

**Innopolis University**  
**Information Technologies**

# **INTRODUCTION TO PROGRAMMING II PROJECT IMPLEMENTATION OF A LIBRARY MANAGEMENT SYSTEM DELIVERY II**

**Professor:**

**Eng. Daniel de Carvalho**

**Students:**

**Vladimir Scherba**

**Godfred Asamoah**

**Roberto Enrique Chavez Rodriguez**

**Semester:**

**Fall, 2018**

## **1 General Objective.**

- To implement a library system management in which we apply all the concepts and theory learned during class.

### **1.1 Specific Objectives.**

- To comply with the requirement established by the Project description.
- To apply the concepts learned in theory class and lab sessions.
- To develop a successful Project.

## **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 are 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 is implemented by means of the DataBase **PostgreSQL**

### 3 Specifications of Project implementation.

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**

#### 3.1 Access to the project core.

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

<https://github.com/Harm/LibrarySystem>

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

#### 3.2 Structure of the project.

The project is subdivided in different modules in order to have a more organized system implementation. Below is a summary of the description of the different parts of the project. 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.

##### 3.2.1 Modules

The project contains the following modules:

- Controller: the function of this module is to control the system behavior. At the moment in this module, there are the functions for booking and returning, adding, and removing an item from the library system.
- Resources: It is the core implementation of the project. This module contains implementations of data structures for library items and user cards and classes that build and serialize/deserialize them.
- Storage: API (application program interface) between the system and the database.
- User interface: Consisting of commands and actual user interface. This module yields commands and GUI implementation, which allow 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.

### **Class Hierarchy of modules.**

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

#### **Classes of module Controller.**

- BookingController
- ReturnController

#### **Classes and subclasses of module Resources.**

- CheckoutRecord
- Item
  - AvMaterial
  - Book
  - JournalArticle
  - JournalIssue
- ItemFactory<T>
  - AvMaterialFactory
  - BookFactory
  - JournalArticleFactory
  - JournalIssueFactory
- ItemSerializer<T>
  - AvMaterialSerializer
  - BookSerializer
  - JournalArticleSerializer
  - JournalIssueSerializer
- User

#### **Classes and interface of module Storage.**

- QueryParameters
- SqlQueryExecutor
  - SqlStorage (implements interface Storage)
- Storage (Interface)

#### **Classes of module User Interface.**

- Main

#### Classes of Sub module Commands:

- Command (Interface)
- LoginCommand (implements interface Command)
- SignUpCommand (implements interface Command)
- StartCommand (implements interface Command)

#### Sub module actual user interface (ui)

- Bot

#### Module Test

- ReturnSystemTest
- StorageTest
- BookingSystemTest

#### 3.2.2 Interface

The interface selected for the project is implemented in a Telegram bot. This the basic information of the bot:

Bot Name: *Konyvtar*

Bot address: *@konyvtar\_bot*

#### 4 Specifications of User.

From the user point of view, the use of the bot is friendly as it provides the user with 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.

##### 4.1 How to access the bot.

In order to access the bot:

- First go to the indicated bot address above on Telegram
- Press */start* – enter.
- Follow the next steps as shown in the Fig 1. The bot will guide the user through them.

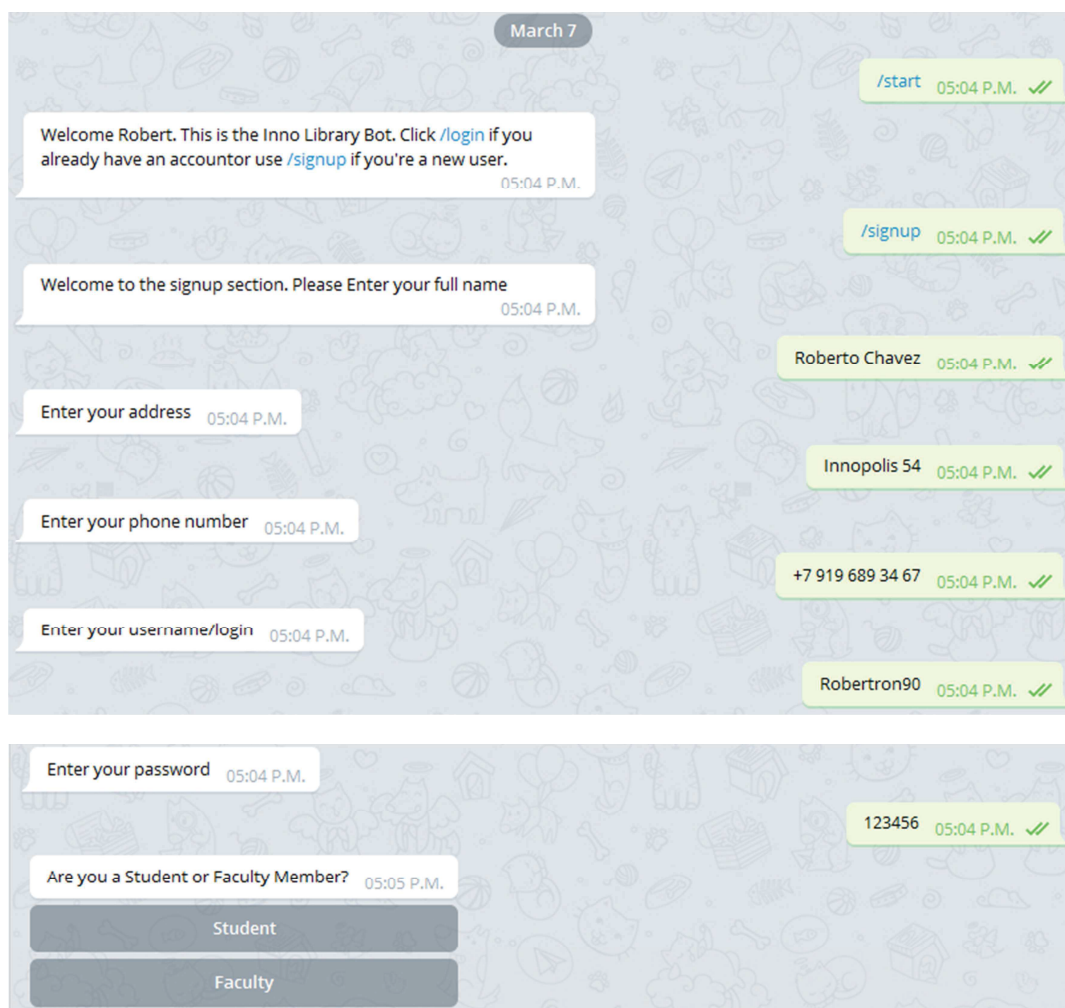


Fig 1. Data to enter in the library system to sign up as a user (Patron)

After the creation procedure as a new user, the user will be able to confirm their sign up process as shown in Fig 2.

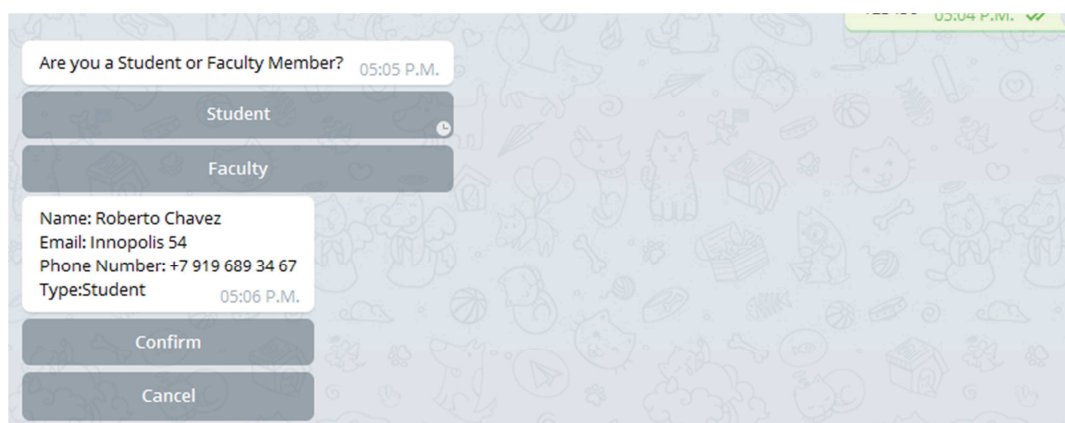
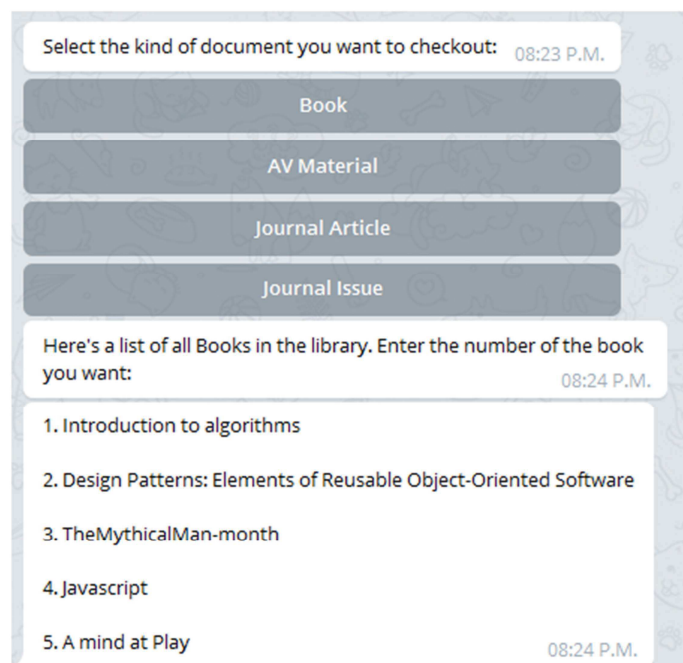
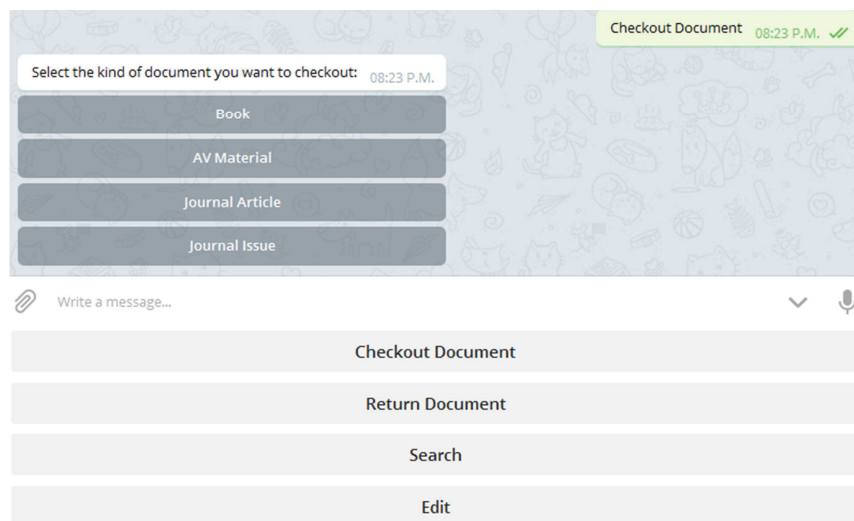


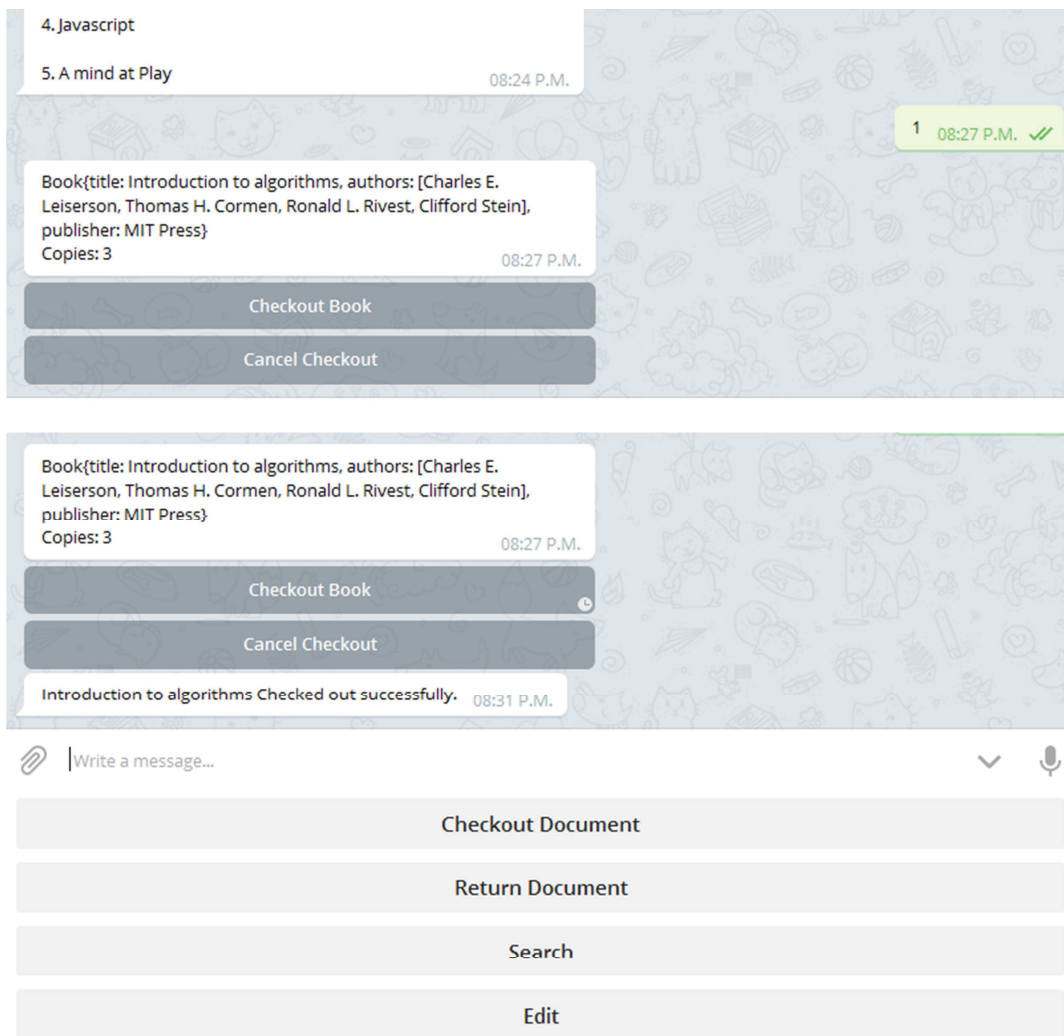
Fig 2. Confirmation before creating a new user in the Telegram Bot.

## 4.2 Features that the user can access in the bot.

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

- Search Documents.
- Checkout Documents.
- Return Documents.
- Edit documents and users – Feature that can be used by Librarians only.





*Fig.3 Telegram bot interface with the user. Example of how to checkout a book*

## 5 UML diagrams

In Annex 1 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:

Letter **m** stands for method, **c** stands for class, **p** stands for property/attribute, letter **l** 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. **Dashed lines** mean for dependencies in classes and **straight lines** mean inheritance in classes



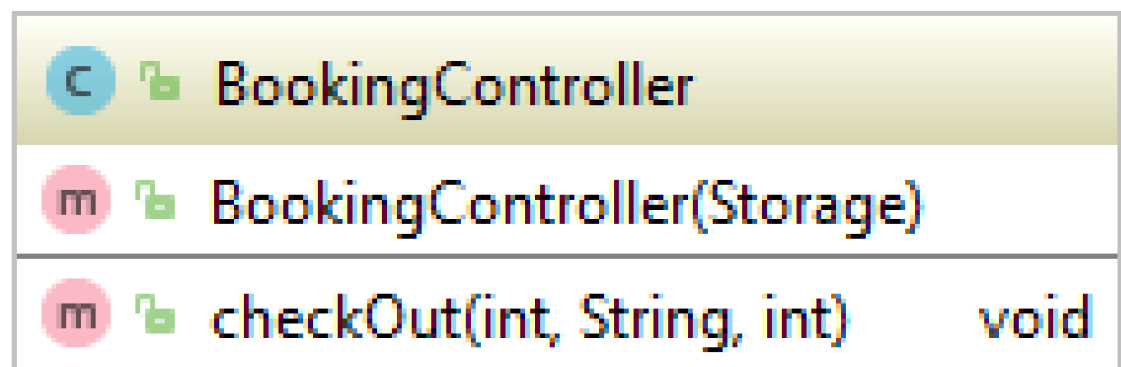
1. Controller Module: Showing dependencies of the classes and inner classes
2. Resources Module 1: Showing dependencies of the classes in relation to one another. Also the methods that each class contains are shown there.
3. Resources Module 2: The properties of each class are shown in this diagram.
4. Storage Module 1: In this diagram, the dependencies with each class are shown.
5. Storage Module 2: The methods implemented in the classes are shown in this diagram.
6. Storage Module 3: This diagram shows the interface “Storage” with the methods that can be used in this interface.
7. User Interface Commands: The dependencies of the module Commands is showed.
8. Telegram Bot Properties.
9. Test Module: The diagram shows basic implementation of this module. This module is where the test cases are implemented.

## **6 Conclusions and Further improvements.**

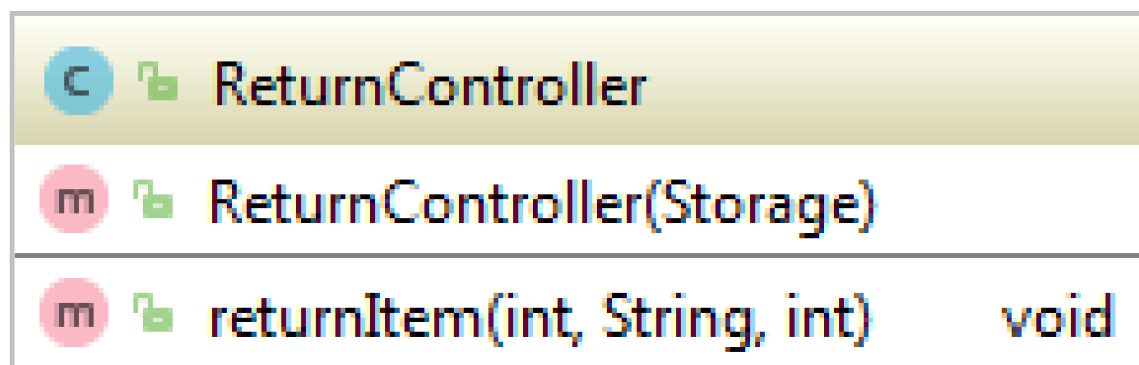
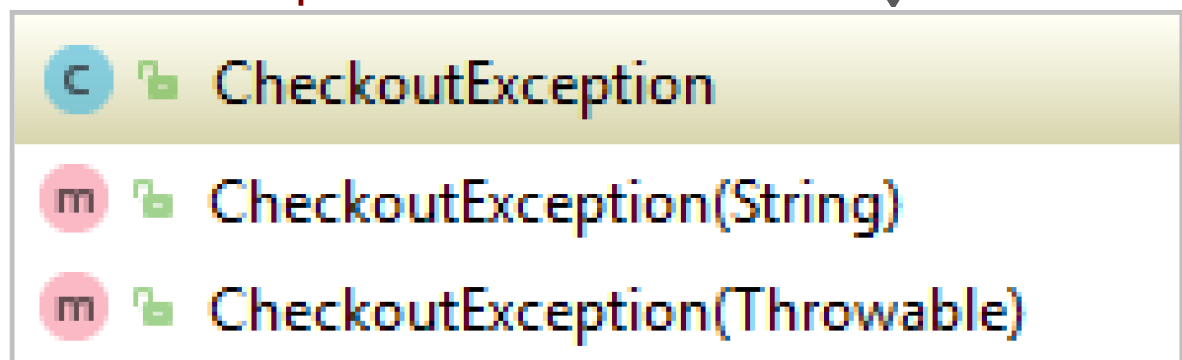
The system works fine for the implemented features

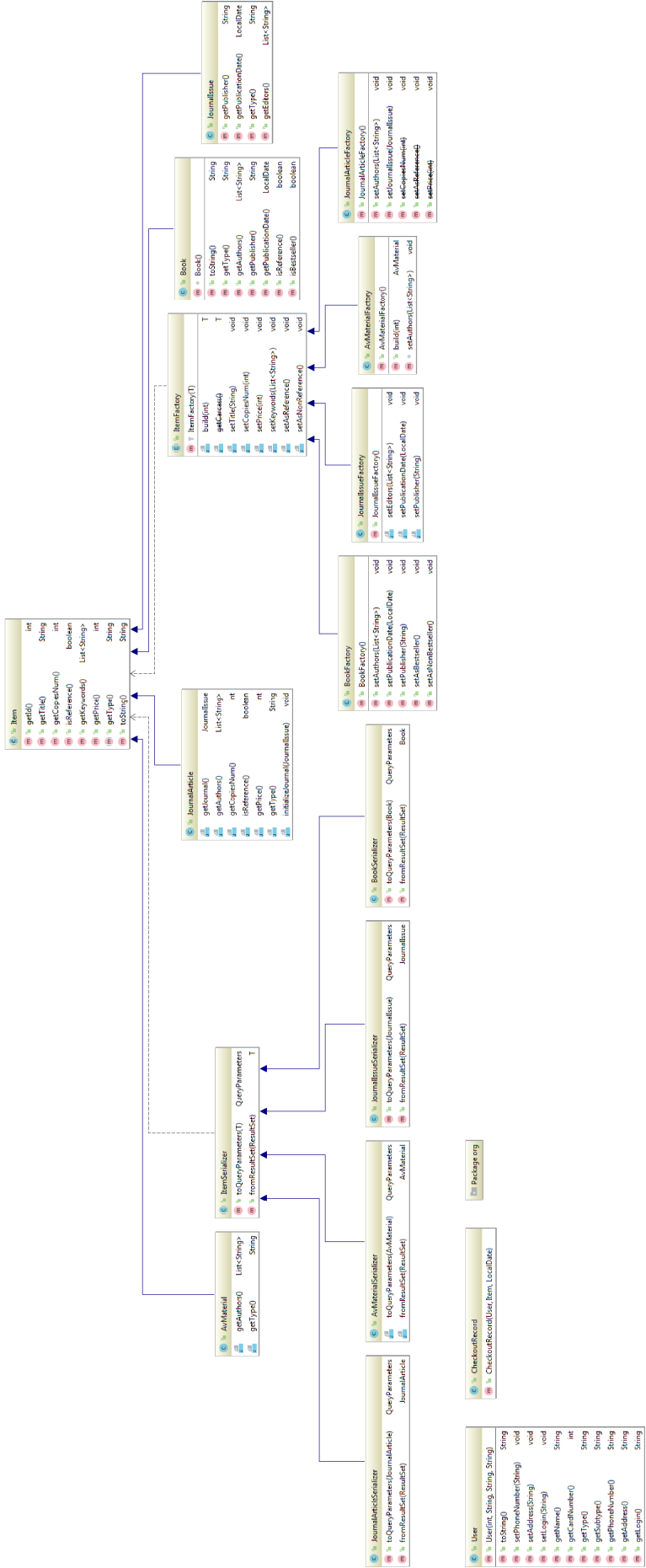
For the next delivery, the new features that the project requires for this delivery will be implemented as well as the observations made by the professor during the project presentation; in particular, the implementation of test cases.

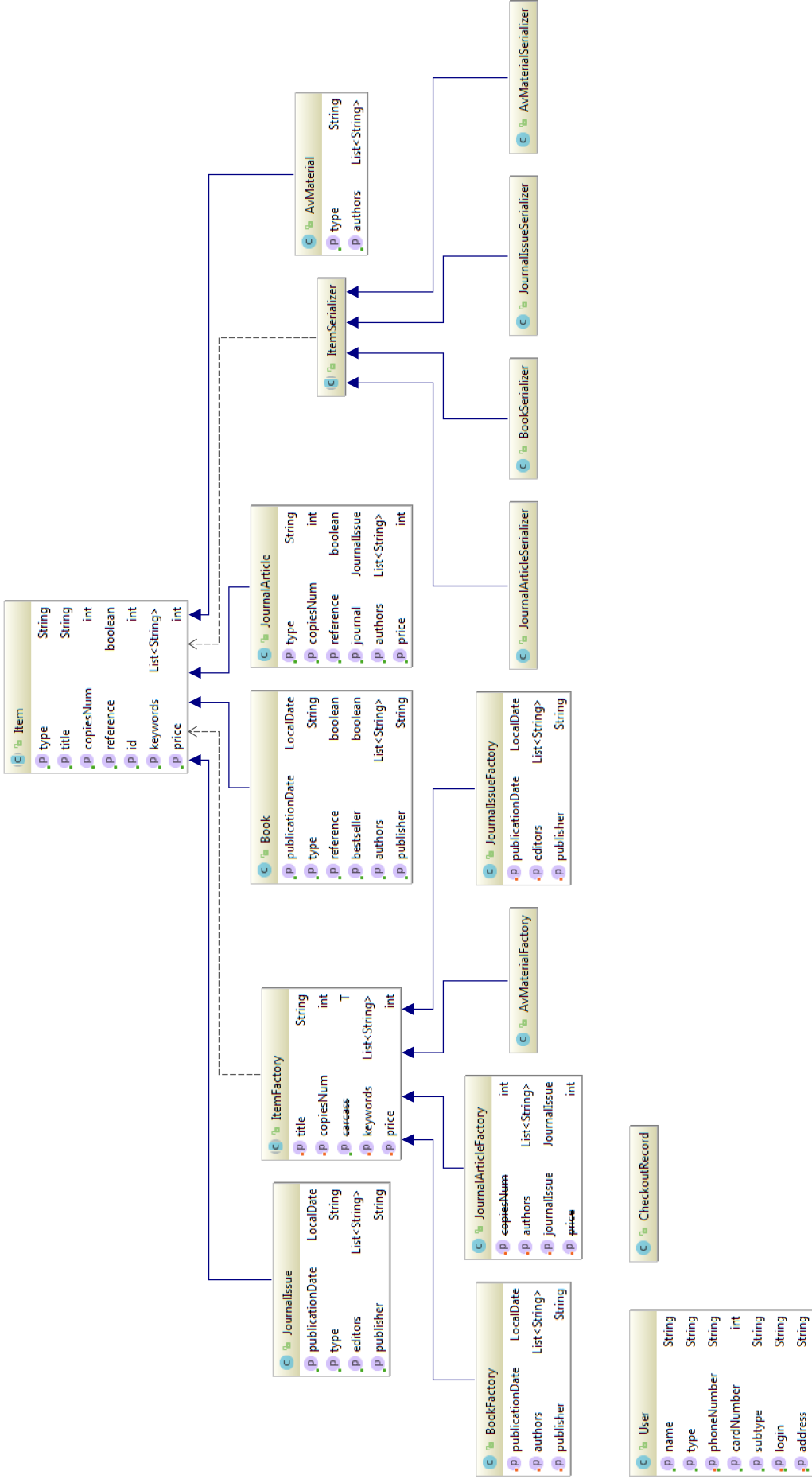
## **ANNEX 1: UML DIAGRAMS OF THE PROJECT IMPLEMENTATION.**

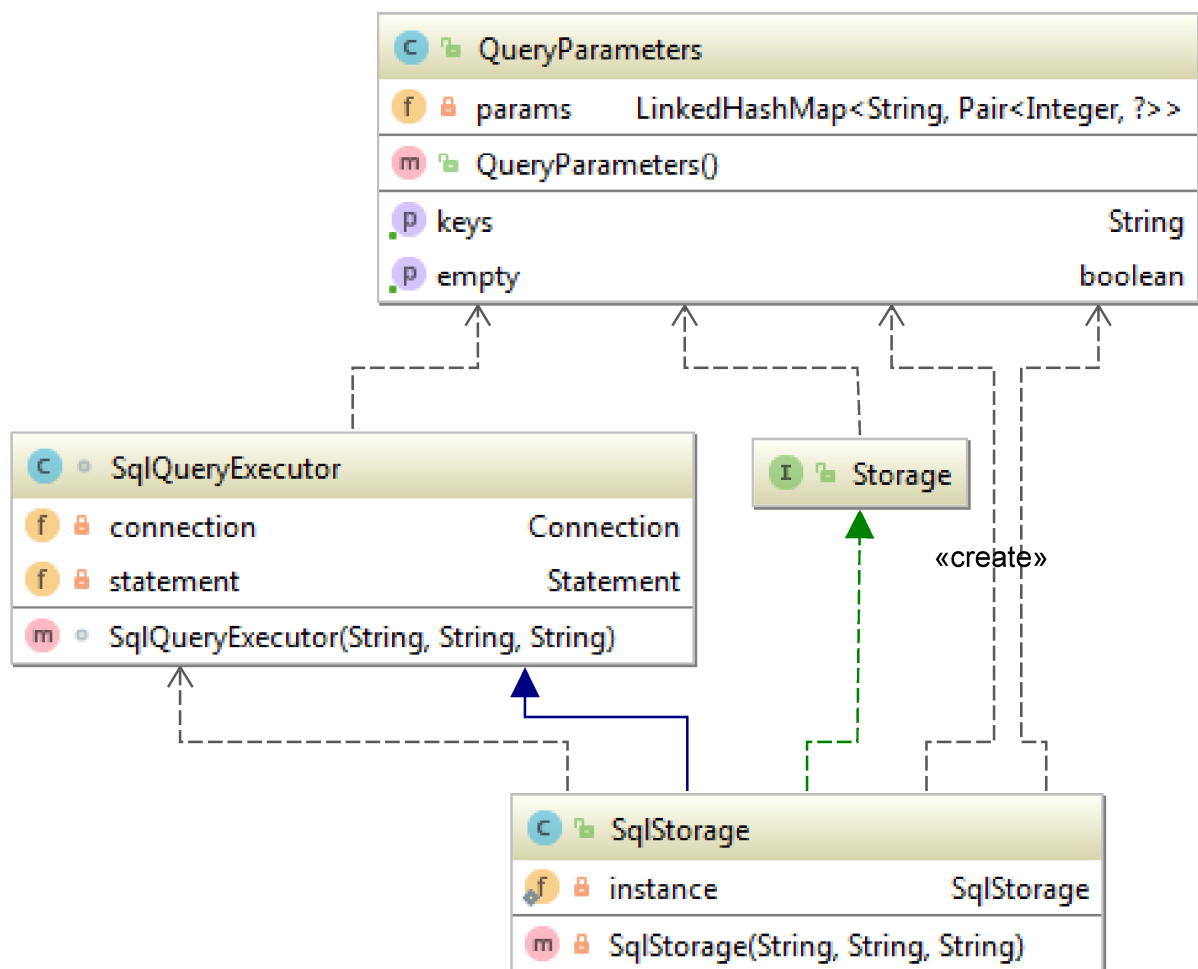


«create»

































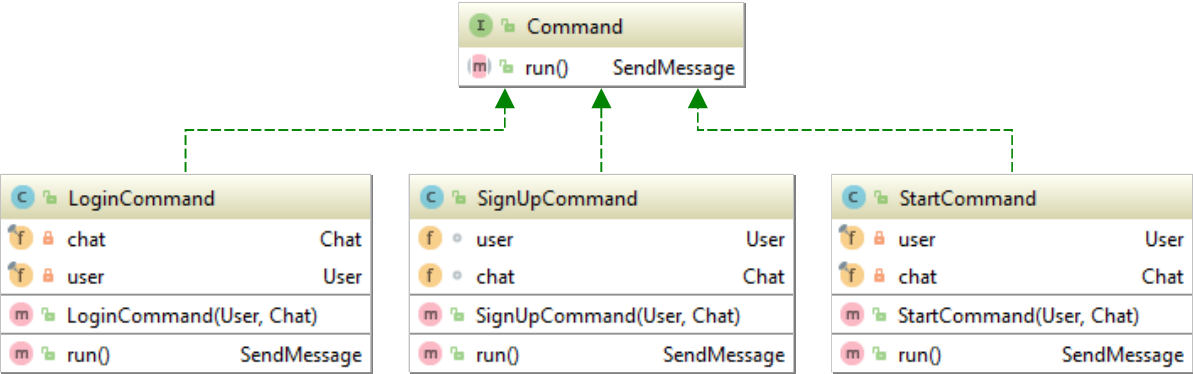
C	o	SqlQueryExecutor	
m	o	select(String, QueryParameters)	ResultSet
m	o	select(String, QueryParameters, List<String>)	ResultSet
m	o	insert(String, QueryParameters)	void
m	o	deleteOne(String, QueryParameters)	void
m	o	closeConnection()	void

C	o	QueryParameters	
m	o	add(String, String)	QueryParameters
m	o	add(String, int)	QueryParameters
m	o	add(String, boolean)	QueryParameters
m	o	add(String, List<?>)	QueryParameters
m	o	add(String, LocalDate)	QueryParameters
m	o	add(String)	QueryParameters
m	o	isEmpty()	boolean
m	o	toInsertParameters()	String
m	o	getKeys()	String
m	o	toWhereCondition()	String
m	o	toSqlEntity(Integer, Object)	String

C	o	SqlStorage	
m	o	getInstance()	SqlStorage
m	o	connect(String, String, String)	void
m	o	findUsers(QueryParameters)	List<User>
m	o	getUser(int)	Optional<User>
m	o	findBooks(QueryParameters)	List<Book>
m	o	findJournals(QueryParameters)	List<JournalIssue>
m	o	getBook(int)	Optional<Book>
m	o	getJournal(int)	Optional<JournalIssue>
m	o	findArticles(QueryParameters)	List<JournalArticle>
m	o	getArticle(int)	Optional<JournalArticle>
m	o	findAvMaterials(QueryParameters)	List<AvMaterial>
m	o	getAvMaterial(int)	Optional<AvMaterial>
m	o	getNumOfCheckouts(int)	int
m	o	getCheckoutRecordsFor(int)	List<CheckoutRecord>
m	o	addUser(User)	User
m	o	addBook(BookFactory)	Book
m	o	addJournalArticle(JournalArticleFactory)	JournalArticle
m	o	addJournal(JournalIssueFactory)	JournalIssue
m	o	addAvMaterial(AvMaterialFactory)	AvMaterial
m	o	addCheckoutRecord(CheckoutRecord)	void
m	o	removeUser(int)	void
m	o	removeBook(int)	void
m	o	removeJournalArticle(int)	void
m	o	removeJournal(int)	void
m	o	removeAvMaterial(int)	void
m	o	removeCheckoutRecord(CheckoutRecord)	void
m	o	closeConnection()	void
m	o	getItem(String, int, ItemSerializer<T>)	Optional<T>
m	o	findItems(String, QueryParameters, ItemSerializer<T>)	List<T>

## I Storage

(m) 	findUsers(QueryParameters)	List<User>
(m) 	getUser(int)	Optional<User>
(m) 	findBooks(QueryParameters)	List<Book>
(m) 	getBook(int)	Optional<Book>
(m) 	findArticles(QueryParameters)	List<JournalArticle>
(m) 	getArticle(int)	Optional<JournalArticle>
(m) 	findAvMaterials(QueryParameters)	List<AvMaterial>
(m) 	getAvMaterial(int)	Optional<AvMaterial>
(m) 	findJournals(QueryParameters)	List<JournalIssue>
(m) 	getJournal(int)	Optional<JournalIssue>
(m) 	getNumOfCheckouts(int)	int
(m) 	getCheckoutRecordsFor(int)	List<CheckoutRecord>
(m) 	addUser(User)	User
(m) 	addBook(BookFactory)	Book
(m) 	addJournalArticle(JournalArticleFactory)	JournalArticle
(m) 	addJournal(JournalIssueFactory)	JournalIssue
(m) 	addAvMaterial(AvMaterialFactory)	AvMaterial
(m) 	addCheckoutRecord(CheckoutRecord)	void
(m) 	removeUser(int)	void
(m) 	removeBook(int)	void
(m) 	removeJournalArticle(int)	void
(m) 	removeJournal(int)	void
(m) 	removeAvMaterial(int)	void
(m) 	removeCheckoutRecord(CheckoutRecord)	void





















## Bot




f	◉	username	String
f	◉	password	String
f	◉	previous	HashMap<Long, String>
f	◉	signUpName	String
f	◉	signUpEmail	String
f	◉	signUpPhone	String
f	◉	signUpPassword	String

m	◉	onUpdateReceived(Update)	void
m	◉	showMainMenuKeyboard(Long, boolean)	void
m	◉	sendMessage(Long, String)	void
m	◉	createAccount(String, String, String)	void
m	◉	setInlineKeyBoard(Long, String, List<String>)	void
m	◉	showCRUDkeyboard(Long)	void

p	botUsername	String
p	botToken	String

	StorageTest	
	 storage	SqlStorage
	StorageTest()	
	 findUsers()	void
	 findBooks()	void
	 addBook()	void
	 addJournal()	void
	 addJournalArticle()	void
	 finalize()	void

	BookingSystemTest	
	 createSystem()	void

	ReturnSystemTest	
	 createSystem()	void