Report UML : Course Selection System

Adel ALI - Marion WILMS EI17

December 3, 2018

Contents

1 UML Design

1.1 Subject presentation

We have to implement a simplified course selection system. First, a student needs to login to the system, then he will have three possibilities :

- Check his current planning : see his course selection (which is initially empty)
- Browse Available Courses : add a course to his planning (the number of courses he can choose is limited to 4)
- Logout

We have add a constraint, to match the reality, we have limited the number of student in a course to 10.

1.2 Use Case Diagram

We have a single actor : a student. So we modeled the communication between the functionality of the course selection system and the users in a Use Case Diagram. The student wants to register to a course, that means that he can want to login or logout, to browse available courses or to check his planning.

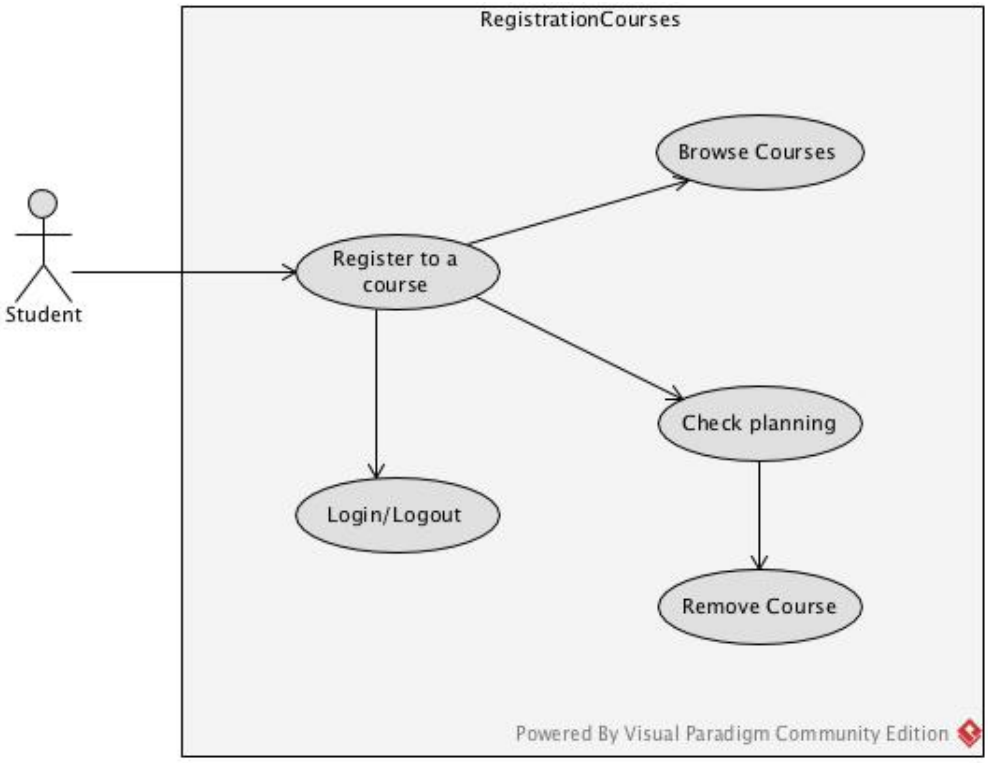


Figure 1: Use Case Diagram

1.3 Sequence Diagrams

We have modeled the sequence diagrams for this use case :

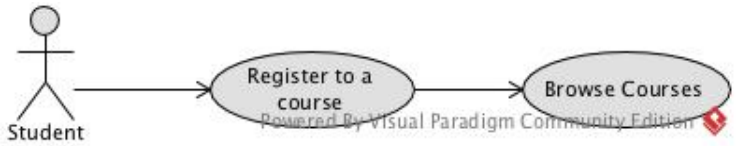


Figure 2: Use Case for the sequence diagrams

1.3.1 Happy Scenario

In the happy scenario : Adel the student logs in to the system successfully, he chooses to browse available courses, as he has less than 4 courses, he can access the list of available courses, as there are available courses (courses with less than 10 students), courses are shown in Adel's interface, Adel chooses a course, the course is added to Adel's planning, the number of student in the course selected is incremented and finally Adel logs out to the system.

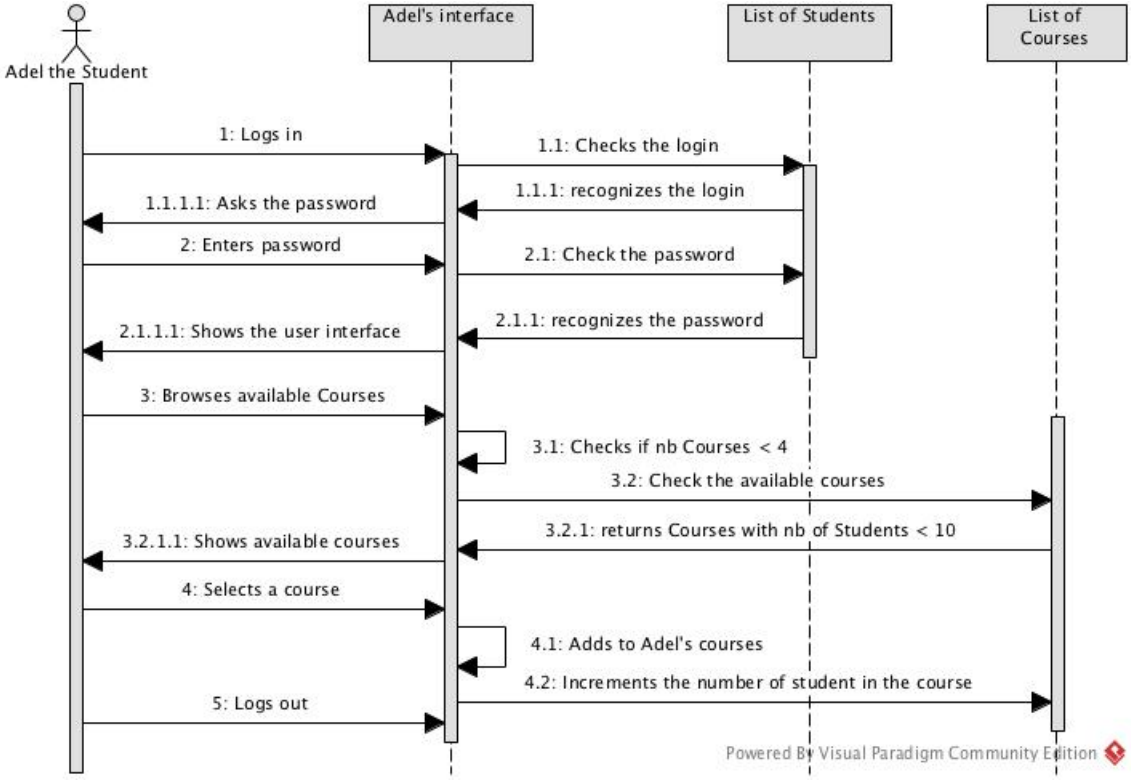


Figure 3: Sequence Diagram Happy Scenario

1.3.2 Sad Scenario

In the bad scenario : Adel the student logs in to the system but he makes a mistake in his password, he tries again successfully, he chooses to browse available courses, as he has less than 4 courses, he can access the list of available courses, there are no available courses so Adel logs out to the system

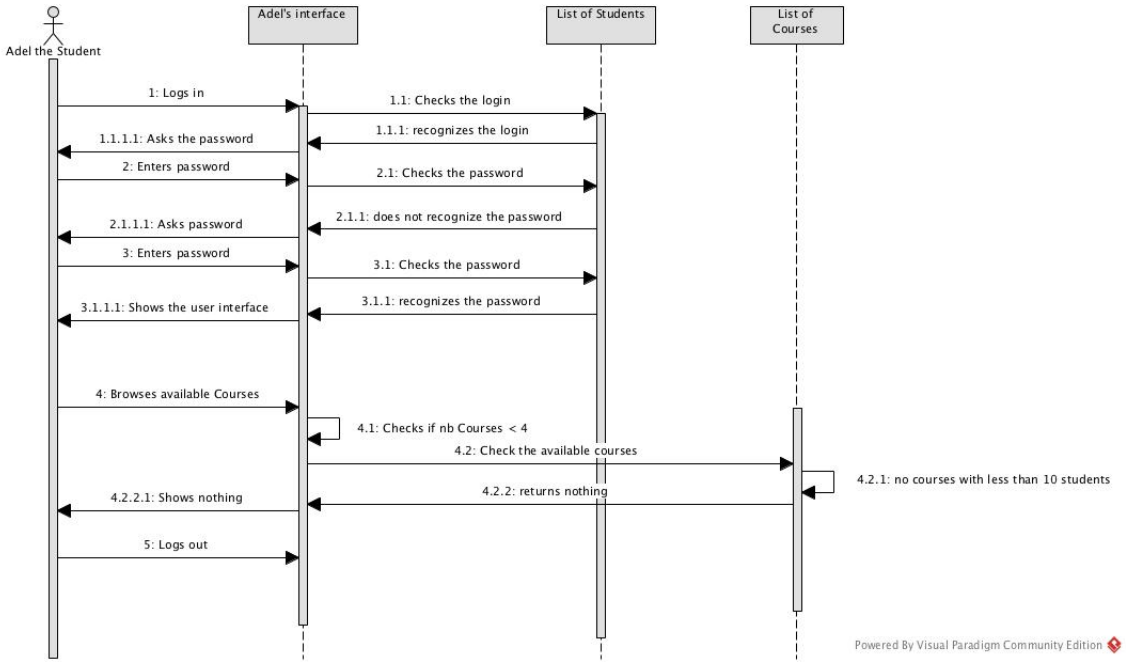
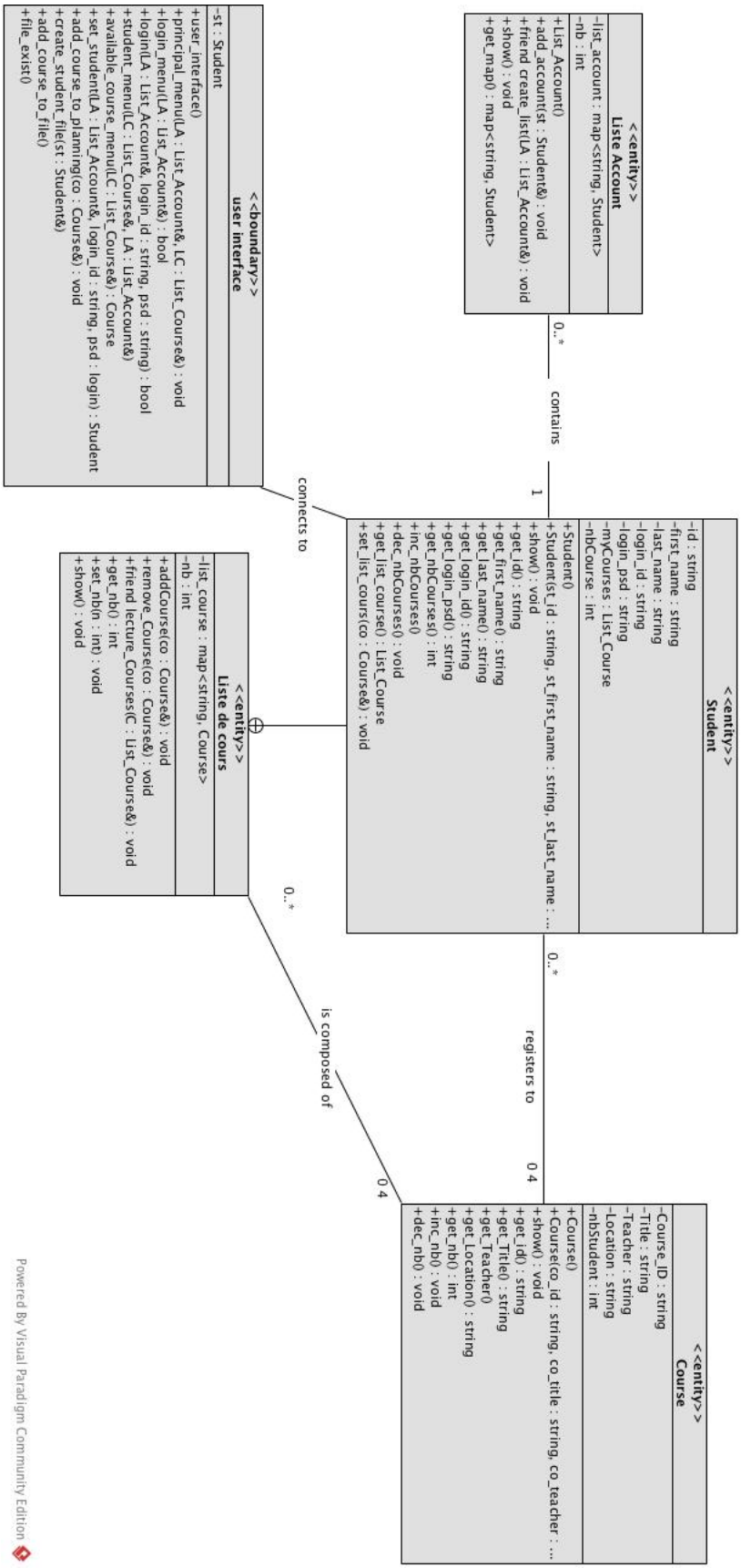


Figure 4: Sequence Diagram Sad Scenario

1.4 Class Diagram

In this diagram, the classes of the system have been modeled. For the implementation, the same names have been chosen.


<<entity>>
Liste Account
-list_account : map<string, Student>
-nb : int
+List_Account()
+add_account(st : Student&) : void
+friend create_list(LA : List_Account&) : void
+show() : void
+get_map() : map<string, Student>
0..*
contains
1
<<entity>>
Student
-id : string
-first_name : string
-last_name : string
-login_id : string
-login_psd : string
-myCourses : List_Course
-nbCourse : int
+Student()
+Student(st_id : string, st_first_name : string, st_last_name : ...
+show() : void
+get_id() : string
+get_first_name() : string
+get_last_name() : string
+get_login_id() : string
+get_login_psd() : string
+get_nbCourses() : int
+inc_nbCourses()
+dec_nbCourses() : void
+get_list_course() : List_Course
+set_list_cours(co : Course&) : void
<<boundary>>
user interface
-st : Student
+user_interface()
+principal_menu(LA : List_Account&, LC : List_Course&) : void
+login_menu(LA : List_Account&) : bool
+login(LA : List_Account&, login_id : string, psd : string) : bool
+student_menu(LC : List_Course&, LA : List_Account&)
+available_course_menu(LC : List_Course&) : Course
+set_student(LA : List_Account&, login_id : string, psd : login) : Student
+add_course_to_planning(co : Course&) : void
+create_student_file(st : Student&)
+add_course_to_file()
+file_exist()
connects to
<<entity>>
Liste de cours
-list_course : map<string, Course>
-nb : int
+addCourse(co : Course&) : void
+remove_Course(co : Course&) : void
+friend lecture_Courses(C : List_Course&) : void
+get_nb() : int
+set_nb(n : int) : void
+show() : void
0..*
is composed of
0 4
registers to
0..*
0 4
<<entity>>
Course
-Course_ID : string
-Title : string
-Teacher : string
-Location : string
-nbStudent : int
+Course()
+Course(co_id : string, co_title : string, co_teacher : ...
+show() : void
+get_id() : string
+get_Title() : string
+get_Teacher()
+get_Location() : string
+get_nb() : int
+inc_nb() : void
+dec_nb() : void
Powered By Visual Paradigm Community Edition

1.5 State Transition Diagram

We have decided to model the state transition diagram of the class Course.

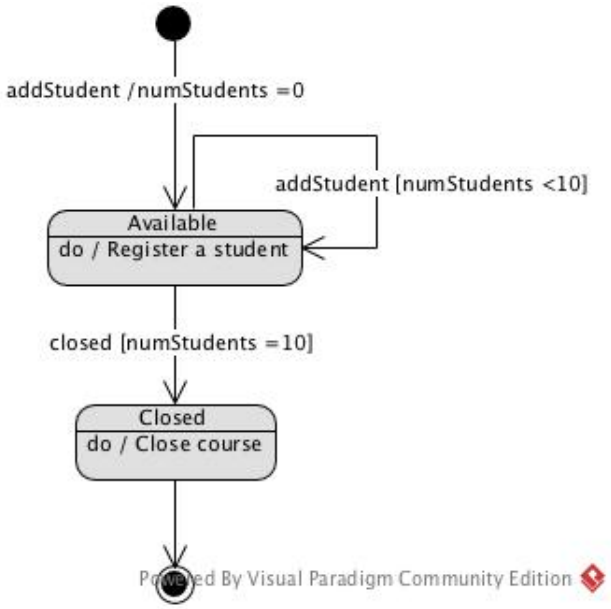


Figure 5: State Transition Diagram of the class Course

2 Implementation and results

2.1 Implementation

We have implemented the course selection system in c++. We have implemented the different classes according to our UML class diagram.

We have used map to stock all the account in Account.txt, to stock all the course in Course.txt and to stock the courses of a student.

To save the courses subscribed by a student, we create a txt file FistnameLastname.txt when the student logs out. Then when he logs in, his map with his course is initialized with the Courses in his file.

2.2 Results

When the student enters the application, a principal menu is shown (fig. 6). He can choose to log in or to quit the application.

```
********** Administrator ISMIN Course Selection System **********
1- Login
2- Quitter
```

Figure 6: Principal Menu

If the students chooses to log in, he enters is user name, the application checks if he is in the list of account then it asks the password. If the password is incorrect, the student must try again. If the password is correct, he can access his user interface.

```
Please enter your id:
jean.lebleu
Password :
elbelnaej
```

Figure 7: Login Menu

```
--------- Welcome to your profile Jean LeBleu. -------------------
********** Administrator ISMIN Course Selection System **********
1- Check Planning
2- Browse available course
3- Logout
```

Figure 8: Student Menu

The student can check his current planning, that is to say all the courses he has registered for. The number maximal of courses he can register for is 4.

If he does not have registered for a course yet, the application shows "no course subscribed". Otherwise, the application shows all the courses he has register for. Then he can remove a course or return to his user interface.

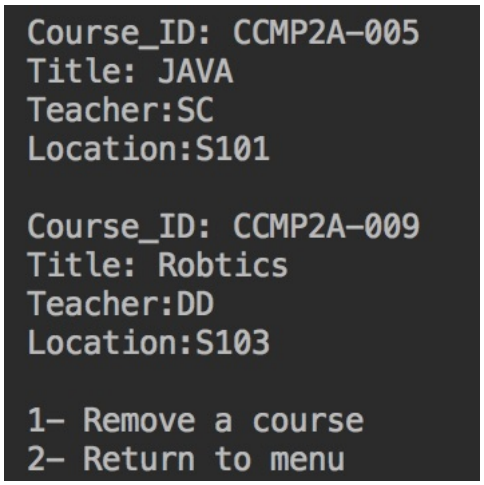


Figure 9: Check Planning

When he chooses "Browse available course", all available courses are shown with a number. To choose a course, the student must enter the number. If he has already 4 courses, the application shows "You have too many courses".

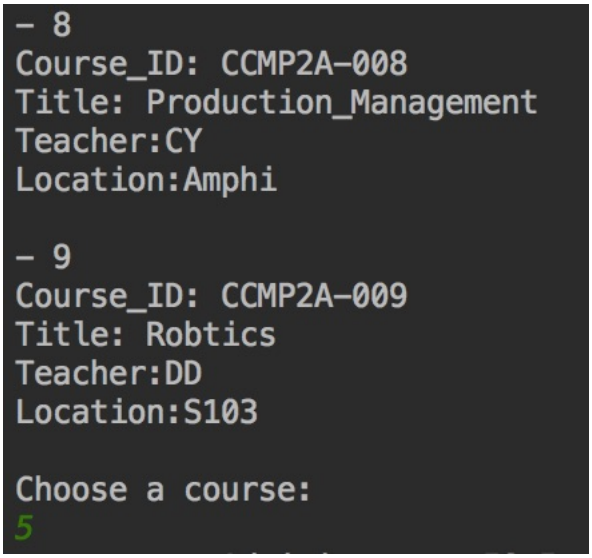


Figure 10: Browse Courses

3 Conclusion

So, we have implemented a simplified course selection system. The results match our needs and expectations.