Intranet project report

Introduction

Our main task in this project was to realize the Intranet of university in java, using all learnt OOP concepts and functionalities. The first step of project development was to create a Use Case Diagram and class diagram. Implementing those diagrams helped us to organize the project, divide them into separate classes, and to add different methods.

Main part

We have abstract classes person and employee extending person, and other classes such as student, teacher, executor, manager, admin extending some of them.

For data validity and restricting some limits we use authorization. Where we used hashcode for password for the purpose of security. Also, users can change the password, it is obtained by using just setPassword(method);

To begin with, we created important classes like student, teacher, admin, executor and etc. Their jobs were mentioned in created Use Case diagram. Majority of classes’ objects we stored using serialization, and we desterilize them, when we need to make some changes.

This is the example of one collection that we store in txt file, and when we deserilize it we do the type casting.

FileInputStream fis = **new** FileInputStream("students.txt");

ObjectInputStream ois = **new** ObjectInputStream(fis);

ArrayList<Student> students = **new** ArrayList<Student>();

students = (ArrayList<Student>) ois.readObject();

**Admin** is the important class in project, admin can: add/delete student, add/delete teacher, add/delete course, write news and so on.

For example, realization of the add teacher method: here, we firstly print out all teachers, then we line by line enter all needed for constructor fields, then we recall the addTeacher() method, written in the admin class, which returns back the object of the teacher then we add that object to our collection and serialize it.

**else** **if** (choice.equals("Add teacher")) {

Iterator it = courses.iterator();

**while**(it.hasNext()){

System.***out***.println(((Course)it.next()).getName());

}

String tname = in.nextLine();

**while** (!tname.isEmpty()) {

**int** salary = Integer.*parseInt*(in.nextLine());

**int** id = Integer.*parseInt*(in.nextLine());

**int** year = Integer.*parseInt*(in.nextLine());

teachers.add(a.addTeachers(tname, id, salary,year,in.nextLine()));

tname = in.nextLine();

} *Serialize*("teachers.txt", teachers);

Next class that we have is **Student** class, one of the it’s abilities is to see attendance, when student chooses “View transcript”, this code is run: student.viewAttendance(username);

ViewAttendance() method in Student class:

Here, it reads from the file, which name is this.name+”attendance.txt”, then prints out the content of that txt file.

**public** **void** viewAttendance(String name) **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** FileReader(name + "attendance.txt"));

String line = br.readLine();

**try**{**while** (!line.isEmpty()) {

System.***out***.println(line);

line = br.readLine();

}}**catch**(NullPointerException e){

}

}

Another main functionality of the student is to register for course, if student enters “Register”, the following code will run: Here, it iterates through the collection courses, then if the course’s faculty and year is equals to the faculty and year of study of the student, who is registering, respectively, also if the time period is in the limited period then we print out all the curses available, and student then types course names that he want to enrole.

**else** **if** (choice.equals("Register")) {

Iterator c = courses.iterator();

**while**(c.hasNext()){

Course course =(Course)c.next();

**if**(course.getFaculty().equals(st.getFaculty()) && course.getYear()==(st.getYearOfStudy())){

Date firstDate = **new** ~~Date~~(117,10,20,23,59,59);

Date secondDate = **new** ~~Date~~(117,10,30,23,59,59);

dateFormat = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

Date date = **new** Date();

**if**(date.before(secondDate) && date.after(firstDate)){

System.***out***.println(course.getName());

}

}

}

st.setCourse(in.nextLine());

System.***out***.println("You successfully registered all courses.");

Thread.*sleep*(3000);

}

Furthermore, another essential class is **teacher** class, which has main functionalities like: putMark(), putAttendance(), sendOrder(), showCourseFiles(), uploadFile() and etc.

Let’s see one of them, sendOrder() method of the teacher class is for sending teacher some orders to the executor, and it is made again using serialization, here, we add to the hashmap with current date key and order context as value, then serialize it.

FileOutputStream fos = **new** FileOutputStream("orders.txt");

ObjectOutputStream oos = **new** ObjectOutputStream(fos);

DateFormat dateFormat = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

Calendar cal = Calendar.*getInstance*();

*orders*.put((Date) cal.getTime(), order);

oos.writeObject(*orders*);

oos.flush();

oos.close();

So, the next class is **executor**, his role is to see the list of came orders, then accept or reject them.

For the **manager** class, we added new functionality that manager will check the news’ comments and remove them if there is some bad content. This part realization is in this way, we already have the class **news**, which has fields like subject, content, from (where, all of them are strings) and collection of comments, so we display the news that is selected by the manger to change, then we just remove from the vector of comments using selected number.

Finally, we added guest part, guest just sees all the news and can write comments to them, we decided to do not create a class for it, because guest does not have any fields and does not need to log in, it is just a choice.

Conclusion

Making this project helped us to revise all have learnt OOP concepts, consolidated new knowledge by the practice. In this team project, we learnt skills of working in a team, actually how to share ideas, combine our knowledge and so on.