user interface: The telegram Bot is implemented for interaction of the system with the user.
- The system has the features: Returning, Editing, Adding and Removing documents.
- The system can add, remove, and modify users.
- Implementation of the storage system: The Data Base was implemented by means of the DataBase ***PostgreSQL***

Also, observations to the project were made regarding the implementation of the test cases. Such observations referred to the approach that was given to the implementation of the ***Test Cases*** for the delivery. The approach given was to test the implementation of the project, whereas the required approach involved an overall testing, i.e. the test cases implementation had to involve also the functionality of the interface for user interaction.

3 Project delivery III.

For the third delivery, the new amendments established for this delivery were implemented.

Also, the observations made in the second delivery (see *section 2.2*) were corrected and now the implementation can fully meet the requirements for this delivery.

3.1 Current Status.

The library system currently operates under the conditions specified by the corresponding Project requirements.

Besides the functionality of the implementation in delivery II, the following features are added:

- The name of the system is finally given for this Delivery: **LSD – *Library System in Debellate***
- A queue is implemented for the patrons who want to check out a certain document and are waiting for it.
- The system integrates now the new patron functionality: Visiting Professor
- The return system is implemented: Fine and renew subsystems
- The functionalities required for each user (Librarians and Patrons) are now implemented, and they work in the interface.

3.2 Improvements Implemented for this version.

Improvements and modifications were implemented for this version of the project:

- The previous class Resources in project implementation for delivery II is split in two different modules now: Items module and Storage-resources module. This allows for interaction in the Data Base.
- Actual user Interface module (ui) is expanded with more classes so that the interface manages the different commands of the users.
- The class OutstandingResquestCommand is implemented in the module Controller. This allows for the librarian to perform an outstanding request.
- Lots of bugs and errors found in the interface were corrected for a better performance of the system.

4. Project Details.

4.1 Technical specifications.

Computer programs used for the implementation:

- Computer Programming language: ***JAVA***
- Development environment for java: ***Intellij IDEA***
- Build system: ***Maven***
- Data Base management system: ***PostgreSQL***
- Programming Interface selected for User-system interaction: ***Telegram Bot***
- Repository hosting Unit: ***GitHub***

4.2 Access to project core.

The project is uploaded on GitHub, and it can be currently accessed to by means of the following link:

https://github.com/Harrm/LibrarySystem

The specifications of running the project from there are detailed in the section ***README.md*** of the site.

5 Architecture of the Project.

Again, the project is subdivided in different modules in order to have a more organized system implementation. For further details in the implementation of the code, all classes are commented in the different methods and functions that are implemented in each single class of the project construction in Java.

5.1 Modules.

The project contains the following modules:

- **Controller:** the function of this module is to control the system behavior. In this module, there are the functions to add items, users, and mainly the functionalities of user librarian are implemented here.
- **Items:** Formerly called *Resources,* it is the core implementation of the project. This module contains implementations of data structures for the library items and Users.
- **Storage**: API (application program interface) between the system and the database. Also in this module, the function to deserialize an item or user into its features (editors, user_id, etc.) is implemented.
- **Storage – Resources:** This module which works inside the Storage module is in a way an extension of the module Items which objective is to have static instances of Resource class to map database tables to the Entry classes. And every Entry class may contain an item or not.

- **User interface:** Consisting of sub modules commands and actual user interface, this module yields commands and GUI implementation, which allows a user to interact with the system.
- **Test:** The purpose of this module is to verify (test) the implementation of the project. Test cases of deliveries are implemented in this module.

5.2 Class Hierarchy of Modules and Classes

The content of the implementation in each module is shown below, showing the inheritance of super classes and subclasses that they contain.

Classes and Interface of module Controller.

- AddItemCommand (implements interface *Command*)
- AddUserCommand (implements interface *Command*)
- CheckOutCommand (implements interface *Command*)
- *Command* (Interface)
- LibraryManager
- OutstandingRequestCommand (implements interface *Command*)
- RenewCommand (implements interface *Command*)
- ReturnCommand (implements interface *Command*)

Classes and subclasses of module Items.

- Item
  - AvMaterial
  - Book
  - JournalArticle
  - JournalIssue
- ItemFactory<T extends Item>
  - AvMaterialFactory
  - BookFactory
  - JournalArticleFactory
  - JournalIssueFactory
- User

Classes and interface of module Storage.

- **Resources Sub-module**
- ItemSerializer
- QueryParameters
- SglQueryExecutor
  - SqlStorage (Implements Interface *Storage*)
    - LibraryStorage
- *Storage* (Interface)

Classes and subclasses of Sub-module Resources.

- DatabaseEntry
  - ItemEntry
    - AvMaterialEntry
    - BookEntry
    - JournalArticleEntry
    - JournalIssueEntry
  - UserEntry
  - Resource
  - CheckoutEntry
  - PendingRequestEntry

Classes of module User Interface.

The module User Interface integrates two main Sub modules as shown in the figure

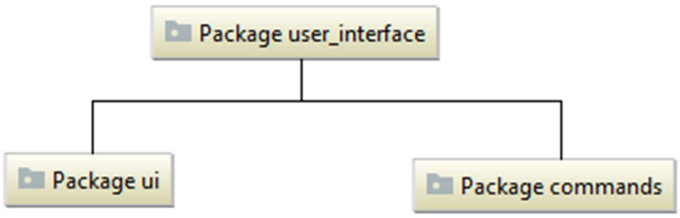


Figure1: Sub-modules of Modules User Interface

Classes of Sub-module Commands:

- Command
  - AddCommand
  - AddParser
  - CheckoutItemCommand
  - EditCommand
  - EditParser
  - ErrorCommand
  - FineCommand
  - LoginCommand
  - MenuCommand
  - OutstandingCommand
  - RenewCommand
  - ReturnItemCommand
  - SignUpCommand
  - StartCommand
- NotificationHandler

Classes of Sub-module actual user interface (ui)

- Bot
- Interface
- KeyboardUtils

Module Test

- BookReturnEditSystemTests
- CheckoutTests
- LibraryStorageTest
- OutStandingRequestTest
- RequestQueueTests
- TestIems

5.3 Interface.

As mentioned before, the interface selected for the project is implemented in a Telegram bot. This is the basic information of the bot:

Bot Name: ***Konyvtar***

Bot address: **@konyvtar_bot**

6. User Specifications.

From the user point of view, the use of a platform such as ***Telegram*** provides a friendly environment for the user to have an easy understanding of how to operate the bot. Furthermore, the system is interactive and lets the user know in every step of accessing the system and booking an item what the user must do. The following points explain what a user (Patron in this case) can do in the bot.

6.1 How to access the bot.

In order to access the bot:

- First go to the indicated bot address above on Telegram
- The user needs a Token (username and password)
- If the user does not have an account, they must contact the Librarian
- Press /start when the user will use the Bot for the first time
- Press /login when the user has already an account.

The next sections describe the features that the system users can do.

6.2 Features of User

6.2.1 Librarian.

The librarian is able to:

- Add and remove users.
- Add and remove items.
- Renew documents and perform fines.
- Perform outstanding requests
- Edit features of documents and users.

6.2.2 Patron.

The following are the features that a user can access to once the user is inside the system:

- Select documents
- Checkout Documents.
- Return Documents.
- Renew loan items for time extension

Some of the functionalities are shown below:

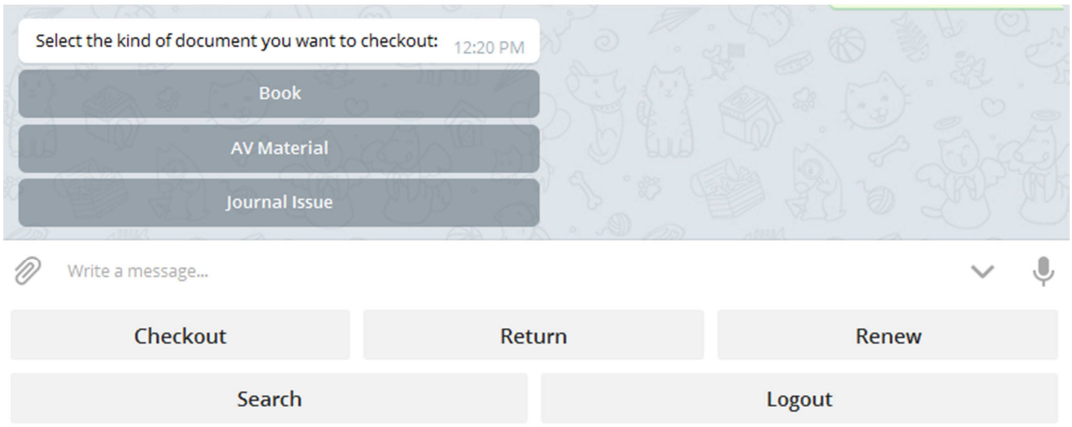


Figure 2: Display of functionalities for a patron when checking out an item

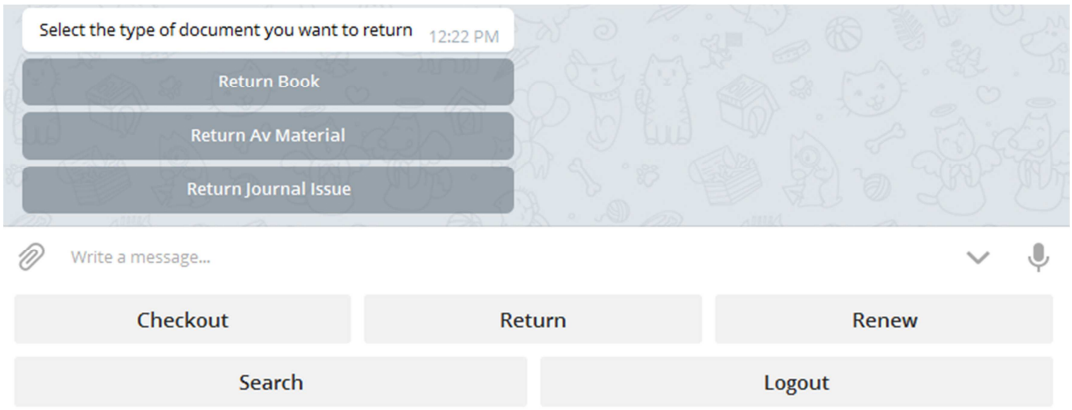


Figure 3: Display of functionalities for a patron when returning an item

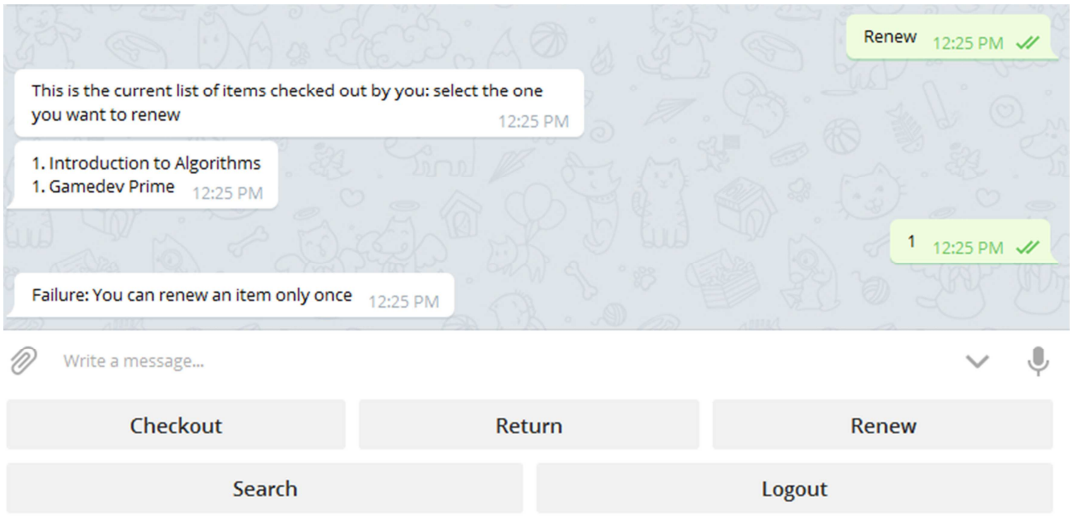


Figure 4: Display of functionalities for a patron when returning an item

**** The interface features for a Faculty Member, Student, and Visiting Professor are the same in the bot. However, the interface handles the differences among functionalities when the patron, either type this might be, performs and operation in the bot.***

7 UML Diagrams.

In this section Unified Modeling Language (UML) diagrams are shown to describe the software architecture using several types of diagrams: use case diagrams, class, package, component, composite structure diagrams. The diagrams are shown in correlatively in the following order:

The legend of the diagram is as follows:

Blue Dashed line: It shows inheritance A ⟶ B (Class A inherits from class B)

Green Dashed line: It shows Interface implementation A -------> B (Class A implements interface B)

Black Dashed line: It shows dependencies A -------> B (A has dependency to B)

Letter **m** stands for method, **c** stands for class, **p** stands for property/attribute, letter **I** stands for interface, **f** stands for field, and the **open and closed hooks** stand for access level modifiers, open hook means public method/property, and closed hook means private/protected method/property.

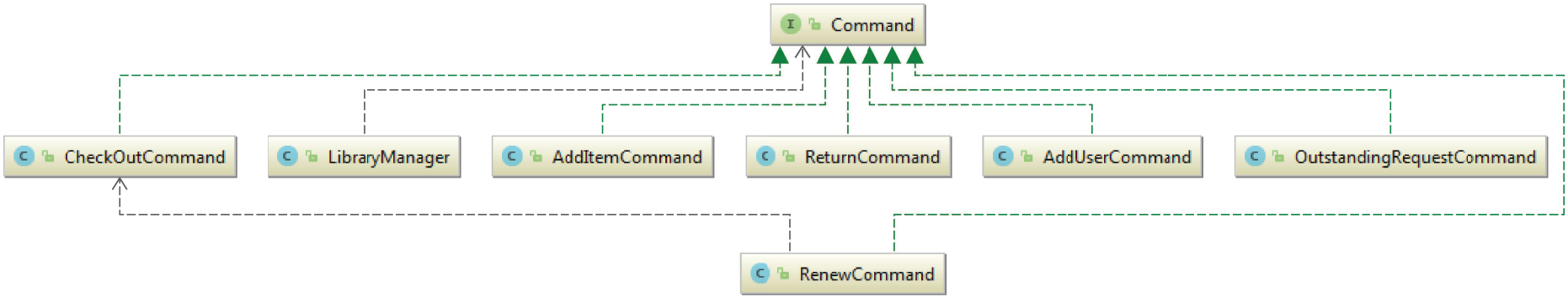
7.1 Inner Structure.

7.1.1 Controller Module

Diagram Description: The diagram shows the connection between the classes and the interface in the module, where the interface Command is implemented by its corresponding classes. The diagram also shows a dependency LibraryManager - Command

Author: Vladimir Shcherba

Content of the Diagram: Classes and interface

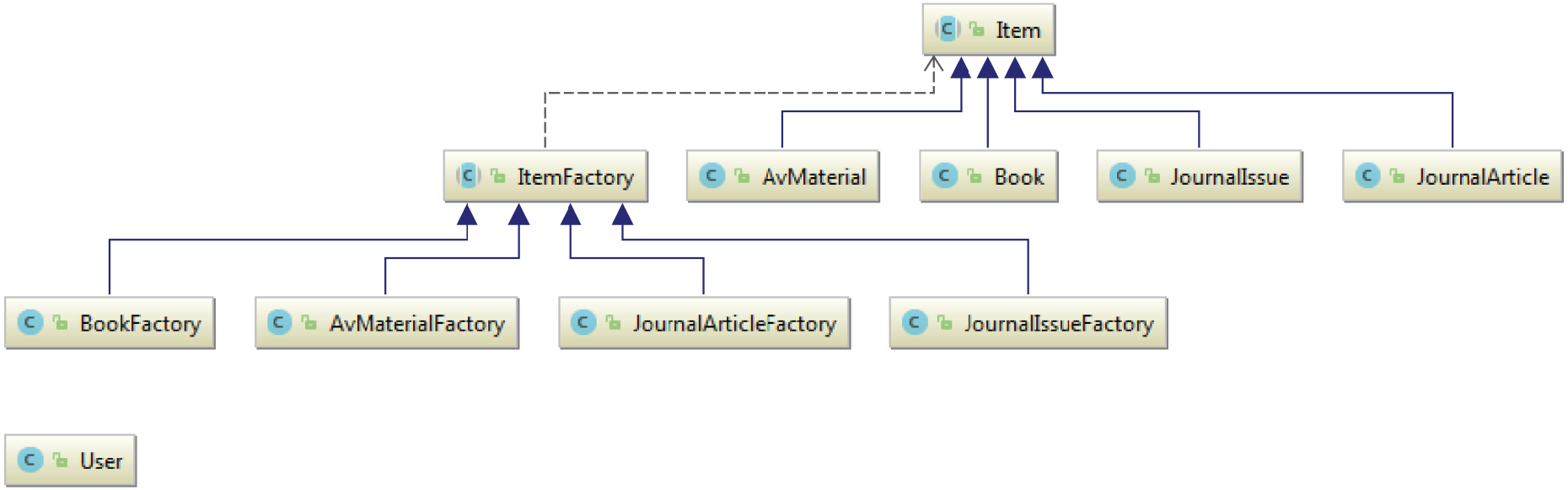

Command
CheckOutCommand
LibraryManager
AddItemCommand
ReturnCommand
AddUserCommand
OutstandingRequestCommand
RenewCommand

7.1.2 Items Module

Diagram Description: The diagram shows class inheritance and dependencies. User class is shown as an independent class, not inheriting from any other.

Author: Roberto Chavez

Content of the Diagram: Classes

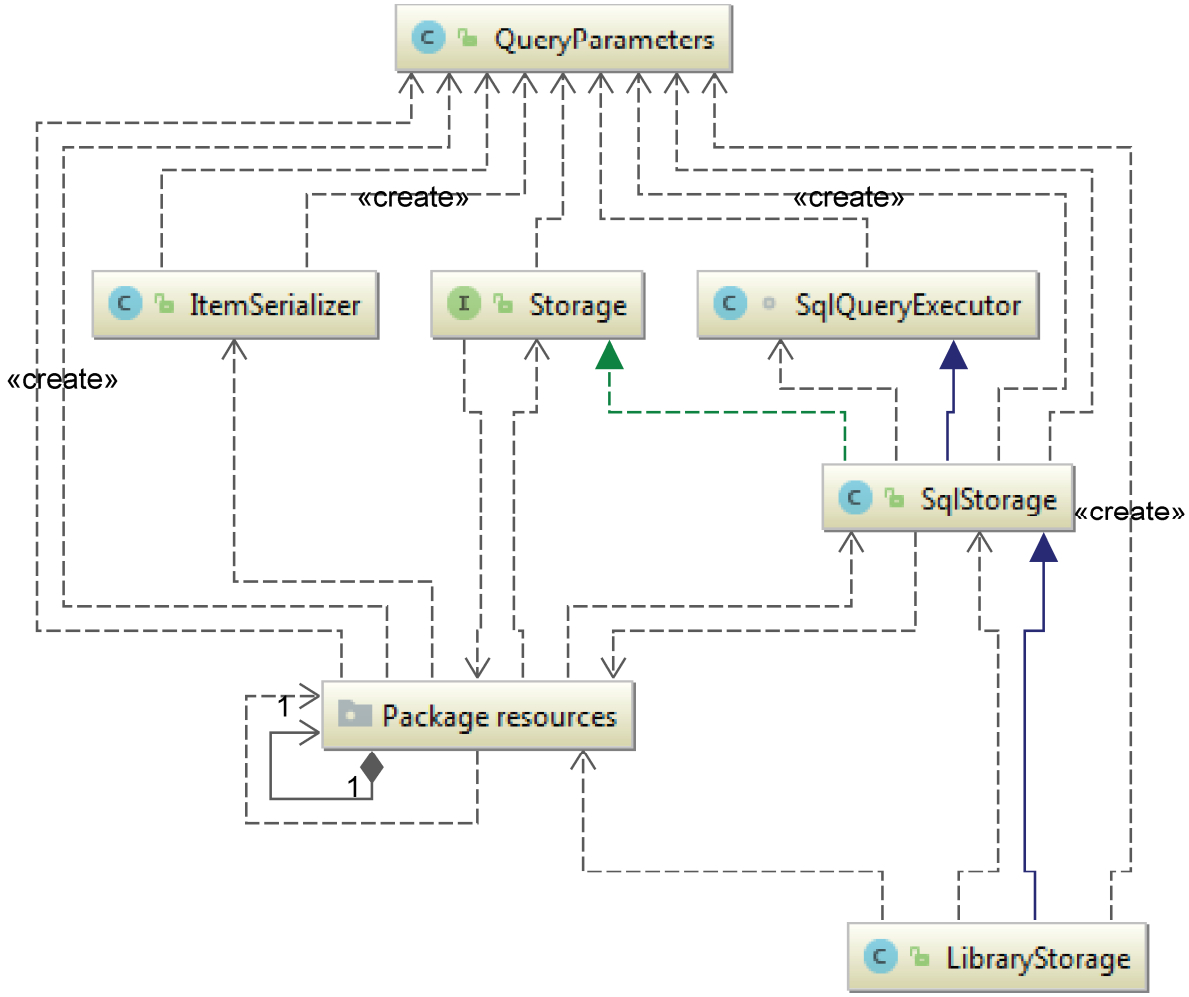

Item
ItemFactory
AvMaterial
Book
JournalIssue
JournalArticle
BookFactory
AvMaterialFactory
JournalArticleFactory
JournalIssueFactory
User

7.2 Storage Module (Data Base).

Diagram Description: The diagram shows the relations between the classes and the interface Storage in the module. The sub module Package resources is shown in this module.

Author: Vladimir Shcherba

Content of the Diagram: Classes, Sub-module and interface

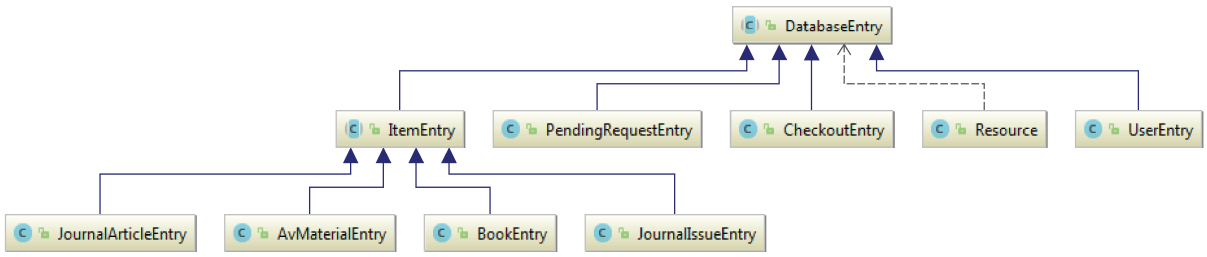

QueryParameters
«create»
«create»
ItemSerializer
Storage
SqlQueryExecutor
«create»
SqlStorage
«create»
1
Package resources
1
LibraryStorage

7.2.1 Resources Module.

Diagram Description: Resources Sub module. The diagram shows class inheritance and the dependency Resource - DatabaseEntry

Author: Vladimir Shcherba

Content of the Diagram: Classes and interface


DatabaseEntry
ItemEntry
PendingRequestEntry
CheckoutEntry
Resource
UserEntry
JournalArticleEntry
AvMaterialEntry
BookEntry
JournalIssueEntry

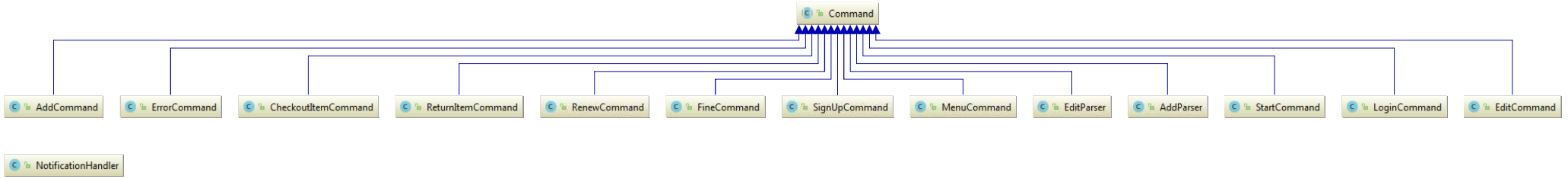
7.3 User Interface.

7.3.1 Commands.

Diagram Description: The diagram shows the class inheritance of the module. Class NotificationHandler is a single class with no inheritance.

Author: Godfred Asamoah

Content of the Diagram: Classes and interface

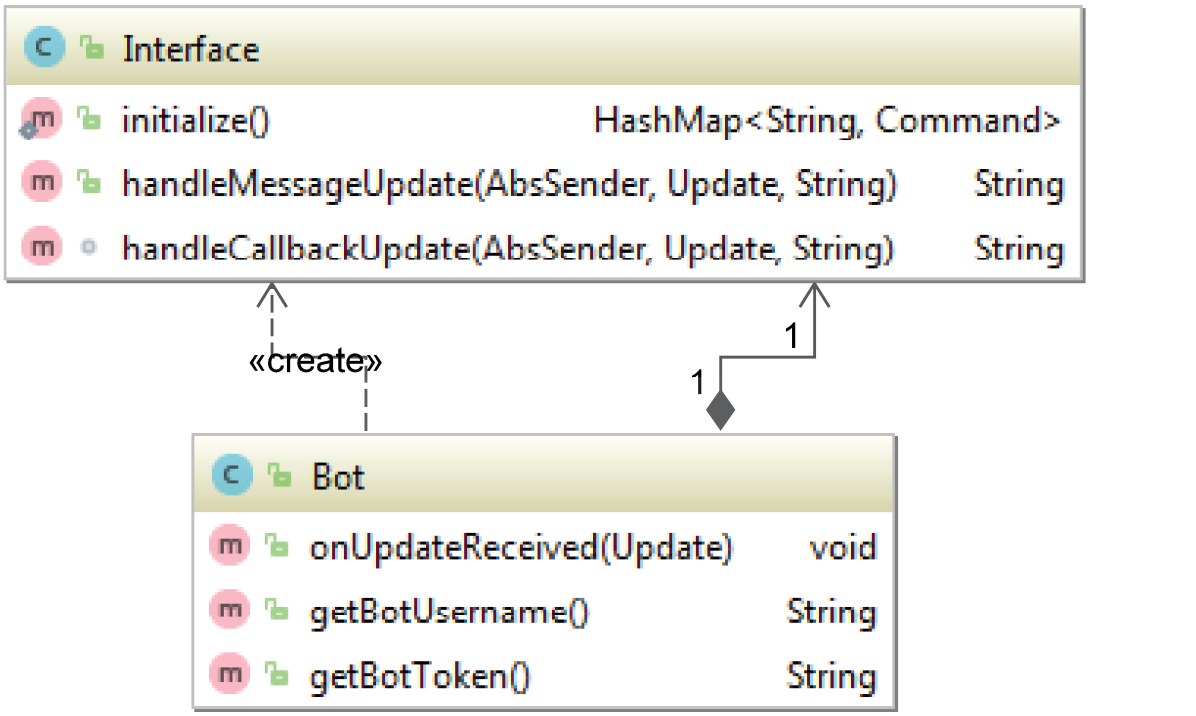

Command
AddCommand
ErrorCommand
CheckoutItemCommand
ReturnItemCommand
RenewCommand
FineCommand
SignUpCommand
MenuCommand
EditParser
AddParser
StartCommand
LoginCommand
EditCommand
NotificationHandler

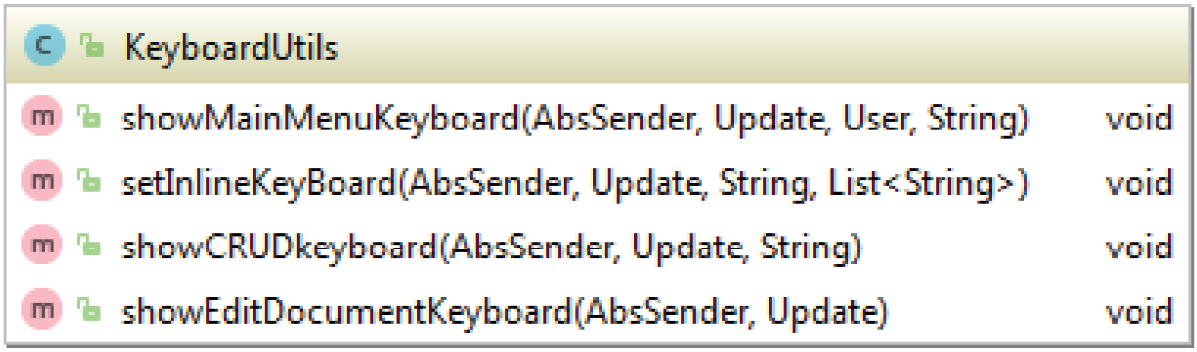
7.3.2 UI Telegram Bot.

Diagram Description: The diagram shows the classes that conform the Telegram Bot and their main methods. There is a dependency as observed with classes Bot and Interface.

Author: Godfred Asamoah

Content of the Diagram: Classes and methods


Interface
initialize() HashMap<String, Command>
handleMessageUpdate(AbsSender, Update, String) String
handleCallbackUpdate(AbsSender, Update, String) String
«create»
1
1
Bot
onUpdateReceived(Update) void
getBotUsername() String
getBotToken() String


KeyboardUtils
showMainMenuKeyboard(AbsSender, Update, User, String) void
setInlineKeyBoard(AbsSender, Update, String, List<String>) void
showCRUDkeyboard(AbsSender, Update, String) void
showEditDocumentKeyboard(AbsSender, Update) void

8 Conclusions and further enhancements.

- The system runs according to the requirements of the project.
- Many of the concepts learned in Theory class and Lab sessions are applied to this project. Some of them are Class inheritance, Data structures, and Object-oriented programming.
- The telegram bot turned out to be a good environment for user, as it provides them with an easy-access way to interact with the library.
- The test cases for the delivery II were successfully implemented and this new delivery of the project also incorporates test case which prove the correctness of the system.