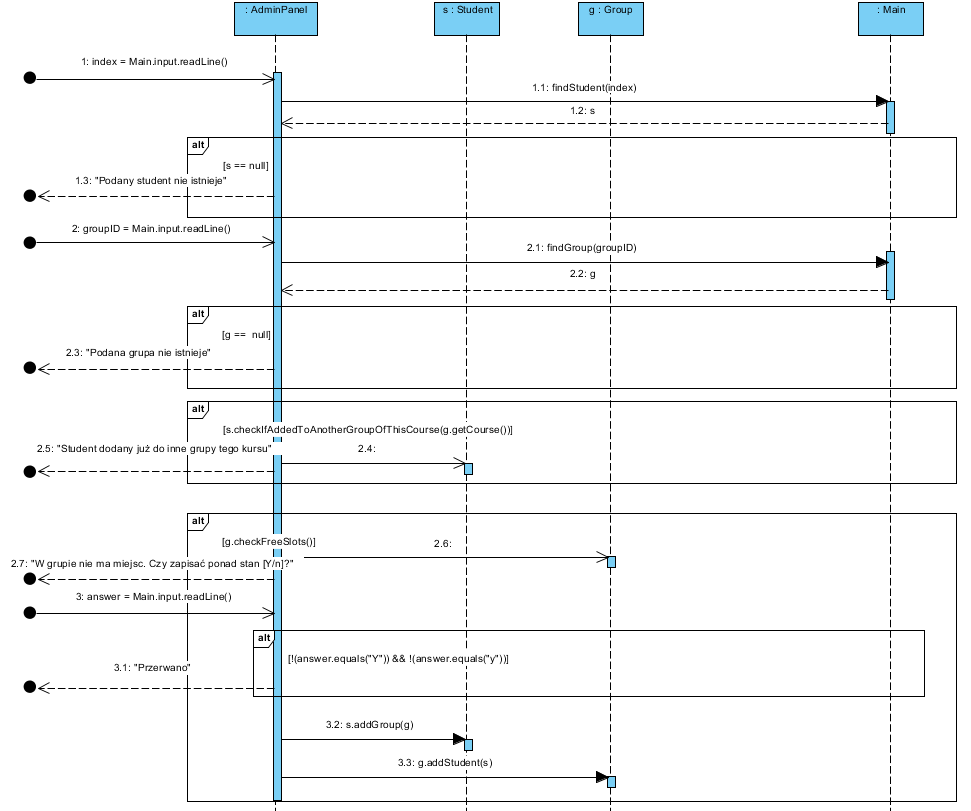
|  |  |
| --- | --- |
| Paweł Biel, 225949  Bartosz Rodziewicz, 226105 | Wrocław 10.01.2018 |

**Laboratorium 8-11 – Sprawozdanie**

* **6 etap** (tydzień 8): Opracowanie diagramów sekwencji dla wybranych przypadków użycia, reprezentujących usługi oprogramowania, wynikających również z wykonanych diagramów czynności. Definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.
* **7 etap** (tygodnie 9–10): Opracowanie diagramów sekwencji dla wybranych przypadków użycia reprezentujących usługi oprogramowania wynikających również z wykonanych diagramów czynności. Definicja operacji klas na podstawie diagramów sekwencji w języku Java. Zastosowanie projektowych wzorców zachowania.
* **8 etap** (tydzień 11): Opracowanie diagramu stanów dla wybranej klasy, reprezentującego wpływ różnych przypadków użycia na zmiany stanów tej klasy, modelowanych za pomocą diagramów sekwencji.

1. **Diagramy sekwencji** 
   1. **addStudentToGroup()**



System.out.print("Podaj numer indeksu studenta: ");

String index = Main.input.readLine();

Student s = Main.findStudent(index);

if (s == null) {

System.out.println("Podany student nie istnieje.");

return;

}

System.out.print("Podaj ID grupy: ");

String groupID = Main.input.readLine();

Group g = Main.findGroup(groupID);

if (g == null) {

System.out.println("Podana grupa nie istnieje.");

return;

}

if(s.checkIfAddedToAnotherGroupOfThisCourse(g.getCourse())) {

System.out.println("Student dodany juz do innej grupy tego kursu.");

return;

}

if(g.checkFreeSlots()) {

System.out.print("W grupie nie ma miejsc. Czy zapisac ponad stan [Y/n]?");

String answer = Main.input.readLine();

if (!(answer.equals("Y")) && !(answer.equals("y"))) {

System.out.println("Przerwano.");

return;

}

}

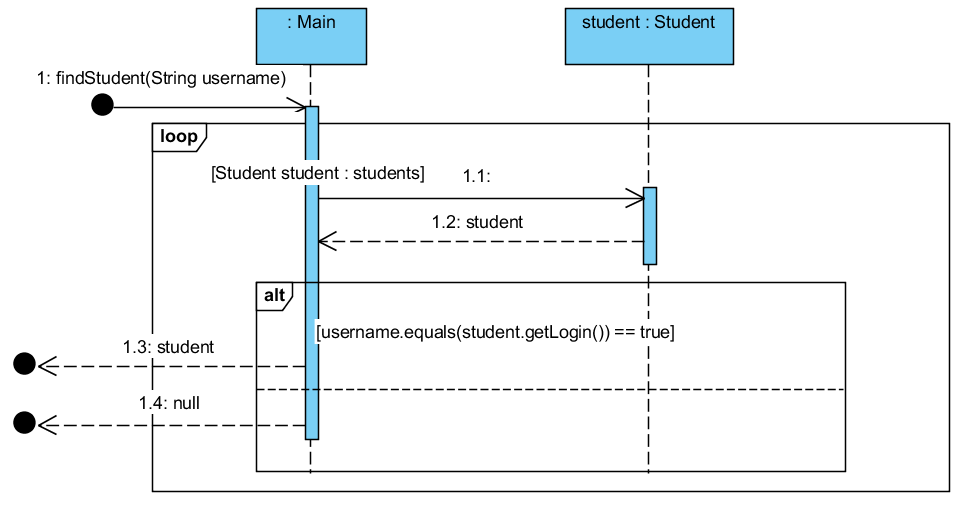
s.addGroup(g);

g.addStudent(s);

System.out.println("Zapisano studenta.");

}

* + 1. **findStudent(index)**



public static Student findStudent(String username) {

for(Student student : students) {

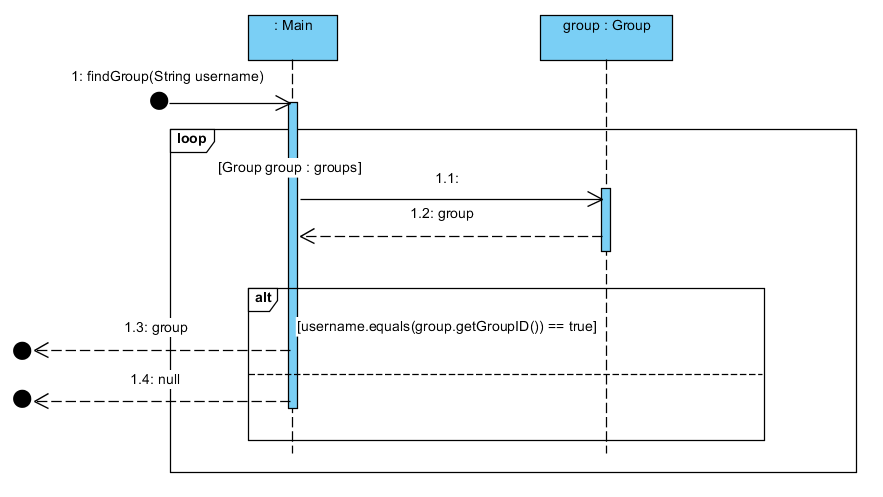
if (username.equals(student.getLogin()))

return student;

}

return null;

}

* + 1.  **findGroup()**

public static Group findGroup(String groupID) {

for(Group group : groups) {

if (groupID.equals(group.getGroupID()))

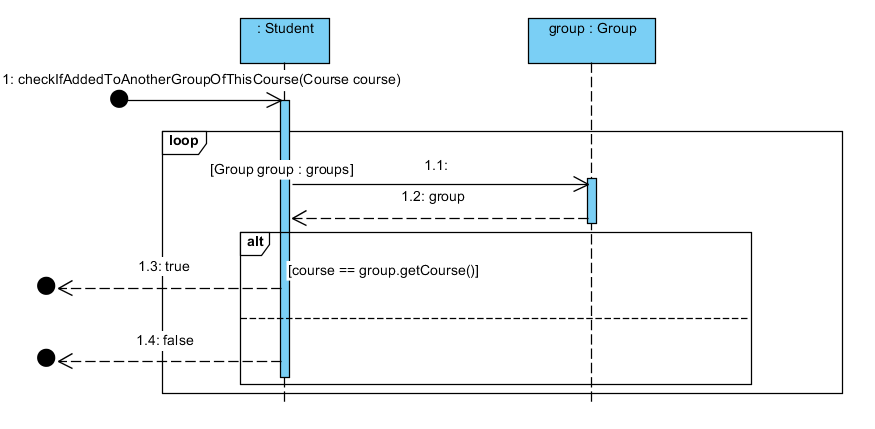
return group;

}

return null;

}

* + 1. **checkIfAddedToAnotherGroupOfThisCourse()**



public boolean checkIfAddedToAnotherGroupOfThisCourse(Course course) {

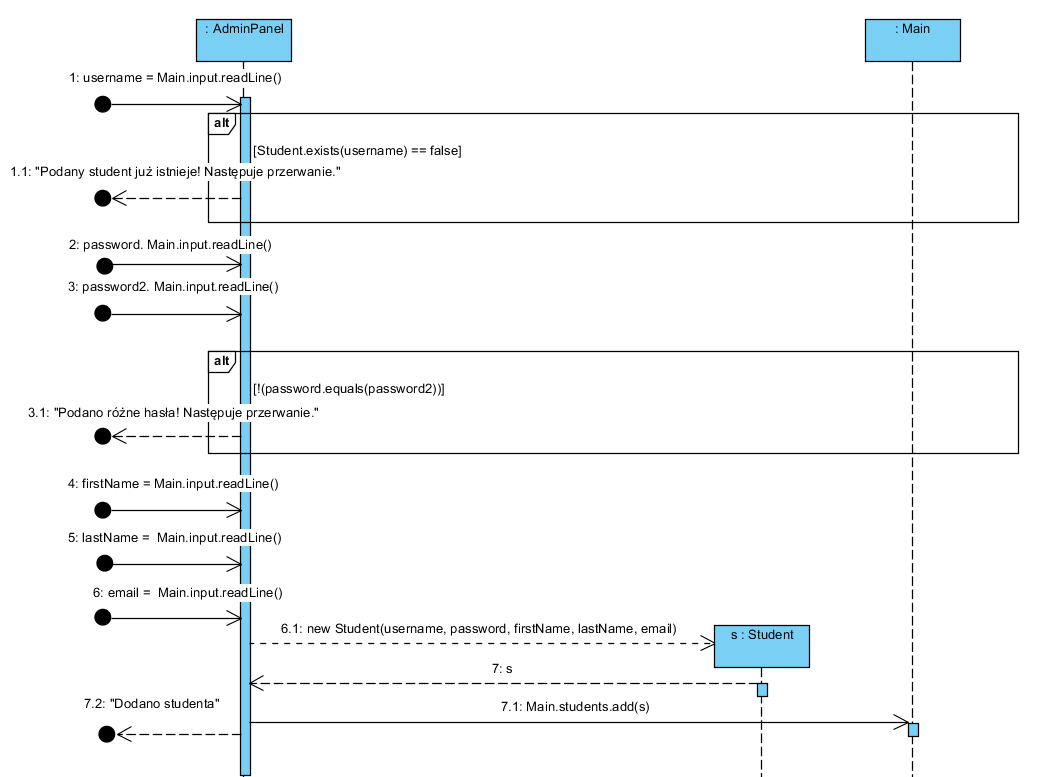
for (Group group : groups) {

if (course == group.getCourse()) return true;

}

return false;

}

* 1. **addStudent()**

private void addStudent() throws IOException {

System.out.print("Podaj indeks (login): ");

String username = Main.input.readLine();

if (Student.exists(username)) {

System.out.println("Podany student juz istnieje! Nastepuje przerwanie.");

return;

}

System.out.print("Podaj haslo: ");

String password = Main.input.readLine();

System.out.print("Powtorz haslo: ");

String password2 = Main.input.readLine();

if (!(password.equals(password2))) {

System.out.println("Podano rozne hasla! Nastepuje przerwanie.");

return;

}

System.out.print("Podaj imie: ");

String firstName = Main.input.readLine();

System.out.print("Podaj nazwisko: ");

String lastName = Main.input.readLine();

System.out.print("Podaj email: ");

String email = Main.input.readLine();

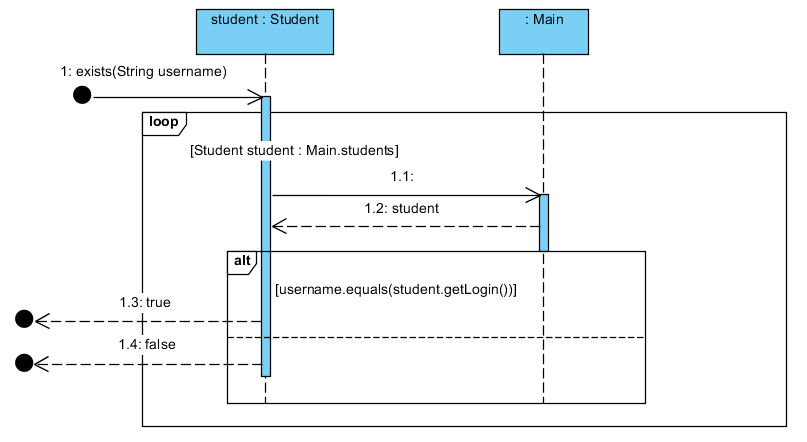
Student s = new Student(username, password, firstName, lastName, email);

Main.students.add(s);

System.out.print("Dodano studenta.");

}

* + 1. Exists()

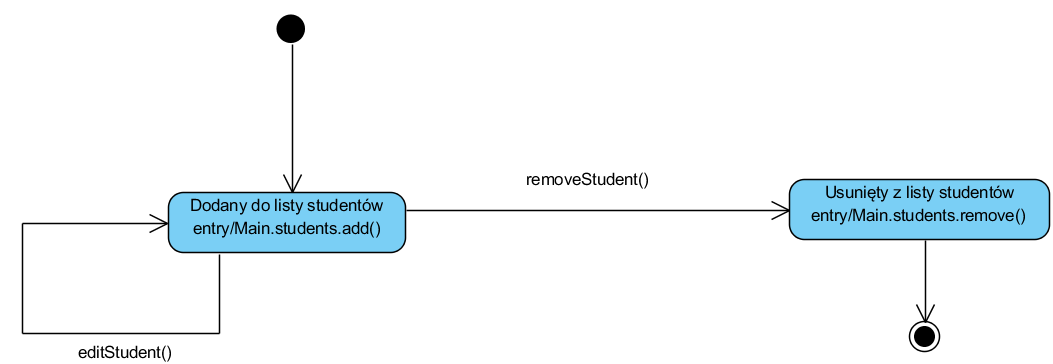


public static boolean exists(String username) {

for (Student student : Main.students) {

if (username.equals(student.getLogin()))

return true; }return false;}

1. Diagramy stanów
   1. Dla obiektu typu Student
   2. Dla obiektu typu Course

