



REPORT

Final project: UNIVERSITY SYSTEM

Course: OBJECT - ORIENTED PROGRAMMING AND DESIGN

Checked by: Shamoi Pakizar

Done by: Seitbekov Sanzhar

Akhmetkan Azhar

Makhmut Abulkhair

2021, Almaty

CONTENTS

Introduction	2
Modeling process	3
UML diagrams	3
Development and programming	5
Menu	6
Database	8
Data Serialization	9
Admin and Manager	10
Patterns	11
Documentation	13
Conclusion	14

Introduction

The main purpose was to recreate the system of the Kazakh British Technical University (KBTU). Using the knowledge gained in the Object - Oriented Programming and design(OOP) course and the experience gained in the Java programming language, which were honed with the implementation of practical tasks for the entire course, were the main assistants in the creation of this final project. Creation university system includes main features of Object - Oriented Programming(OOP): inheritance, polymorphism, encapsulation, classes and objects, patterns, serialization etc.

Development environments



Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.



IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains.



GenMyModel is a modeling platform in the cloud for software architects and developers. It enables to create Flowchart diagrams and UML diagrams with ease for more efficient business processes and models.



GitHub is a code hosting platform for version control and collaboration. It lets people work together on projects from anywhere.



Discord is a VoIP, instant messaging and digital distribution platform designed for creating communities. Users communicate with voice calls, video calls, text messaging, media and files.

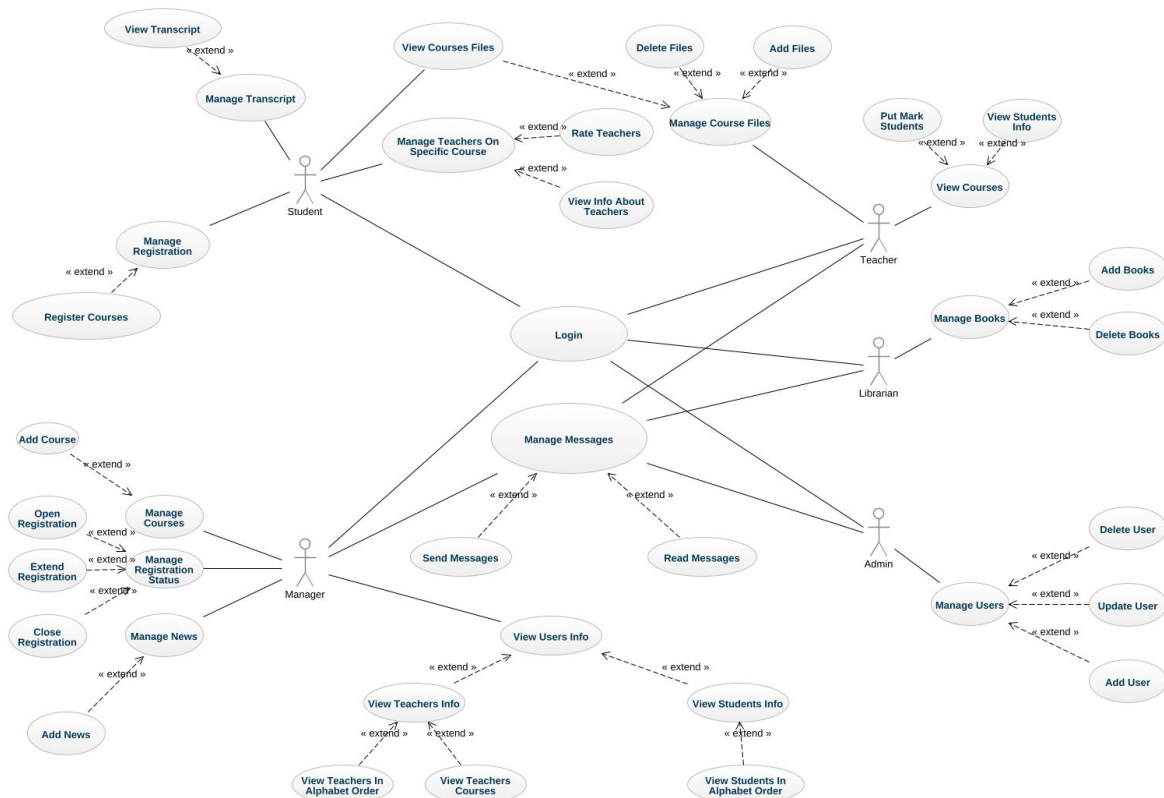
Modeling process

UML diagrams

The first step in creating a project is to study the existing system and initially model the system. The system was simulated using the modeling platform “GenMyModel”. Modeling is characterized by the creation of UML diagrams, where we modeled a Class Diagram as a structural diagram and Use Case Diagram as a behavioural diagram.

Use Case Diagram:

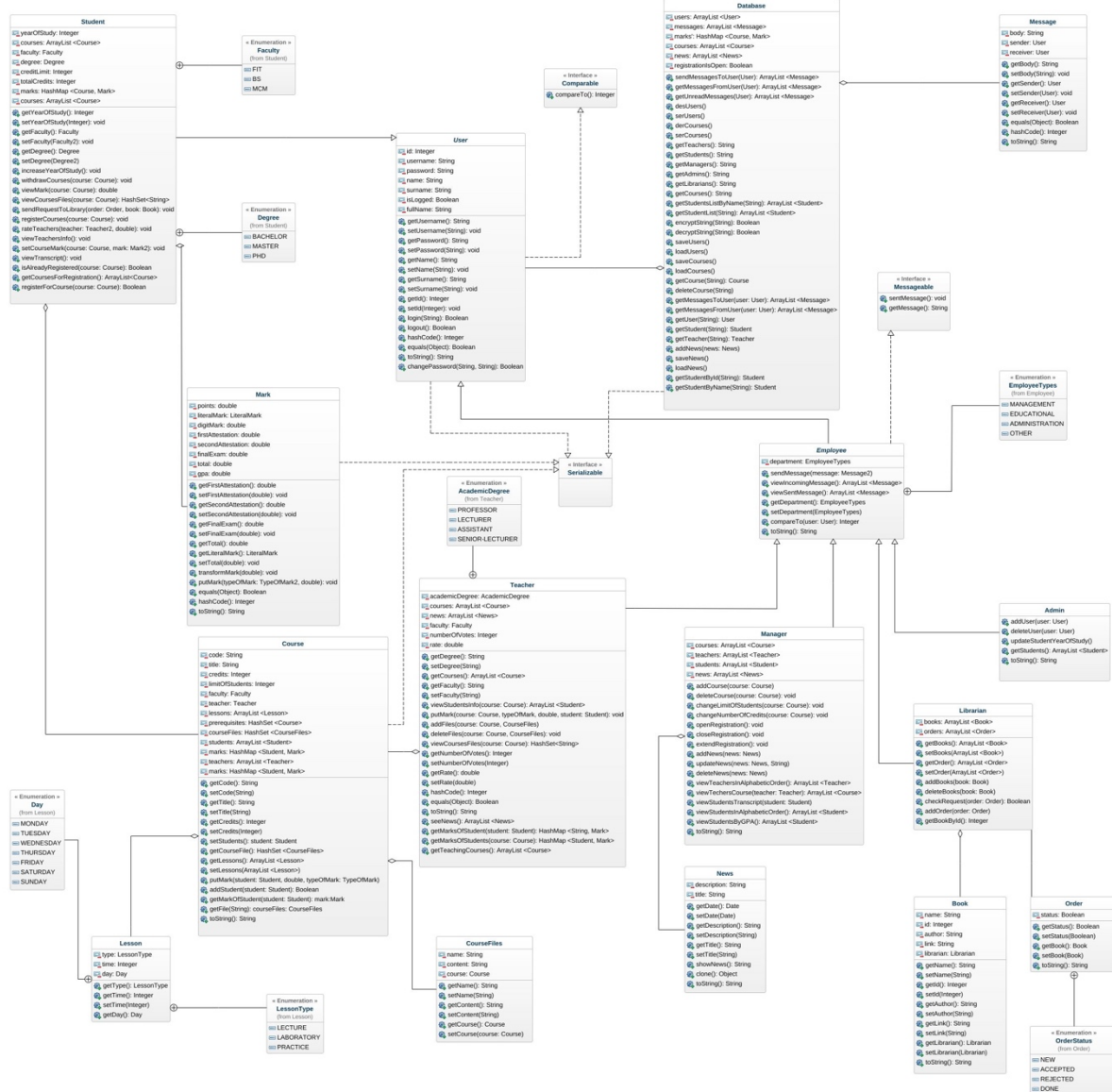
We started with creating a Use Case Diagram, which gives a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. Our actors: Student, Teacher, Manager, Librarian and Admin. This is the final version of Use Case Diagram, which changed, according to the process of development (Picture 1).



Picture 1.

Class Diagram:

We started creating a Class Diagram after having an idea of what the system would do. Class Diagram is the main building block of our final project, which shows the classes in a system, attributes, and operations of each class and the relationship between each class. This is the final version of Class Diagram (Picture 2).



Picture 2.

The modeling platform “GenMyModel” made it possible to generate a class diagram, which made the developing and programming work easier. The point of development and programming is discussed in detail in the next part.

Development and programming

The final project has only main classes, which characterized the university system.

Main classes:

User	Parent class, which represents optional information about users (username, name, surname)
Employee	Represents Employee and parent class for main employees in university
Admin	Represents Admin's information and account
Manager	Represents Manager's information and account
Teacher	Represents Teacher's information and account
Student	Represents Student's information and account
Librarian	Represents Librarian's information and account
Course	Represents Course information and methods
Database	Represents storage information in the system: users, courses, marks and etc.
Mark	Represents Mark's information and methods

We also have additional classes to describe the function of the classes above: Book, CourseFiles, Lesson, Message, News and Order.

Enumerations:

- AcademicDegree - represents teacher's academic degree
- Degree - represents student's study program
- EmployeeTypes - represents department of employee

- Faculty - represents faculty of student and teacher
- OrderStatus - represents status of request to Librarian
- TypeOfMark - represents type of mark

Menu

We created extra classes to make relationships between main classes and to make the system more usable and understandable for users. There are main Menu and separate classes AdminMenu, ManagerMenu, TeacherMenu, StudentMenu and LibrarianMenu, having connection to main Menu.

These classes show the outside of the project and redirect to classes based on the user's choice.

```
public static void startSystem() {
    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        while (true) {

            String welcomePage = ""

            | _ | \ \ | _ | _ \ \ / \ \ | \ \ | _ _ _ | _ | | | | |
            | | \ \ | | | | | ) | / _ \ \ | \ \ | _ | |
            | | \ \ | | | | _ < / _ _ \ \ | \ \ | _ _ | |
            | _ _ | \ \ | | | | \ \ \ \ / \ \ \ \ | \ \ | _ _ _ | |

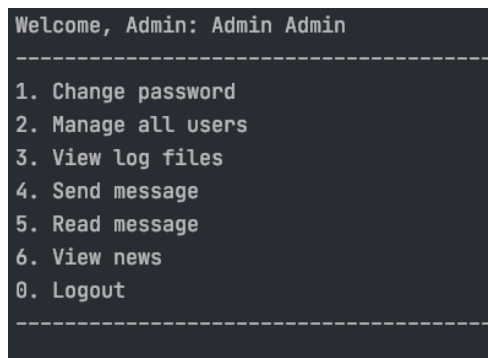
            1. Login
            2. Exit"";

            System.out.println(welcomePage);
            String input = reader.readLine();
            if (input.equals("1")) {
                User user = menuLogin(reader);
                if (user != null) {
                    while (user.getIsLogged()) {
                        if (user instanceof Admin) AdminMenu.menu(user, reader);
                        else if (user instanceof Student) StudentMenu.menu(user, reader);
                        else if (user instanceof Teacher) TeacherMenu.menu(user, reader);
                        else if (user instanceof Manager) ManagerMenu.menu(user, reader);
                        else if (user instanceof Librarian) LibrarianMenu.menu(user, reader);
                        else System.out.println("\nUsername or password incorrect. Please try aga
                    }
                }
            }
        }
    }
}
```

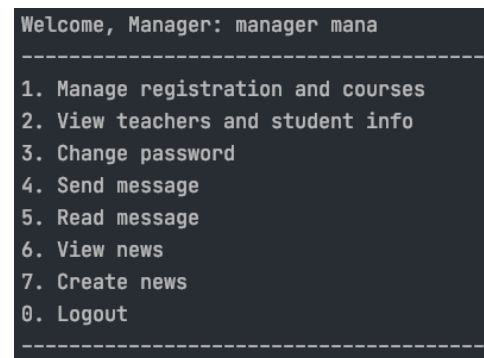
At first, the user will see the menu like below and after that input username and password. Checking correctness redirects user to the menu depending on who (Manager, Admin, Student, Teacher and Librarian).



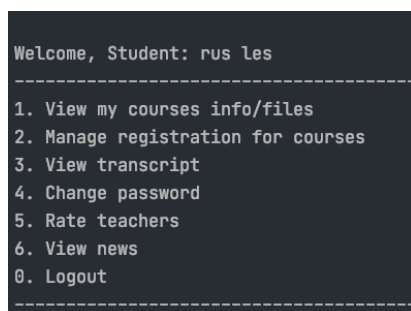
Main menu



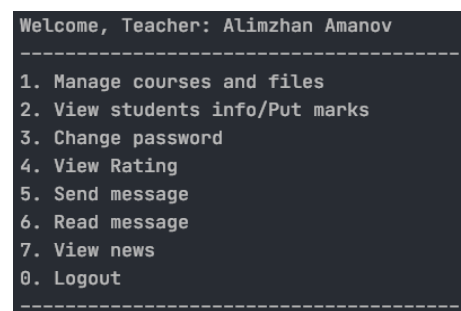
AdminMenu



ManagerMenu



StudentMenu



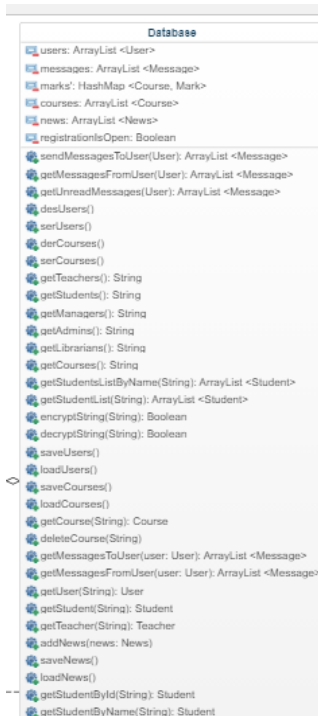
TeacherMenu

Database

The Database class is the main repository of our system, which stores all the data. Data on registered users, registered courses, teachers, students, grades data. All changes, such as creation, deletion, and modification, go through the Database class so that all system changes are visible to all users.

```
/**
 * Database is a storage of all objects: users, courses, etc.
 * Only exists in one copy.
 */
public class Database implements Serializable {

    /**
     * ArrayList which contains all users.
     */
    protected static ArrayList<User> users = new ArrayList<>();
    /**
     * ArrayList which contains all courses.
     */
    protected static ArrayList<Course> courses = new ArrayList<>();
    /**
     * HashMap which contains all marks of particular course.
     */
    protected static HashMap<Course, Mark> marks = new HashMap<>();
    /**
     * ArrayList which contains all news.
     */
    protected static ArrayList<News> news = new ArrayList<>();
```



There are the main methods to get users, information, list, messages, and news, saving and loading respectively. Storing property of Database helps users to make changes, modify information, store and all changes are made by the user will be visible to other users.

Data Serialization

We used serialization in the Database class. The main methods of saving and storing information needed serialize and deserialize information from Database class.

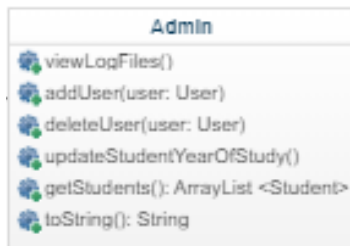
For example, serialization and deserialization of users and courses, where we write user to users and course to course and getting objects.

```
/**
 * Serialize all users.
 */
public static void saveUsers() {
    try (ObjectOutputStream oot = new ObjectOutputStream
        (new FileOutputStream( name: "users.dat"))) {
        oot.writeObject(users);
        oot.flush();
        encryptString(getKeyword());
    }
    catch (IOException e) {
        System.err.println("users.dat: IOException");
    }
}

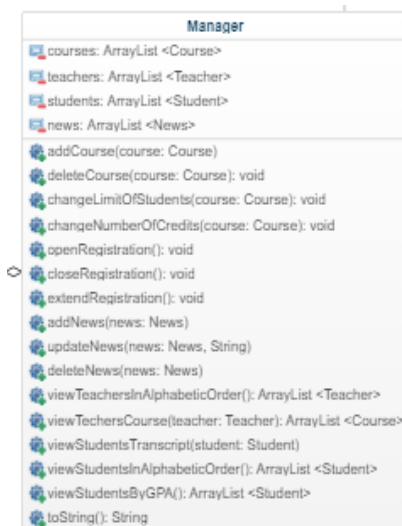
/**
 * Deserialize all users.
 */
```

Admin and Manager

Admin and Manager are the most important classes of our system. Admin is responsible for manipulating users' accounts and the Manager is those responsible for manipulating the educational section in university system.



Admin class is responsible for creating and deleting users in the university system. Admin gives access and register existing students and employees in university.



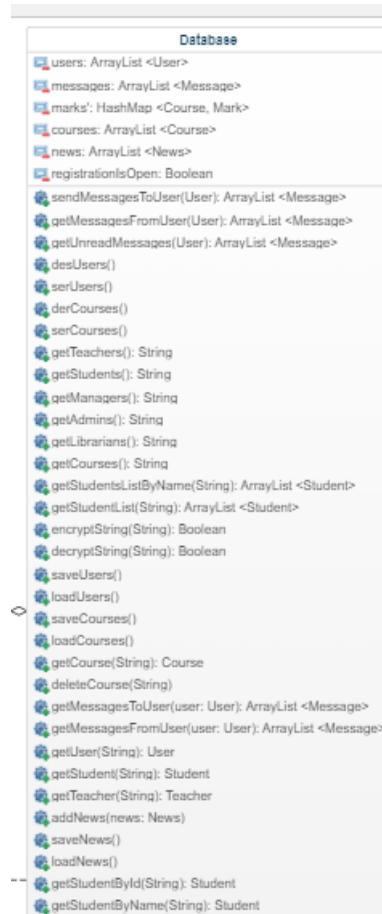
If the Admin class is responsible for accounts, then the Manager class is responsible for the educational part, consisting of managing courses, status of registration, managing news and view information about Student's mark, Teacher's courses.

Both classes have methods like equals() and toString(), inheriting all methods from User and Employee.

We also used collections and created AdminMenu and ManagerMenu with methods from Admin and Manager. They are responsible for redirecting users, who registered as manager and admin.

Patterns

Singleton pattern. We use a Singleton pattern to Database class, to ensure that its object is unique.



```
private static Database INSTANCE = null;

private Database() {}

public static Database getInstance() {
    if (INSTANCE == null) INSTANCE = new Database();
    return INSTANCE;
}

public static void serUsers() throws IOException{
    FileOutputStream fos = new FileOutputStream("users.out");
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(users);
    oos.close();
}

@SuppressWarnings("unchecked")
```

Facade pattern. We use a Facade pattern that is bound with a password. It performs the method of changing passwords. All users can change their password. And we added hash! Hooray!

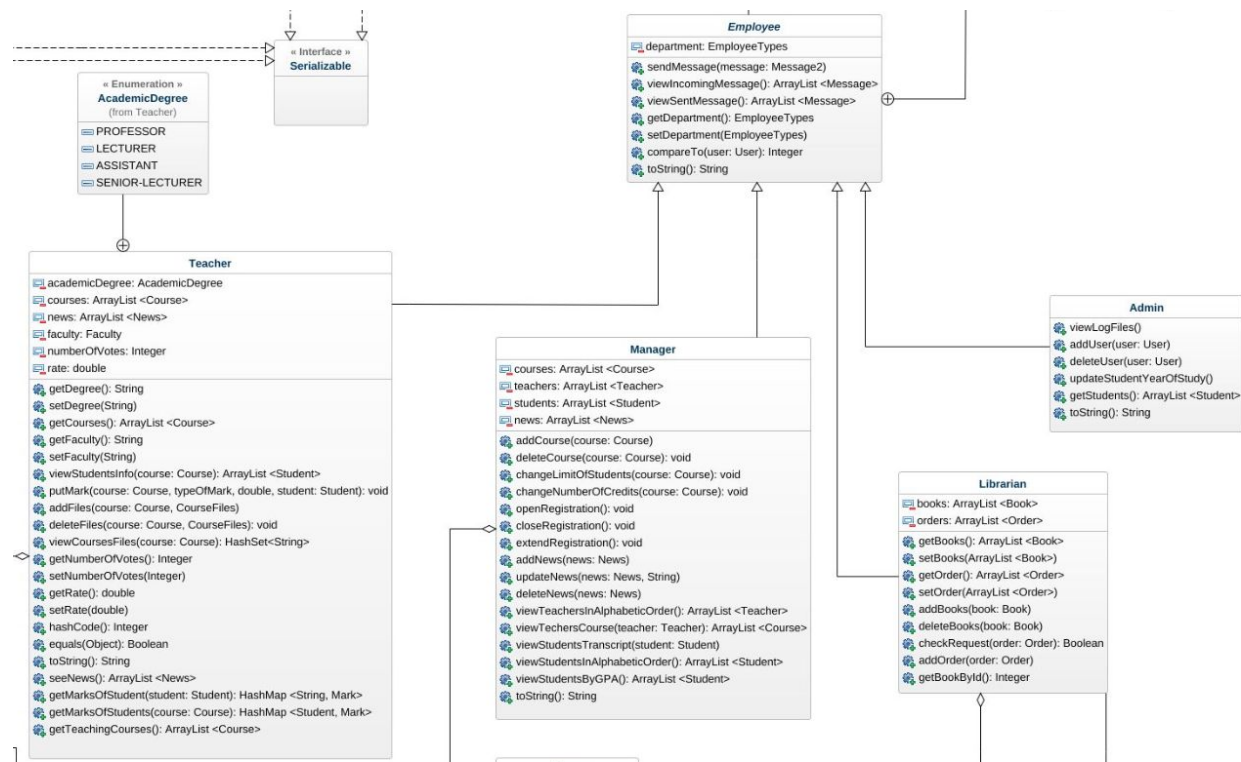
```

public static void showMenuForChangePassword(User user, BufferedReader reader) throws IOException {
    System.out.print("""
        Changing password.
        Please enter new password:\s
        """);
    String password = reader.readLine();
    System.out.println("Please repeat your new password: ");
    String repeatedPassword = reader.readLine();
    System.out.println(checkAndChangePassword(user, password, repeatedPassword));
}

public static String checkAndChangePassword(User user, String password, String repeatedPassword) {
    if (password.hashCode() == (repeatedPassword.hashCode())) {
        if (!(password.hashCode() == (user.getPassword().hashCode()))) {
            user.setPassword(password);
            return "Password successfully changed";
        }
        else {
            return "This password is used now. Choose another";
        }
    }
    return "Passwords does not matches each other";
}

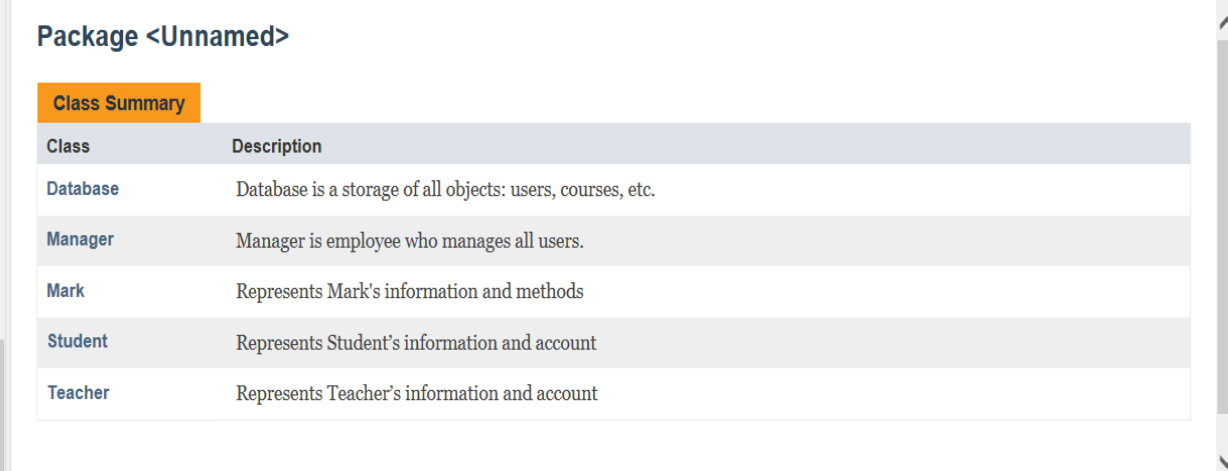
```

Template pattern. We have abstract classes User and Employee. They allow subclasses to override some steps of the algorithm without altering it altogether.



Documentation

We created documentation for 5 classes: Database, Manager, Teacher, Student and Mark. We wrote comments in the Eclipse platform for these classes and then automatically generated documentation. Our comments describe explanations of methods, fields, and classes.



The screenshot shows a web-based documentation interface for a package named "<Unnamed>". It features a "Class Summary" section with a table listing five classes: Database, Manager, Mark, Student, and Teacher. Each class has a corresponding description. The interface includes a sidebar on the left and a top navigation bar.

Package <Unnamed>	
Class Summary	
Class	Description
Database	Database is a storage of all objects: users, courses, etc.
Manager	Manager is employee who manages all users.
Mark	Represents Mark's information and methods
Student	Represents Student's information and account
Teacher	Represents Teacher's information and account

Conclusion

To summarize, this Intranet project was our final draft of the OOP course. The creation of the project required not only the acquired knowledge of the course or programming experience, but also the skills of working with a team, assigning tasks and time management.

The team captain regulated the whole process and assigned tasks related to the project. Discussion and project management took place in the Discord for meetings and GitHub platform for having access to project.

We would like to express our enormous gratitude for taking the OOP course. We realized that this course is important in the world of programming, in problem solving. We have mastered the most important knowledge and strengthened it by doing practical tasks, the project itself. Special thanks to Shamoï Pakizar, who is the professor and lecturer of this course. You have made this course not only easy to understand, but also fun. Thank you for developing our technical and human skills.