KAZAKH BRITISH TECHNICAL UNIVERSITY

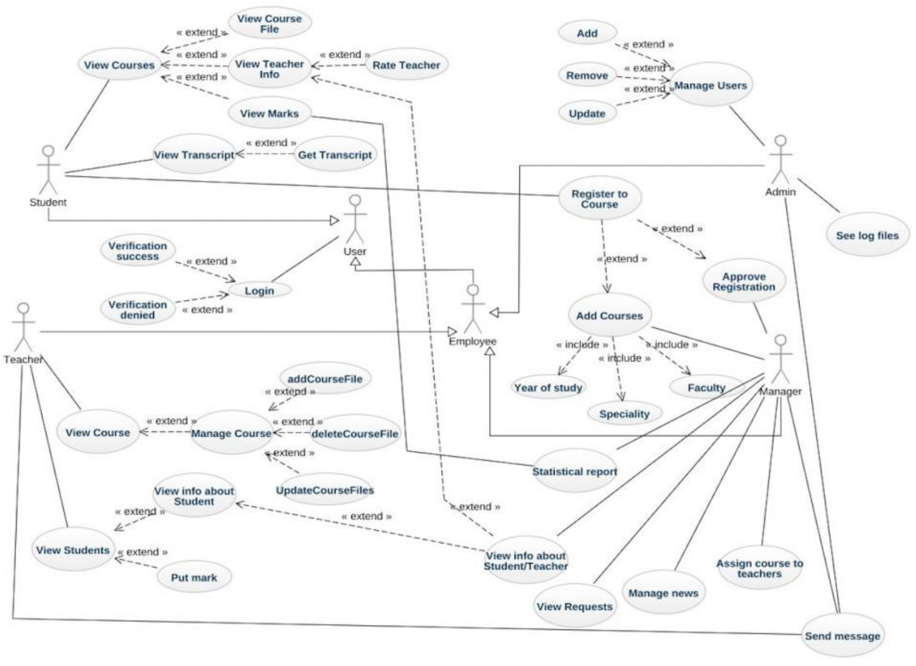
REPORT

OBJECT-ORIENTED PROGRAMMING AND DESIGN

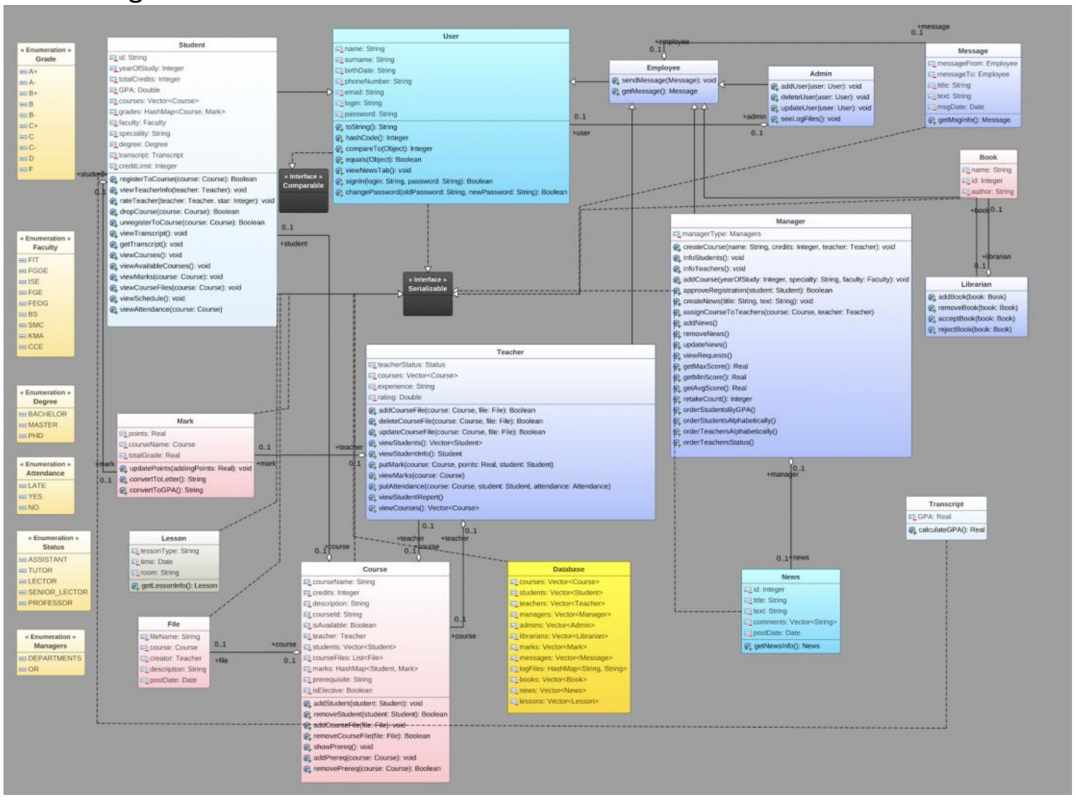
Sarsengaliyev Zhaisan
Tolegen Ernazar
Nurakhmet Eldar

UML

USE CASE diagram



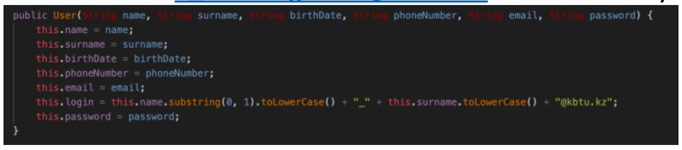
CLASS diagram



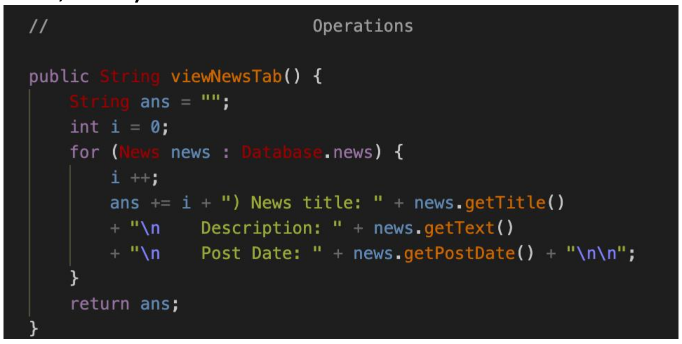
Code implementation

*User class

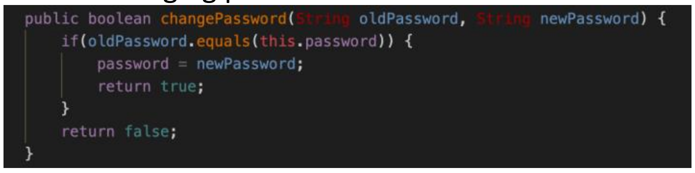
Firstly, I want to start describing from User class, but *I will not describe methods that are too simple in order to avoid super huge and boring report*.
So, in User class, in authentication, the program recognizes by login which user is logged in. Here it converts name and surname to kbtu login. For Example, Zhaisan Sarsengaliev converts like z_sarsengaliev@kbtu.kz automatically.


```java
public User(String name, String surname, String birthDate, String phoneNumber, String email, String password) {
    this.name = name;
    this.surname = surname;
    this.birthDate = birthDate;
    this.phoneNumber = phoneNumber;
    this.email = email;
    this.login = this.name.substring(0, 1).toLowerCase() + "_" + this.surname.toLowerCase() + "@kbtu.kz";
    this.password = password;
}
```

Also, every user can see news

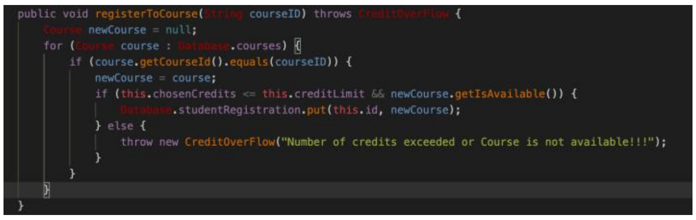

```java
//                           Operations

public String viewNewsTab() {
    String ans = "";
    int i = 0;
    for (News news : Database.news) {
        i ++;
        ans += i + ") News title: " + news.getTitle()
        + "\n    Description: " + news.getText()
        + "\n    Post Date: " + news.getPostDate() + "\n\n";
    }
    return ans;
}
```

Here is changing password

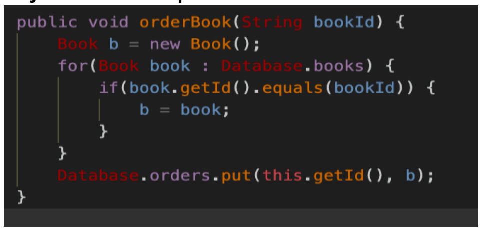

```java
public boolean changePassword(String oldPassword, String newPassword) {
    if(oldPassword.equals(this.password)) {
        password = newPassword;
        return true;
    }
    return false;
}
```

*Student class

Here registerToCourse method which shows CreditOverFlow exception when the selected credits exceed the credit limit.

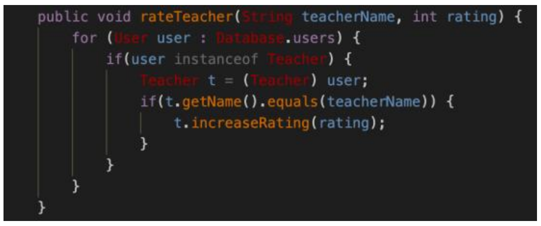

```java
public void registerToCourse(String courseID) throws CreditOverFlow {
    Course newCourse = null;
    for (Course course : Database.courses) {
        if (course.getCourseId().equals(courseID)) {
            newCourse = course;
            if (this.chosenCredits <= this.creditLimit && newCourse.getIsAvailable()) {
                Database.studentRegistration.put(this.id, newCourse);
            } else {
                throw new CreditOverFlow("Number of credits exceeded or Course is not available!!!");
            }
        }
    }
}
```

In orderBook method student can send request to librarian where librarian can accept or reject this request.



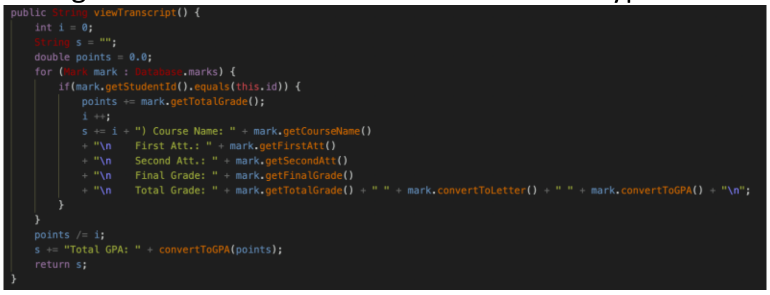
```java
public void orderBook(String bookId) {
    Book b = new Book();
    for(Book book : Database.books) {
        if(book.getId().equals(bookId)) {
            b = book;
        }
    }
    Database.orders.put(this.getId(), b);
}
```

In rateTeacher method student can rate teacher on a five-point scale.



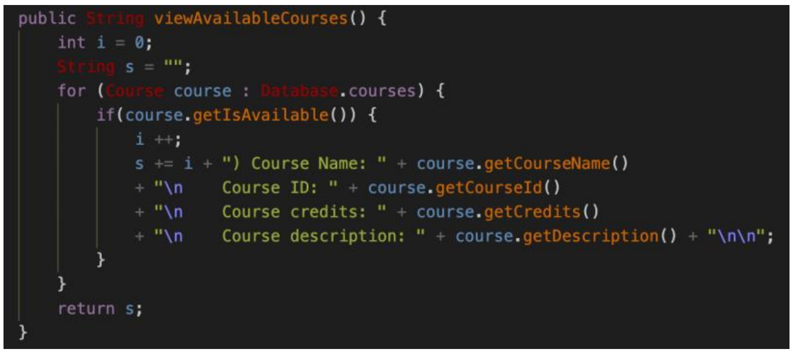
```java
public void rateTeacher(String teacherName, int rating) {
    for (User user : Database.users) {
        if(user instanceof Teacher) {
            Teacher t = (Teacher) user;
            if(t.getName().equals(teacherName)) {
                t.increaseRating(rating);
            }
        }
    }
}
```

Here student can view transcript with grades in first, second attestation, final grade and total grade with conversion to letter and GPA type.



```java
public String viewTranscript() {
    int i = 0;
    String s = "";
    double points = 0.0;
    for (Mark mark : Database.marks) {
        if(mark.getStudentId().equals(this.id)) {
            points += mark.getTotalGrade();
            i ++;
            s += i + ") Course Name: " + mark.getCourseName()
            + "\n    First Att.: " + mark.getFirstAtt()
            + "\n    Second Att.: " + mark.getSecondAtt()
            + "\n    Final Grade: " + mark.getFinalGrade()
            + "\n    Total Grade: " + mark.getTotalGrade() + " " + mark.convertToLetter() + " " + mark.convertToGPA() + "\n";
        }
    }
    points /= i;
    s += "Total GPA: " + convertToGPA(points);
    return s;
}
```

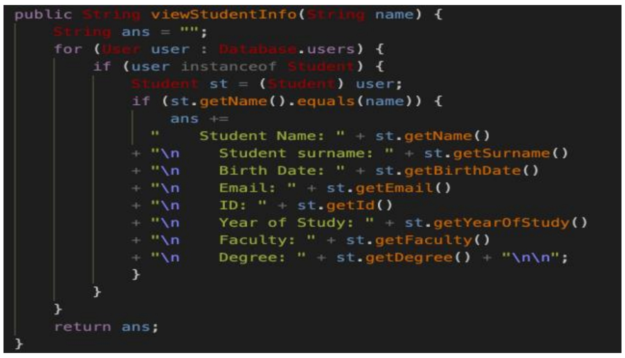
Here student can see all available courses with number of credits and description.



```java
public String viewAvailableCourses() {
    int i = 0;
    String s = "";
    for (Course course : Database.courses) {
        if(course.getIsAvailable()) {
            i ++;
            s += i + ") Course Name: " + course.getCourseName()
            + "\n    Course ID: " + course.getCourseId()
            + "\n    Course credits: " + course.getCredits()
            + "\n    Course description: " + course.getDescription() + "\n\n";
        }
    }
    return s;
}
```

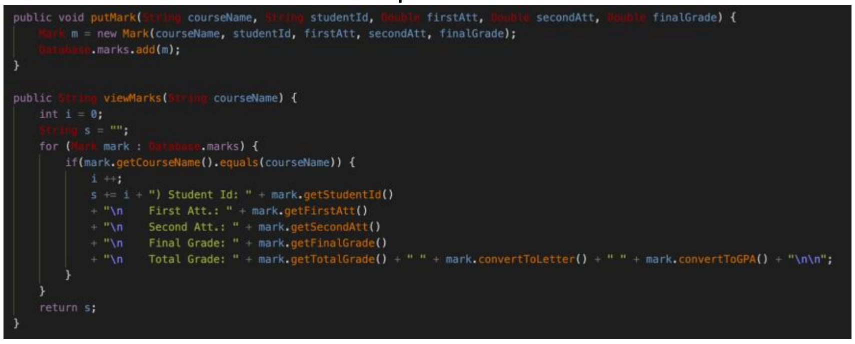
***Teacher class**

In this method teacher can view detailed info about students.



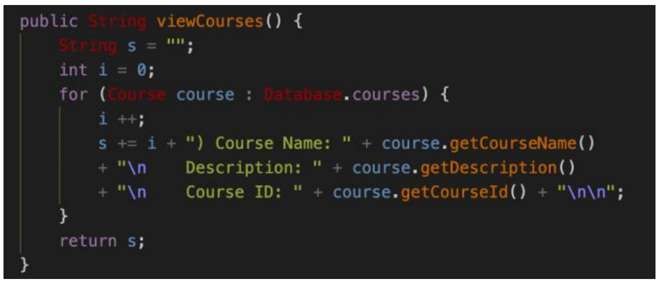
```java
public String viewStudentInfo(String name) {
    String ans = "";
    for (User user : Database.users) {
        if (user instanceof Student) {
            Student st = (Student) user;
            if (st.getName().equals(name)) {
                ans +=
                "    Student Name: " + st.getName()
                + "\n    Student surname: " + st.getSurname()
                + "\n    Birth Date: " + st.getBirthDate()
                + "\n    Email: " + st.getEmail()
                + "\n    ID: " + st.getId()
                + "\n    Year of Study: " + st.getYearOfStudy()
                + "\n    Faculty: " + st.getFaculty()
                + "\n    Degree: " + st.getDegree() + "\n\n";
            }
        }
    }
    return ans;
}
```

In these methods teacher can put and view marks.



```java
public void putMark(String courseName, String studentId, Double firstAtt, Double secondAtt, Double finalGrade) {
    Mark m = new Mark(courseName, studentId, firstAtt, secondAtt, finalGrade);
    Database.marks.add(m);
}

public String viewMarks(String courseName) {
    int i = 0;
    String s = "";
    for (Mark mark : Database.marks) {
        if(mark.getCourseName().equals(courseName)) {
            i ++;
            s += i + ") Student Id: " + mark.getStudentId()
            + "\n    First Att.: " + mark.getFirstAtt()
            + "\n    Second Att.: " + mark.getSecondAtt()
            + "\n    Final Grade: " + mark.getFinalGrade()
            + "\n    Total Grade: " + mark.getTotalGrade() + " " + mark.convertToLetter() + " " + mark.convertToGPA() + "\n\n";
        }
    }
    return s;
}
```

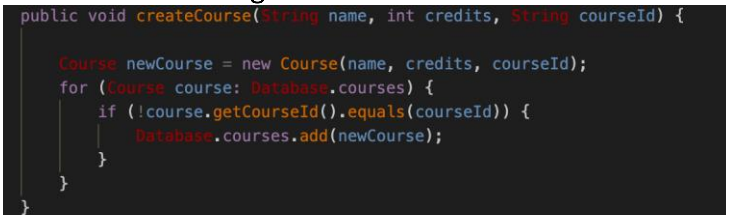
Method to view courses.



```java
public String viewCourses() {
    String s = "";
    int i = 0;
    for (Course course : Database.courses) {
        i ++;
        s += i + ") Course Name: " + course.getCourseName()
        + "\n    Description: " + course.getDescription()
        + "\n    Course ID: " + course.getCourseId() + "\n\n";
    }
    return s;
}
```

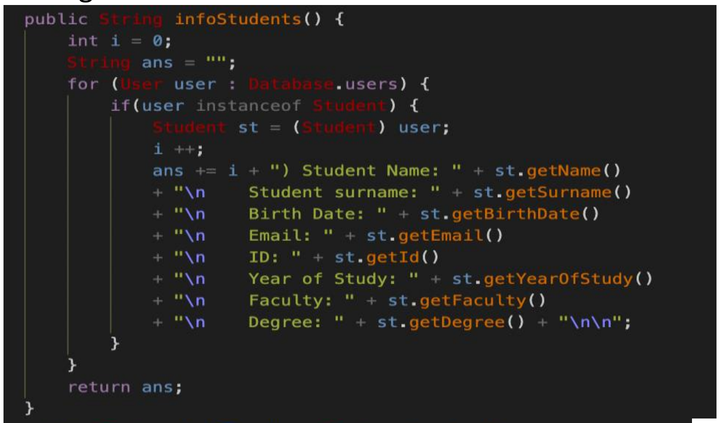
***Manager class**

Method for creating new course.

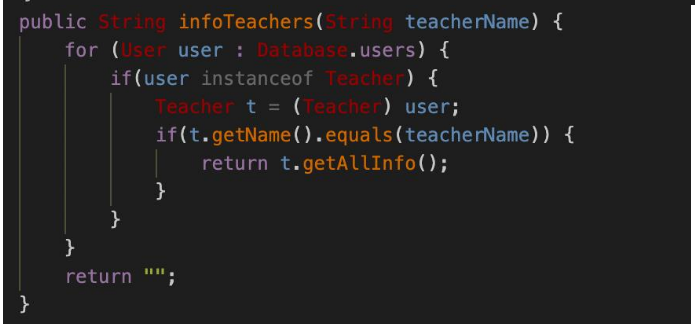


```java
public void createCourse(String name, int credits, String courseId) {

    Course newCourse = new Course(name, credits, courseId);
    for (Course course: Database.courses) {
        if (!course.getCourseId().equals(courseId)) {
            Database.courses.add(newCourse);
        }
    }
}
```

Manager can see information about students and teachers.

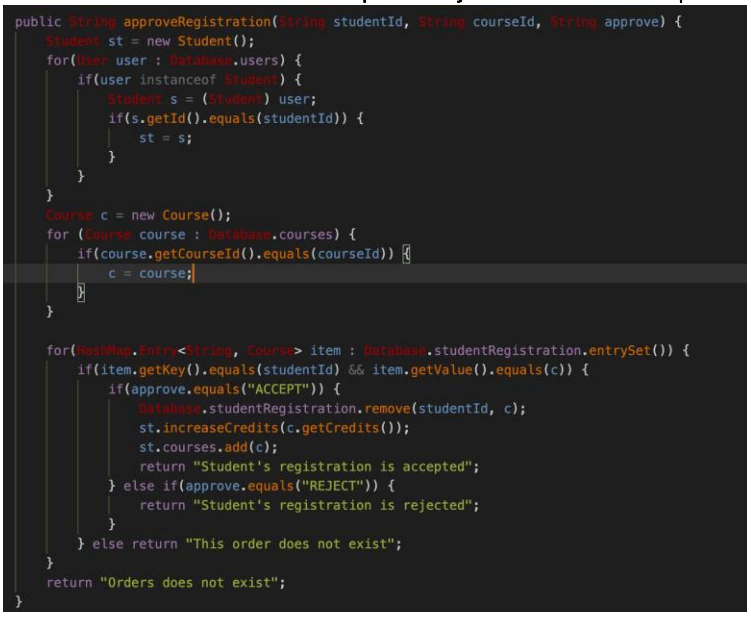


```java
public String infoStudents() {
    int i = 0;
    String ans = "";
    for (User user : Database.users) {
        if(user instanceof Student) {
            Student st = (Student) user;
            i ++;
            ans += i + ") Student Name: " + st.getName()
            + "\n    Student surname: " + st.getSurname()
            + "\n    Birth Date: " + st.getBirthDate()
            + "\n    Email: " + st.getEmail()
            + "\n    ID: " + st.getId()
            + "\n    Year of Study: " + st.getYearOfStudy()
            + "\n    Faculty: " + st.getFaculty()
            + "\n    Degree: " + st.getDegree() + "\n\n";
        }
    }
    return ans;
}
```



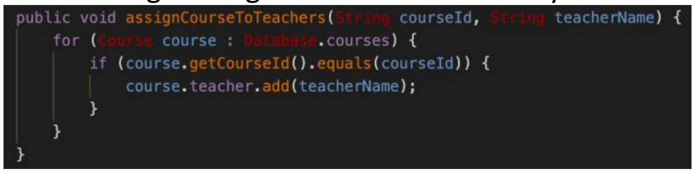
```java
public String infoTeachers(String teacherName) {
    for (User user : Database.users) {
        if(user instanceof Teacher) {
            Teacher t = (Teacher) user;
            if(t.getName().equals(teacherName)) {
                return t.getAllInfo();
            }
        }
    }
    return "";
}
```

Here manager can accept or reject student's request about registration. In more detail, from hashMap studentRegistration it gets as a key studentID and as a value it gets course and from this we decide to accept or reject student's request.



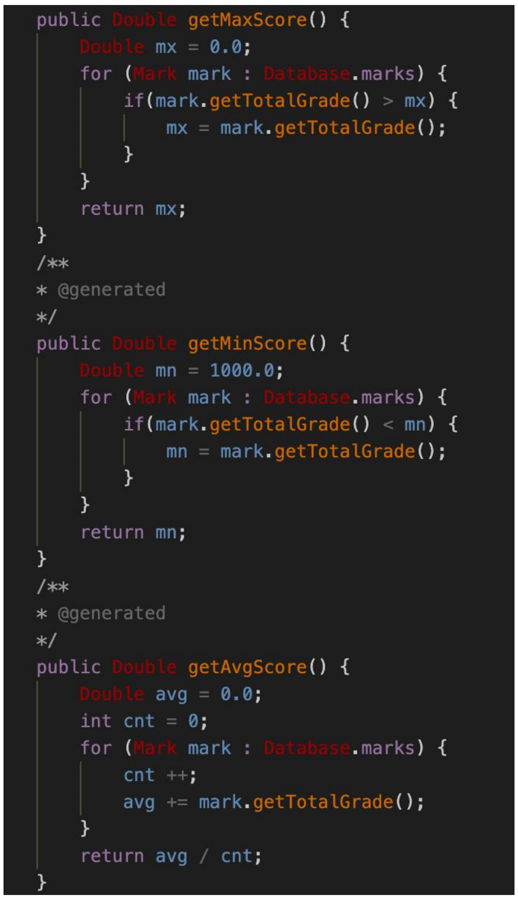
```java
public String approveRegistration(String studentId, String courseId, String approve) {
    Student st = new Student();
    for(User user : Database.users) {
        if(user instanceof Student) {
            Student s = (Student) user;
            if(s.getId().equals(studentId)) {
                st = s;
            }
        }
    }
    Course c = new Course();
    for (Course course : Database.courses) {
        if(course.getCourseId().equals(courseId)) {
            c = course;
        }
    }

    for(HashMap.Entry<String, Course> item : Database.studentRegistration.entrySet()) {
        if(item.getKey().equals(studentId) && item.getValue().equals(c)) {
            if(approve.equals("ACCEPT")) {
                Database.studentRegistration.remove(studentId, c);
                st.increaseCredits(c.getCredits());
                st.courses.add(c);
                return "Student's registration is accepted";
            } else if(approve.equals("REJECT")) {
                return "Student's registration is rejected";
            }
        } else return "This order does not exist";
    }
    return "Orders does not exist";
}
```

Here manages assigns course to teacher by courseID

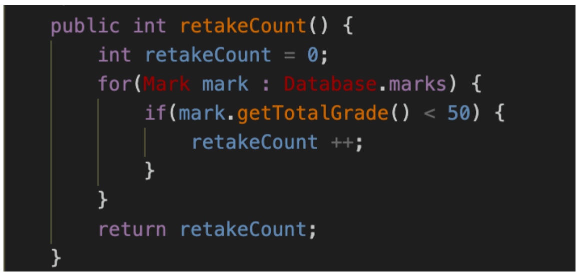


```java
public void assignCourseToTeachers(String courseId, String teacherName) {
    for (Course course : Database.courses) {
        if (course.getCourseId().equals(courseId)) {
            course.teacher.add(teacherName);
        }
    }
}
```

Getting info as academic performance with maximum, minimum, average grades, number of retakes.

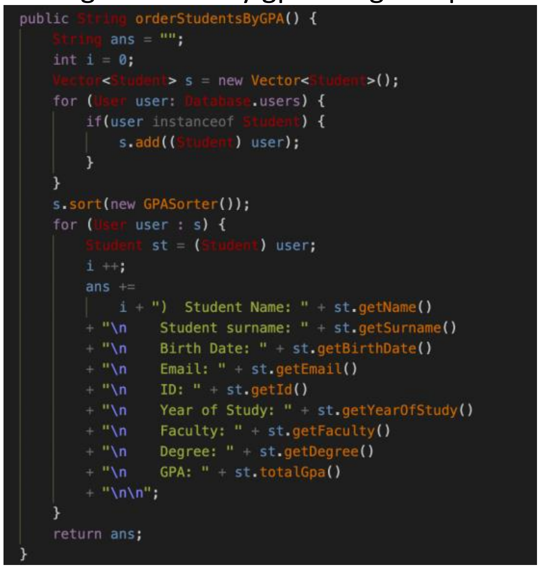


```java
public Double getMaxScore() {
    Double mx = 0.0;
    for (Mark mark : Database.marks) {
        if(mark.getTotalGrade() > mx) {
            mx = mark.getTotalGrade();
        }
    }
    return mx;
}
/**
* @generated
*/
public Double getMinScore() {
    Double mn = 1000.0;
    for (Mark mark : Database.marks) {
        if(mark.getTotalGrade() < mn) {
            mn = mark.getTotalGrade();
        }
    }
    return mn;
}
/**
* @generated
*/
public Double getAvgScore() {
    Double avg = 0.0;
    int cnt = 0;
    for (Mark mark : Database.marks) {
        cnt ++;
        avg += mark.getTotalGrade();
    }
    return avg / cnt;
}
```

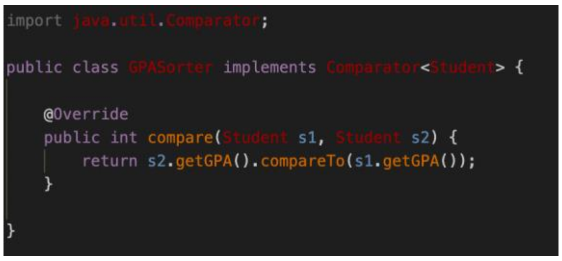


```java
public int retakeCount() {
    int retakeCount = 0;
    for(Mark mark : Database.marks) {
        if(mark.getTotalGrade() < 50) {
            retakeCount ++;
        }
    }
    return retakeCount;
}
```

Sorting students by gpa using comparator

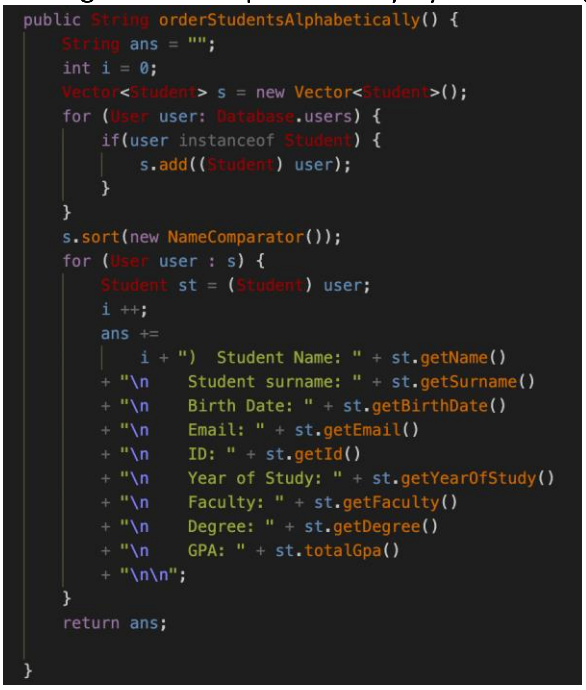


```java
public String orderStudentsByGPA() {
    String ans = "";
    int i = 0;
    Vector<Student> s = new Vector<Student>();
    for (User user: Database.users) {
        if(user instanceof Student) {
            s.add((Student) user);
        }
    }
    s.sort(new GPASorter());
    for (User user : s) {
        Student st = (Student) user;
        i ++;
        ans +=
            i + ")  Student Name: " + st.getName()
        + "\n    Student surname: " + st.getSurname()
        + "\n    Birth Date: " + st.getBirthDate()
        + "\n    Email: " + st.getEmail()
        + "\n    ID: " + st.getId()
        + "\n    Year of Study: " + st.getYearOfStudy()
        + "\n    Faculty: " + st.getFaculty()
        + "\n    Degree: " + st.getDegree()
        + "\n    GPA: " + st.totalGpa()
        + "\n\n";
    }
    return ans;
}
```

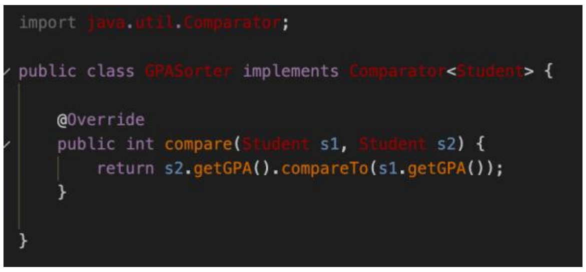


```java
import java.util.Comparator;

public class GPASorter implements Comparator<Student> {

    @Override
    public int compare(Student s1, Student s2) {
        return s2.getGPA().compareTo(s1.getGPA());
    }

}
```

Sorting students alphabetically by name using comparator

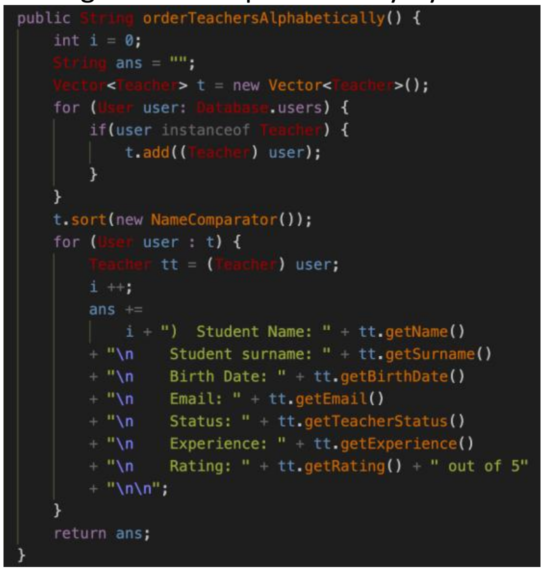


```java
public String orderStudentsAlphabetically() {
    String ans = "";
    int i = 0;
    Vector<Student> s = new Vector<Student>();
    for (User user: Database.users) {
        if(user instanceof Student) {
            s.add((Student) user);
        }
    }
    s.sort(new NameComparator());
    for (User user : s) {
        Student st = (Student) user;
        i ++;
        ans +=
            i + ")  Student Name: " + st.getName()
        + "\n    Student surname: " + st.getSurname()
        + "\n    Birth Date: " + st.getBirthDate()
        + "\n    Email: " + st.getEmail()
        + "\n    ID: " + st.getId()
        + "\n    Year of Study: " + st.getYearOfStudy()
        + "\n    Faculty: " + st.getFaculty()
        + "\n    Degree: " + st.getDegree()
        + "\n    GPA: " + st.totalGpa()
        + "\n\n";
    }
    return ans;

}
```



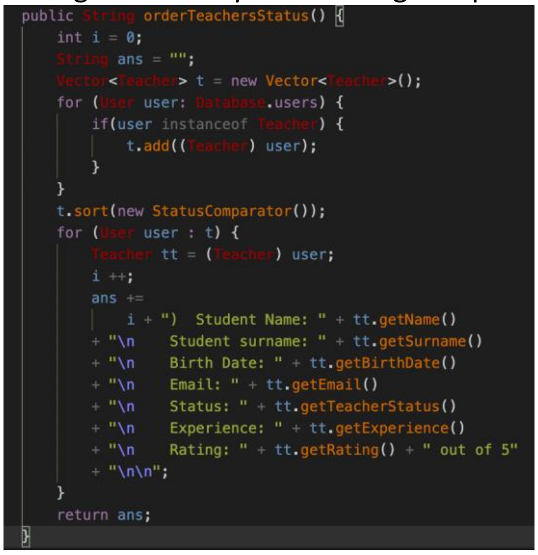
```java
import java.util.Comparator;

public class GPASorter implements Comparator<Student> {

    @Override
    public int compare(Student s1, Student s2) {
        return s2.getGPA().compareTo(s1.getGPA());
    }

}
```

Sorting teachers alphabetically by name using comparator



```java
public String orderTeachersAlphabetically() {
    int i = 0;
    String ans = "";
    Vector<Teacher> t = new Vector<Teacher>();
    for (User user: Database.users) {
        if(user instanceof Teacher) {
            t.add((Teacher) user);
        }
    }
    t.sort(new NameComparator());
    for (User user : t) {
        Teacher tt = (Teacher) user;
        i ++;
        ans +=
            i + ")  Student Name: " + tt.getName()
        + "\n    Student surname: " + tt.getSurname()
        + "\n    Birth Date: " + tt.getBirthDate()
        + "\n    Email: " + tt.getEmail()
        + "\n    Status: " + tt.getTeacherStatus()
        + "\n    Experience: " + tt.getExperience()
        + "\n    Rating: " + tt.getRating() + " out of 5"
        + "\n\n";
    }
    return ans;
}
```

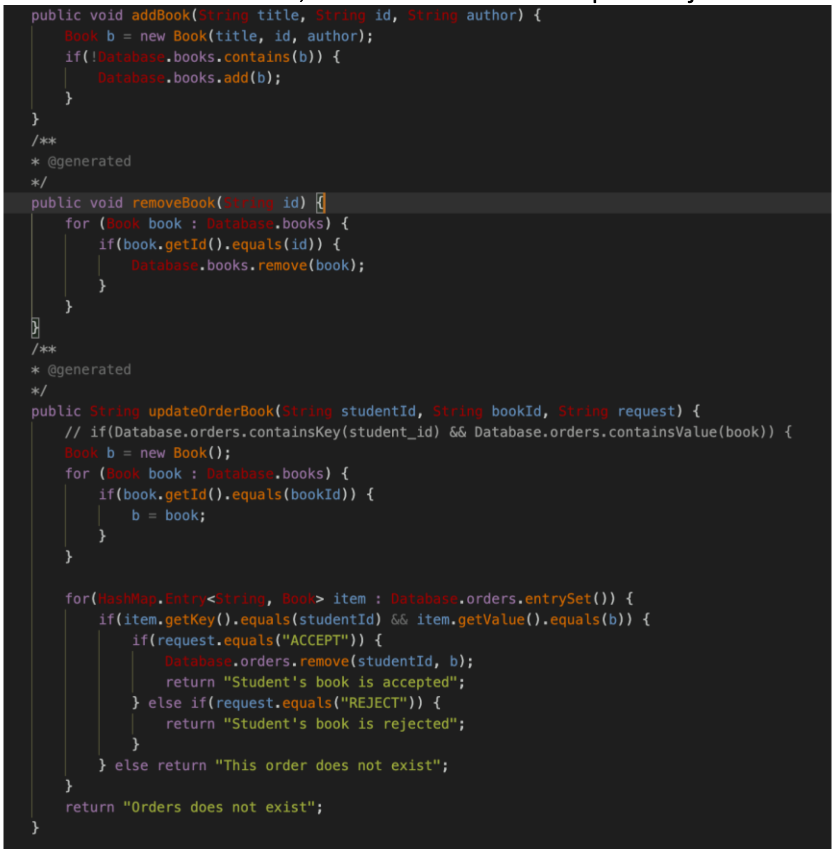
Sorting teachers by status using comparator



```
public String orderTeachersStatus() {
    int i = 0;
    String ans = "";
    Vector<Teacher> t = new Vector<Teacher>();
    for (User user: Database.users) {
        if(user instanceof Teacher) {
            t.add((Teacher) user);
        }
    }
    t.sort(new StatusComparator());
    for (User user : t) {
        Teacher tt = (Teacher) user;
        i ++;
        ans +=
            i + ")  Student Name: " + tt.getName()
        + "\n    Student surname: " + tt.getSurname()
        + "\n    Birth Date: " + tt.getBirthDate()
        + "\n    Email: " + tt.getEmail()
        + "\n    Status: " + tt.getTeacherStatus()
        + "\n    Experience: " + tt.getExperience()
        + "\n    Rating: " + tt.getRating() + " out of 5"
        + "\n\n";
    }
    return ans;
}
```

***Librarian class**

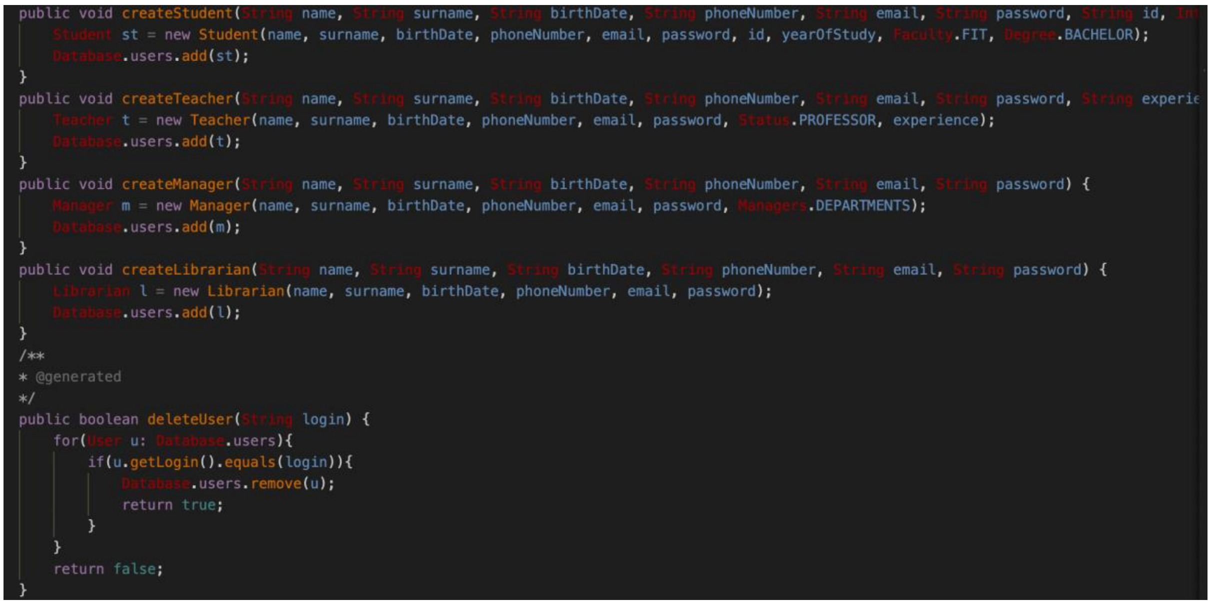
Librarian can add book, delete book and accept or reject student request to order book.



```
public void addBook(String title, String id, String author) {
    Book b = new Book(title, id, author);
    if(!Database.books.contains(b)) {
        Database.books.add(b);
    }
}
/**
* @generated
*/
public void removeBook(String id) {
    for (Book book : Database.books) {
        if(book.getId().equals(id)) {
            Database.books.remove(book);
        }
    }
}
/**
* @generated
*/
public String updateOrderBook(String studentId, String bookId, String request) {
    // if(Database.orders.containsKey(student_id) && Database.orders.containsValue(book)) {
    Book b = new Book();
    for (Book book : Database.books) {
        if(book.getId().equals(bookId)) {
            b = book;
        }
    }

    for(HashMap.Entry<String, Book> item : Database.orders.entrySet()) {
        if(item.getKey().equals(studentId) && item.getValue().equals(b)) {
            if(request.equals("ACCEPT")) {
                Database.orders.remove(studentId, b);
                return "Student's book is accepted";
            } else if(request.equals("REJECT")) {
                return "Student's book is rejected";
            }
        } else return "This order does not exist";
    }
    return "Orders does not exist";
}
```

***Admin class**

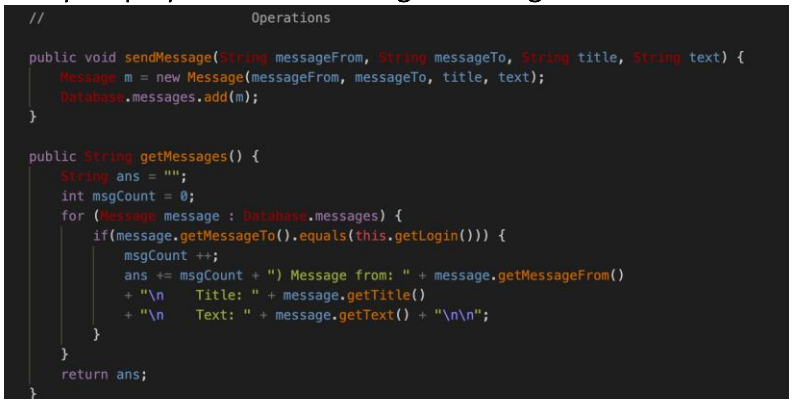
Admin can create users and delete them.



```
public void createStudent(String name, String surname, String birthDate, String phoneNumber, String email, String password, String id, In
    Student st = new Student(name, surname, birthDate, phoneNumber, email, password, id, yearOfStudy, Faculty.FIT, Degree.BACHELOR);
    Database.users.add(st);
}
public void createTeacher(String name, String surname, String birthDate, String phoneNumber, String email, String password, String experie
    Teacher t = new Teacher(name, surname, birthDate, phoneNumber, email, password, Status.PROFESSOR, experience);
    Database.users.add(t);
}
public void createManager(String name, String surname, String birthDate, String phoneNumber, String email, String password) {
    Manager m = new Manager(name, surname, birthDate, phoneNumber, email, password, Managers.DEPARTMENTS);
    Database.users.add(m);
}
public void createLibrarian(String name, String surname, String birthDate, String phoneNumber, String email, String password) {
    Librarian l = new Librarian(name, surname, birthDate, phoneNumber, email, password);
    Database.users.add(l);
}
/**
* @generated
*/
public boolean deleteUser(String login) {
    for(User u: Database.users){
        if(u.getLogin().equals(login)){
            Database.users.remove(u);
            return true;
        }
    }
    return false;
}
```

***Employee class**

Every employee can send and get messages.



```
//                          Operations

public void sendMessage(String messageFrom, String messageTo, String title, String text) {
    Message m = new Message(messageFrom, messageTo, title, text);
    Database.messages.add(m);
}

public String getMessages() {
    String ans = "";
    int msgCount = 0;
    for (Message message : Database.messages) {
        if(message.getMessageTo().equals(this.getLogin())) {
            msgCount ++;
            ans += msgCount + ") Message from: " + message.getMessageFrom()
            + "\n    Title: " + message.getTitle()
            + "\n    Text: " + message.getText() + "\n\n";
        }
    }
    return ans;
}
```

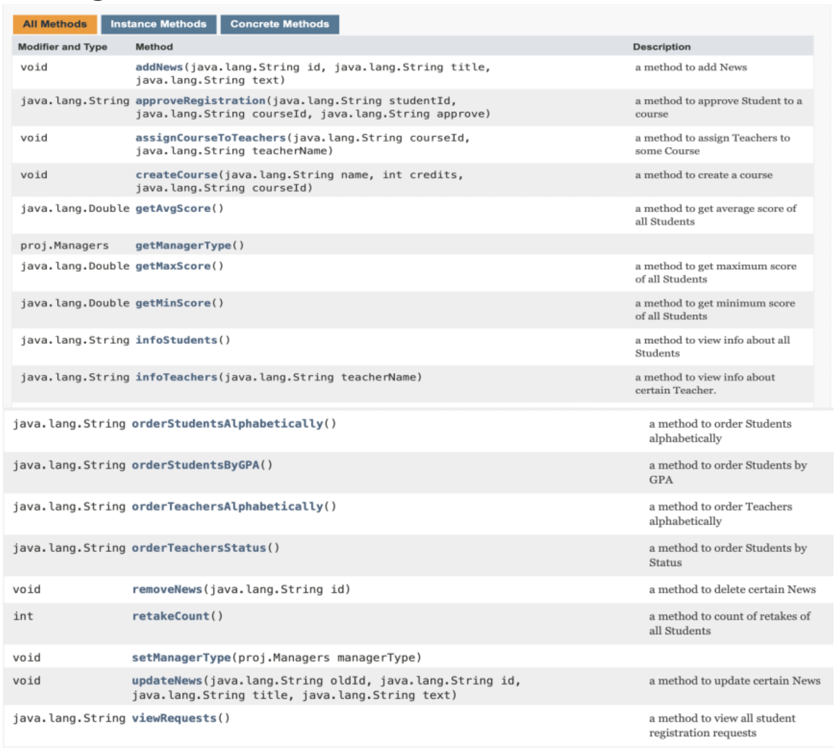
I repeat, there much more realized methods but we decided to attach here only necessary ones.

Issues Found

- In database some fields were null because of some bugs with serialization, so we spent a lot of time to fix it.
- Couldn't realize attendance.

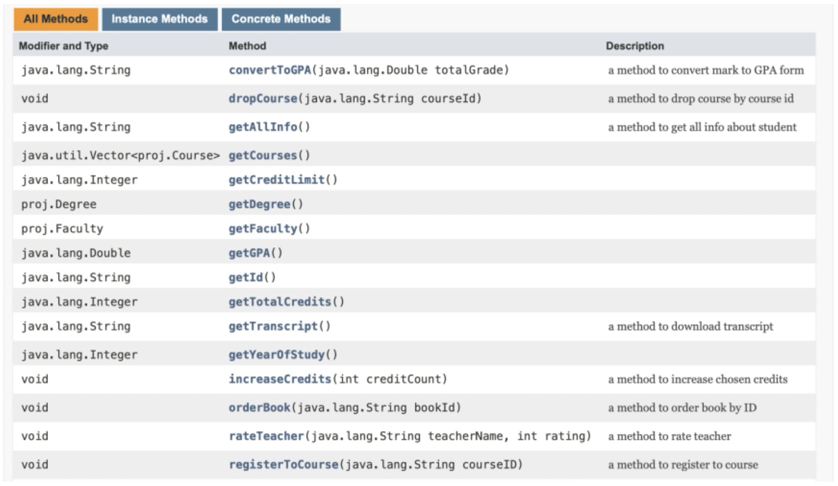
DOCUMENTATION

Manager

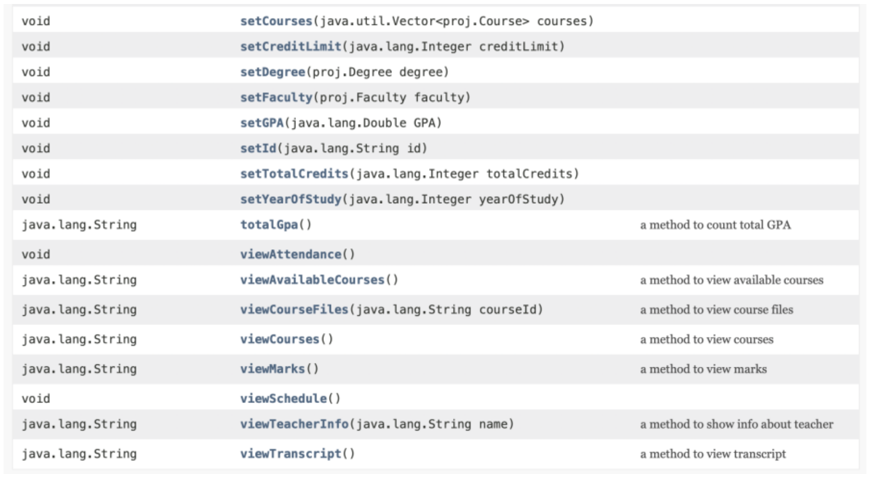


Modifier and Type	Method	Description
void	addNews(java.lang.String id, java.lang.String title, java.lang.String text)	a method to add News
java.lang.String	approveRegistration(java.lang.String studentId, java.lang.String courseId, java.lang.String approve)	a method to approve Student to a course
void	assignCourseToTeachers(java.lang.String courseId, java.lang.String teacherName)	a method to assign Teachers to some Course
void	createCourse(java.lang.String name, int credits, java.lang.String courseId)	a method to create a course
java.lang.Double	getAvgScore()	a method to get average score of all Students
proj.Managers	getManagerType()	
java.lang.Double	getMaxScore()	a method to get maximum score of all Students
java.lang.Double	getMinScore()	a method to get minimum score of all Students
java.lang.String	infoStudents()	a method to view info about all Students
java.lang.String	infoTeachers(java.lang.String teacherName)	a method to view info about certain Teacher.
java.lang.String	orderStudentsAlphabetically()	a method to order Students alphabetically
java.lang.String	orderStudentsByGPA()	a method to order Students by GPA
java.lang.String	orderTeachersAlphabetically()	a method to order Teachers alphabetically
java.lang.String	orderTeachersStatus()	a method to order Students by Status
void	removeNews(java.lang.String id)	a method to delete certain News
int	retakeCount()	a method to count of retakes of all Students
void	setManagerType(proj.Managers managerType)	
void	updateNews(java.lang.String oldId, java.lang.String id, java.lang.String title, java.lang.String text)	a method to update certain News
java.lang.String	viewRequests()	a method to view all student registration requests

Student

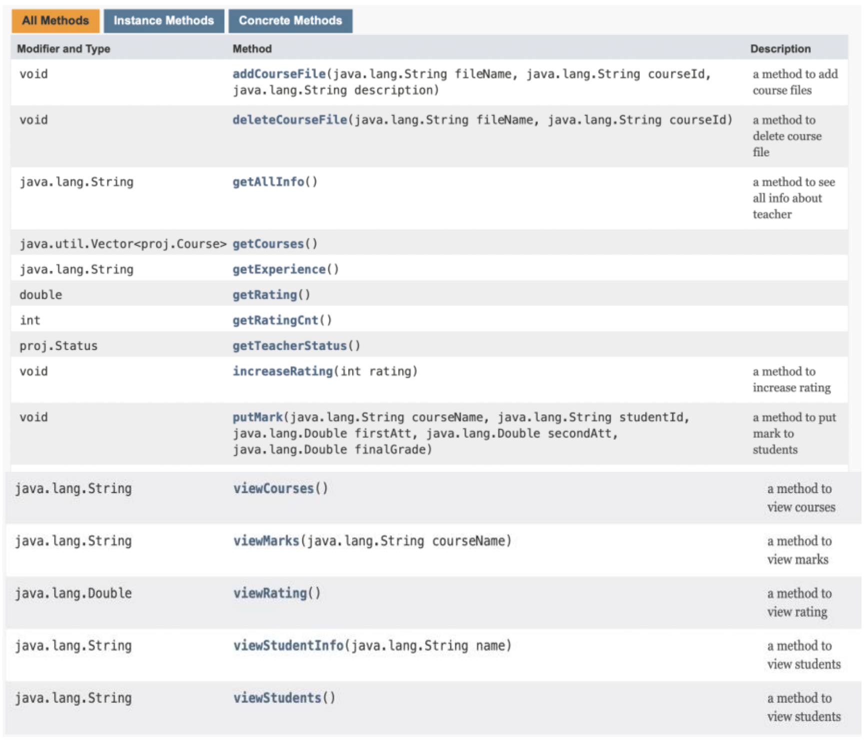


Modifier and Type	Method	Description
java.lang.String	convertToGPA(java.lang.Double totalGrade)	a method to convert mark to GPA form
void	dropCourse(java.lang.String courseId)	a method to drop course by course id
java.lang.String	getAllInfo()	a method to get all info about student
java.util.Vector<proj.Course>	getCourses()	
java.lang.Integer	getCreditLimit()	
proj.Degree	getDegree()	
proj.Faculty	getFaculty()	
java.lang.Double	getGPA()	
java.lang.String	getId()	
java.lang.Integer	getTotalCredits()	
java.lang.String	getTranscript()	a method to download transcript
java.lang.Integer	getYearOfStudy()	
void	increaseCredits(int creditCount)	a method to increase chosen credits
void	orderBook(java.lang.String bookId)	a method to order book by ID
void	rateTeacher(java.lang.String teacherName, int rating)	a method to rate teacher
void	registerToCourse(java.lang.String courseID)	a method to register to course



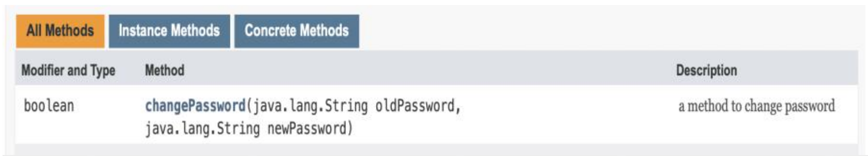
void	**setCourses**(java.util.Vector<proj.Course> courses)	
void	**setCreditLimit**(java.lang.Integer creditLimit)	
void	**setDegree**(proj.Degree degree)	
void	**setFaculty**(proj.Faculty faculty)	
void	**setGPA**(java.lang.Double GPA)	
void	**setId**(java.lang.String id)	
void	**setTotalCredits**(java.lang.Integer totalCredits)	
void	**setYearOfStudy**(java.lang.Integer yearOfStudy)	
java.lang.String	**totalGpa**()	a method to count total GPA
void	**viewAttendance**()	
java.lang.String	**viewAvailableCourses**()	a method to view available courses
java.lang.String	**viewCourseFiles**(java.lang.String courseId)	a method to view course files
java.lang.String	**viewCourses**()	a method to view courses
java.lang.String	**viewMarks**()	a method to view marks
void	**viewSchedule**()	
java.lang.String	**viewTeacherInfo**(java.lang.String name)	a method to show info about teacher
java.lang.String	**viewTranscript**()	a method to view transcript

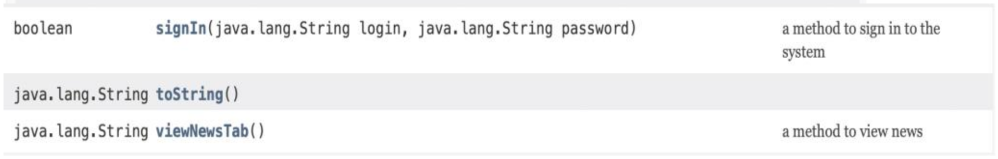
Teacher


All Methods
Instance Methods
Concrete Methods

Modifier and Type	Method	Description
void	**addCourseFile**(java.lang.String fileName, java.lang.String courseId, java.lang.String description)	a method to add course files
void	**deleteCourseFile**(java.lang.String fileName, java.lang.String courseId)	a method to delete course file
java.lang.String	**getAllInfo**()	a method to see all info about teacher
java.util.Vector<proj.Course>	**getCourses**()	
java.lang.String	**getExperience**()	
double	**getRating**()	
int	**getRatingCnt**()	
proj.Status	**getTeacherStatus**()	
void	**increaseRating**(int rating)	a method to increase rating
void	**putMark**(java.lang.String courseName, java.lang.String studentId, java.lang.Double firstAtt, java.lang.Double secondAtt, java.lang.Double finalGrade)	a method to put mark to students
java.lang.String	**viewCourses**()	a method to view courses
java.lang.String	**viewMarks**(java.lang.String courseName)	a method to view marks
java.lang.Double	**viewRating**()	a method to view rating
java.lang.String	**viewStudentInfo**(java.lang.String name)	a method to view students
java.lang.String	**viewStudents**()	a method to view students

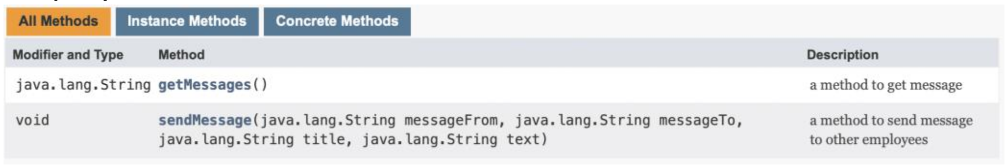
User


All Methods
Instance Methods
Concrete Methods



Modifier and Type	Method	Description
boolean	**changePassword**(java.lang.String oldPassword, java.lang.String newPassword)	a method to change password
boolean	**signIn**(java.lang.String login, java.lang.String password)	a method to sign in to the system
java.lang.String	**toString**()	
java.lang.String	**viewNewsTab**()	a method to view news

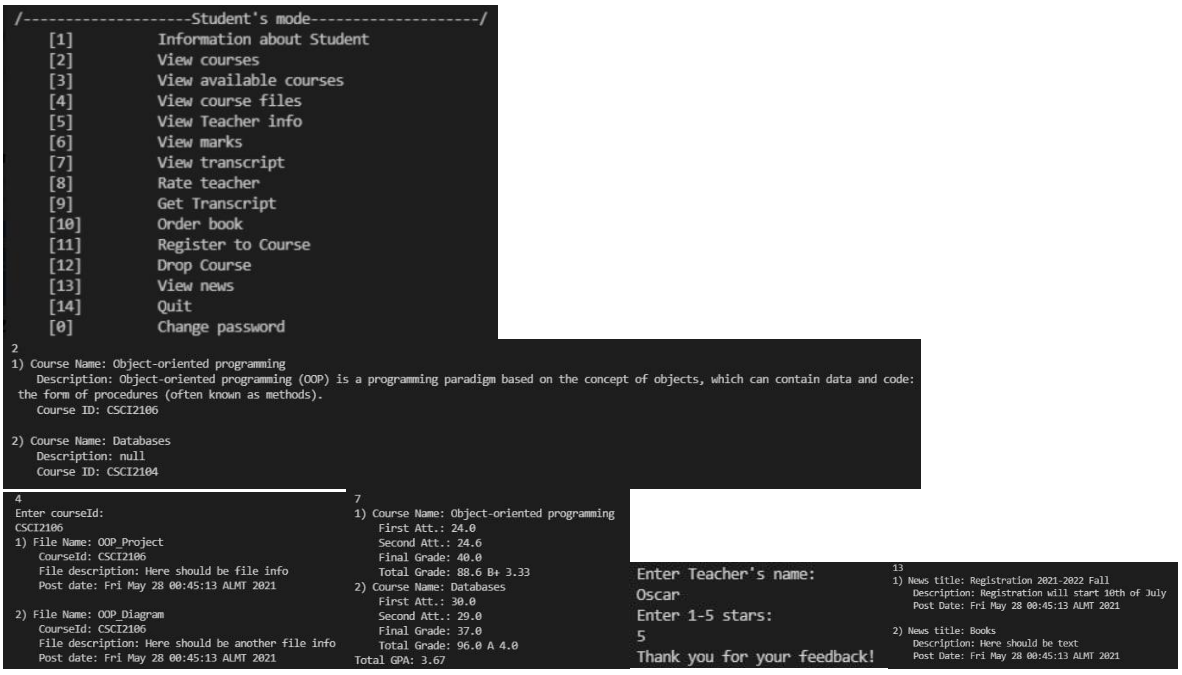
Employee



All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`java.lang.String`	`getMessages()`	a method to get message
`void`	`sendMessage(java.lang.String messageFrom, java.lang.String messageTo, java.lang.String title, java.lang.String text)`	a method to send message to other employees

How our project looks in console



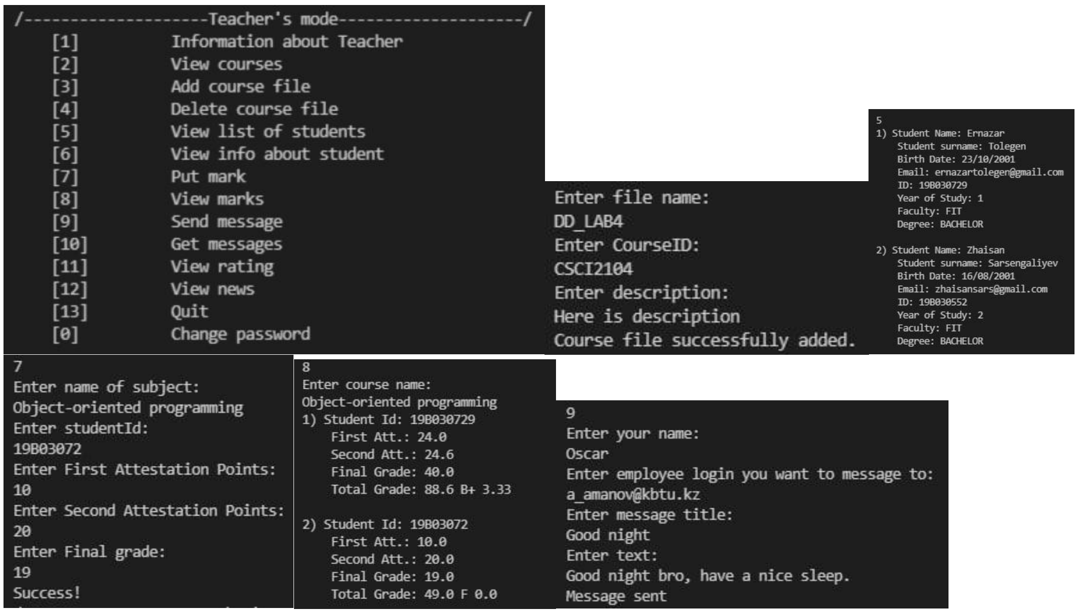
```
/--------------------Student's mode--------------------/
     [1]           Information about Student
     [2]           View courses
     [3]           View available courses
     [4]           View course files
     [5]           View Teacher info
     [6]           View marks
     [7]           View transcript
     [8]           Rate teacher
     [9]           Get Transcript
     [10]          Order book
     [11]          Register to Course
     [12]          Drop Course
     [13]          View news
     [14]          Quit
     [0]           Change password
2
1) Course Name: Object-oriented programming
    Description: Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code:
 the form of procedures (often known as methods).
    Course ID: CSCI2106

2) Course Name: Databases
    Description: null
    Course ID: CSCI2104
```

```
4
Enter courseId:
CSCI2106
1) File Name: OOP_Project
    CourseId: CSCI2106
    File description: Here should be file info
    Post date: Fri May 28 00:45:13 ALMT 2021

2) File Name: OOP_Diagram
    CourseId: CSCI2106
    File description: Here should be another file info
    Post date: Fri May 28 00:45:13 ALMT 2021
```

```
7
1) Course Name: Object-oriented programming
    First Att.: 24.0
    Second Att.: 24.6
    Final Grade: 40.0
    Total Grade: 88.6 B+ 3.33
2) Course Name: Databases
    First Att.: 30.0
    Second Att.: 29.0
    Final Grade: 37.0
    Total Grade: 96.0 A 4.0
Total GPA: 3.67
```

```
Enter Teacher's name:
Oscar
Enter 1-5 stars:
5
Thank you for your feedback!
```

```
13
1) News title: Registration 2021-2022 Fall
    Description: Registration will start 10th of July
    Post Date: Fri May 28 00:45:13 ALMT 2021

2) News title: Books
    Description: Here should be text
    Post Date: Fri May 28 00:45:13 ALMT 2021
```



```
/--------------------Teacher's mode--------------------/
     [1]           Information about Teacher
     [2]           View courses
     [3]           Add course file
     [4]           Delete course file
     [5]           View list of students
     [6]           View info about student
     [7]           Put mark
     [8]           View marks
     [9]           Send message
     [10]          Get messages
     [11]          View rating
     [12]          View news
     [13]          Quit
     [0]           Change password
```

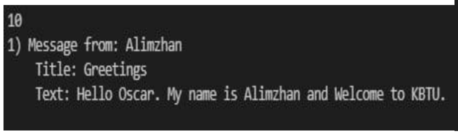
```
Enter file name:
DD_LAB4
Enter CourseID:
CSCI2104
Enter description:
Here is description
Course file successfully added.
```

```
5
1) Student Name: Ernazar
    Student surname: Tolegen
    Birth Date: 23/10/2001
    Email: ernazartolegen@gmail.com
    ID: 19B030729
    Year of Study: 1
    Faculty: FIT
    Degree: BACHELOR

2) Student Name: Zhaisan
    Student surname: Sarsengaliyev
    Birth Date: 16/08/2001
    Email: zhaisansars@gmail.com
    ID: 19B030552
    Year of Study: 2
    Faculty: FIT
    Degree: BACHELOR
```

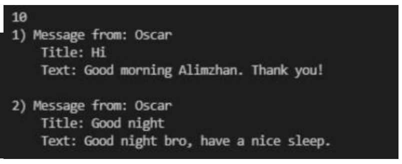
```
7
Enter name of subject:
Object-oriented programming
Enter studentId:
19B03072
Enter First Attestation Points:
10
Enter Second Attestation Points:
20
Enter Final grade:
19
Success!
```

```
8
Enter course name:
Object-oriented programming
1) Student Id: 19B030729
    First Att.: 24.0
    Second Att.: 24.6
    Final Grade: 40.0
    Total Grade: 88.6 B+ 3.33

2) Student Id: 19B03072
    First Att.: 10.0
    Second Att.: 20.0
    Final Grade: 19.0
    Total Grade: 49.0 F 0.0
```

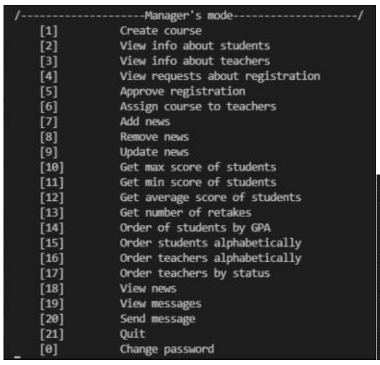
```
9
Enter your name:
Oscar
Enter employee login you want to message to:
a_amanov@kbtu.kz
Enter message title:
Good night
Enter text:
Good night bro, have a nice sleep.
Message sent
```



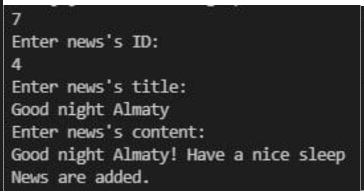
```
10
1) Message from: Alimzhan
    Title: Greetings
    Text: Hello Oscar. My name is Alimzhan and Welcome to KBTU.
```



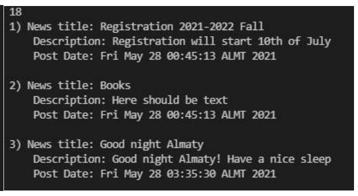
```
10
1) Message from: Oscar
    Title: Hi
    Text: Good morning Alimzhan. Thank you!

2) Message from: Oscar
    Title: Good night
    Text: Good night bro, have a nice sleep.
```



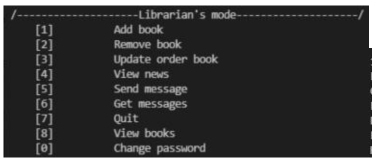
```
/--------------------Manager's mode--------------------/
    [1]          Create course
    [2]          View info about students
    [3]          View info about teachers
    [4]          View requests about registration
    [5]          Approve registration
    [6]          Assign course to teachers
    [7]          Add news
    [8]          Remove news
    [9]          Update news
    [10]         Get max score of students
    [11]         Get min score of students
    [12]         Get average score of students
    [13]         Get number of retakes
    [14]         Order of students by GPA
    [15]         Order students alphabetically
    [16]         Order teachers alphabetically
    [17]         Order teachers by status
    [18]         View news
    [19]         View messages
    [20]         Send message
    [21]         Quit
    [0]          Change password
```



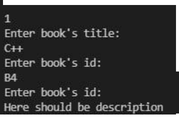
```
7
Enter news's ID:
4
Enter news's title:
Good night Almaty
Enter news's content:
Good night Almaty! Have a nice sleep
News are added.
```



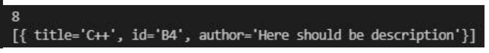
```
18
1) News title: Registration 2021-2022 Fall
    Description: Registration will start 10th of July
    Post Date: Fri May 28 00:45:13 ALMT 2021

2) News title: Books
    Description: Here should be text
    Post Date: Fri May 28 00:45:13 ALMT 2021

3) News title: Good night Almaty
    Description: Good night Almaty! Have a nice sleep
    Post Date: Fri May 28 03:35:30 ALMT 2021
```



```
/--------------------Librarian's mode--------------------/
    [1]          Add book
    [2]          Remove book
    [3]          Update order book
    [4]          View news
    [5]          Send message
    [6]          Get messages
    [7]          Quit
    [8]          View books
    [0]          Change password
```



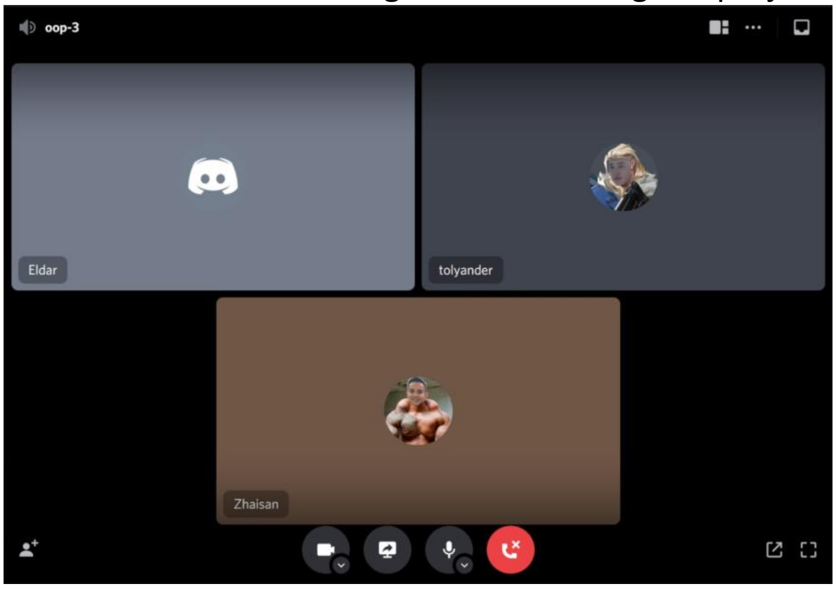
```
1
Enter book's title:
C++
Enter book's id:
B4
Enter book's id:
Here should be description
```

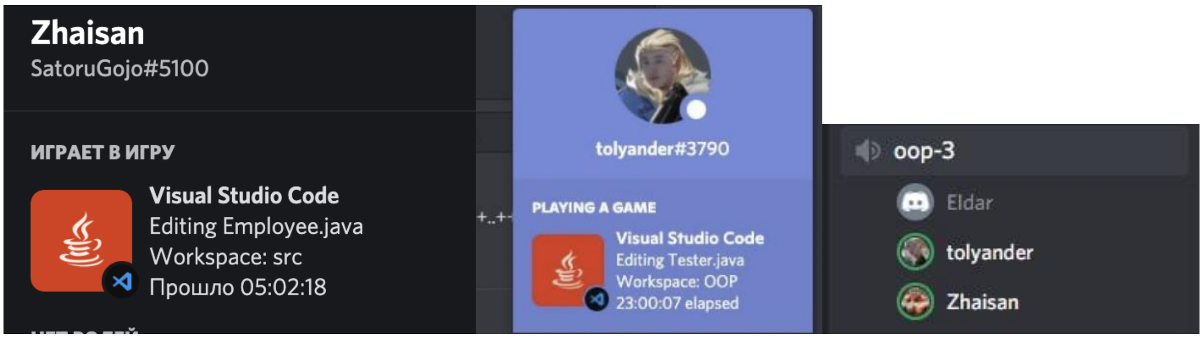


```
8
[{ title='C++', id='B4', author='Here should be description'}]
```

Team processes

We communicated through Discord during the project implementation.





Zhaisan
SatoruGojo#5100

ИГРАЕТ В ИГРУ

Visual Studio Code
Editing Employee.java
Workspace: src
Прошло 05:02:18

tolyander#3790

PLAYING A GAME

Visual Studio Code
Editing Tester.java
Workspace: OOP
23:00:07 elapsed

oop-3
Eldar
tolyander
Zhaisan