AIAD

Library Management Simulation with Distributed Agents

GROUP 01:
Ricardo Cardoso - 201604686
João Lemos - 201000660
Simão Silva - 201605422

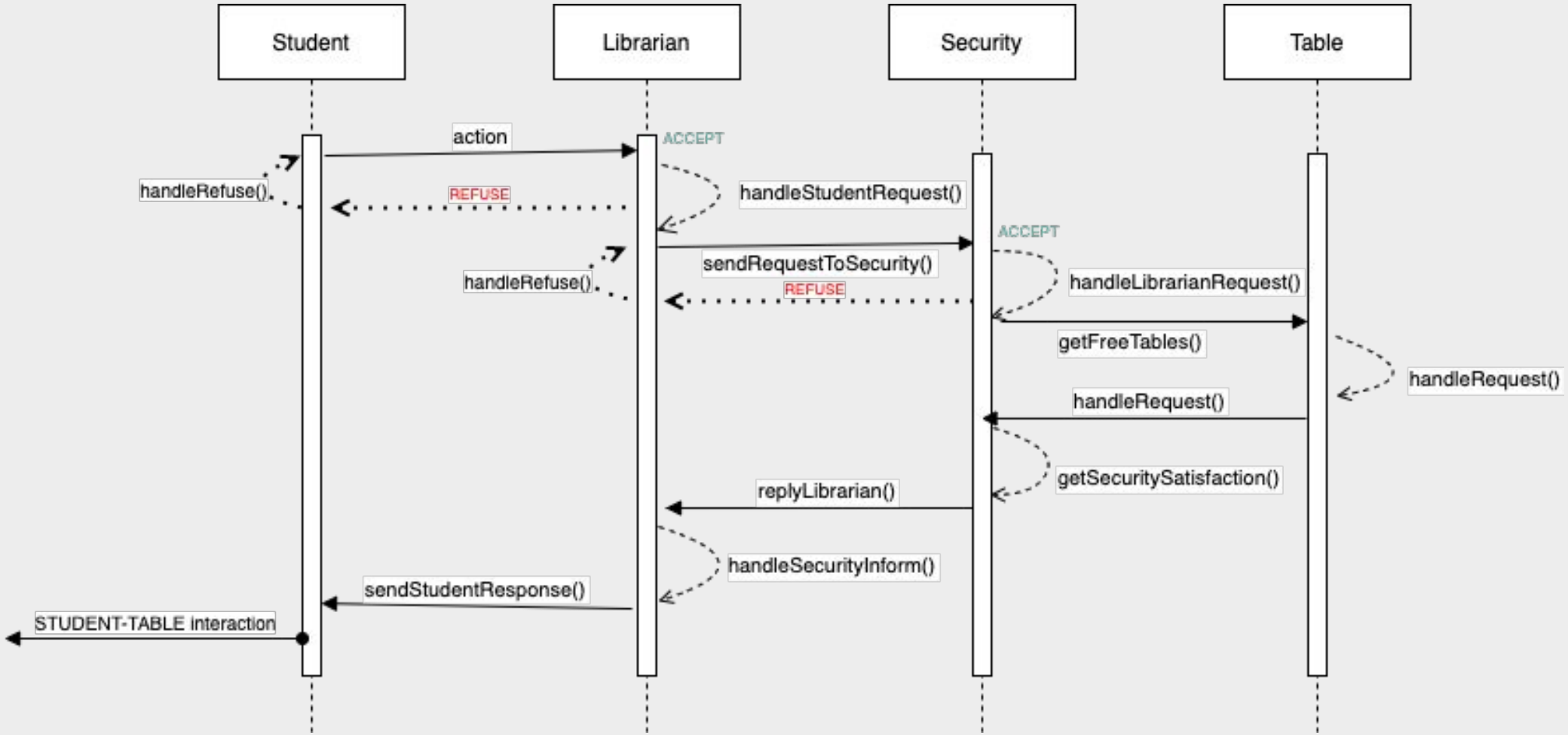
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master's Degree in Informatics and Computing Engineering

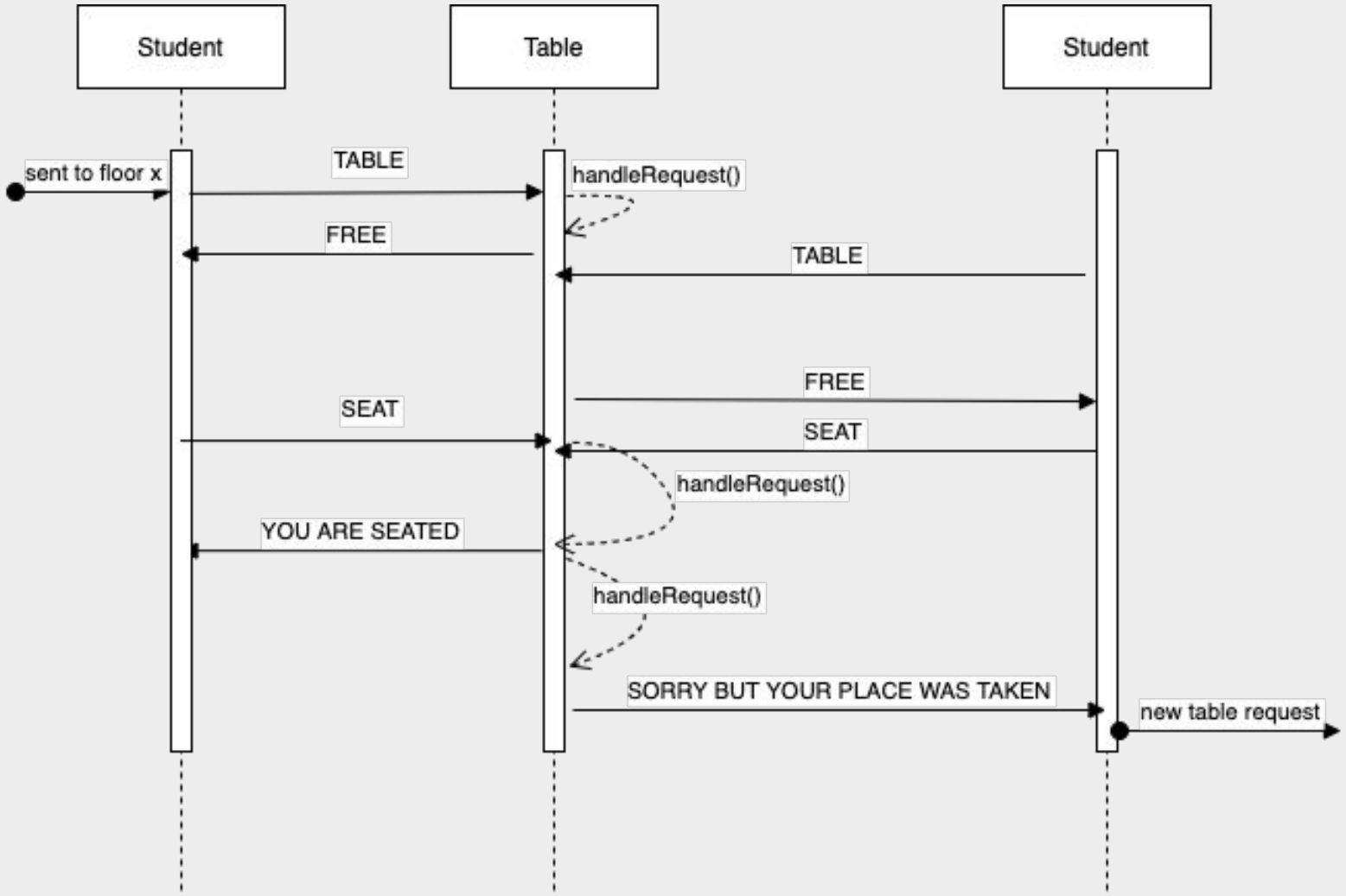
1. Problem Description

- The project was made to represent a student allocation problem on the faculty's library;
- Each Student (client) requests allocation to a table;
- The Librarian sends the student's information to each floor in order to get an approval rating from each floor's Security;
- Upon receiving the Librarian request, the floor's Security consults with each of the tables in his floor to see if any table is free to receive the Student. The Security also calculates it's desire to accept that Student (approval rating) based on the number of free tables and the floor's main course.
- This information is passed to the Librarian that then shows the floor with the higher approval rating to the Student;
- Upon receiving the floor information, the Student looks for a free table in that floor and sits.

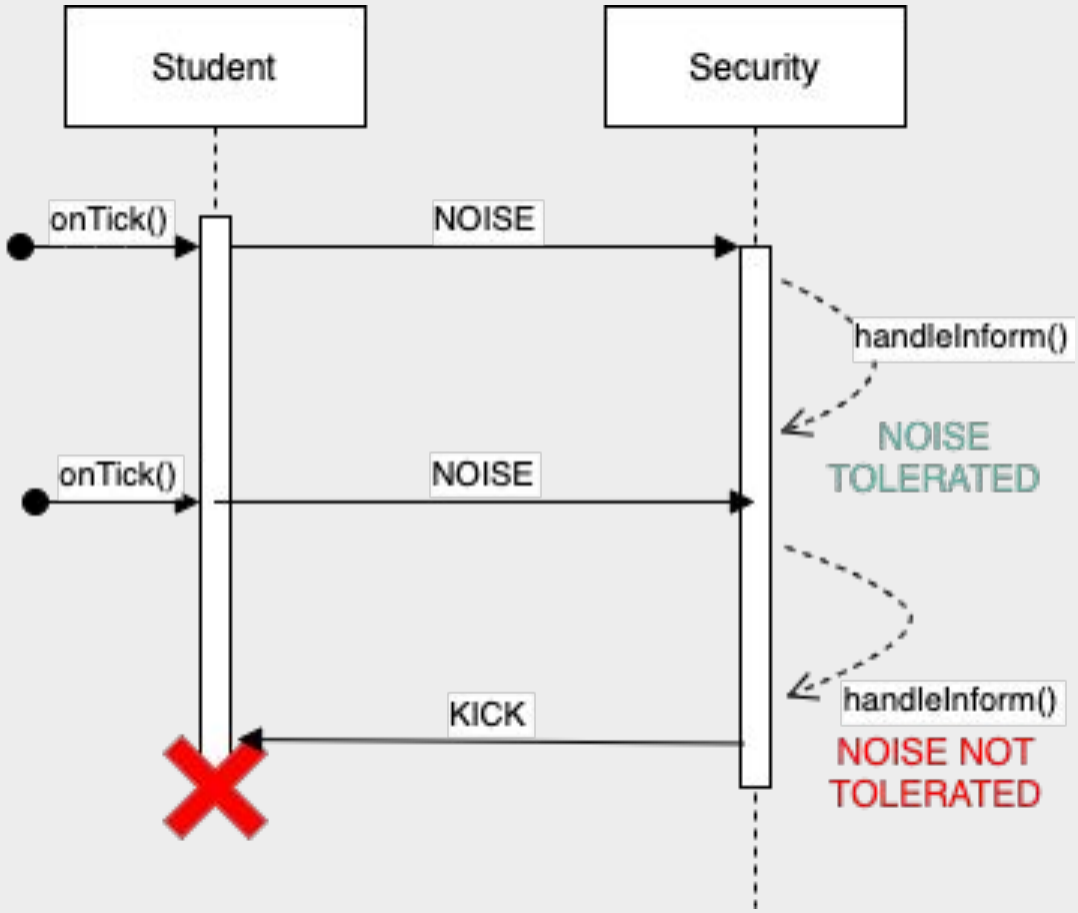
2. Scheme representation for Table Request



3. Scheme representation for Student-Table interaction



4. Representation of Student-Security interaction



4. Project Structure

The main class (LibraryManager) is located on the library folder and starts by reading a creation file located in the res folder with the following structure:

- ➔ 1st line represents the library working time (in seconds)
- ➔ 2nd line has two numbers: <num_of_floors> <av_security_noise_toleramce>
- ➔ The following <num_of_floors> lines have two values: <number_of_tables> <floor_course>;
- ➔ After that the next line has two numbers: <num_students> <av_std_noise>
- ➔ The following y lines have 2 values: <student_name> <student_course>

4. Project Structure

This file is then parsed in the Library class, the floors and the agents are created according to the creation file.

All agents are registered on Directory Facilitator (“Yellow Pages”) on setup.

Agents can be found in the respective folders:

- The Librarian is used to interpret Students requests;
- Each Security is allocated to a floor, has a tolerance to noise (probability of kicking a student making noise) and has the ability do see if any table on his floor is free and to kick Students making noise;
- Students have a name, course, probability of making noise value and time of arrival generated randomly (meaning that the times between each student arrival is different);
- Each Table agent is allocated to a floor, knows if it is free of occupied and has a satisfaction degree based in if the student sitting there has the same course as that floor.

5. Communication

Communication between the agents is dealt by their respective behaviours that can be found on the agentBehaviours folder:

- The Librarian has a Listen Behaviour (LibrarianListenBehaviour) that is cyclic. This behaviour receives requests from the student, sends the request to the Security and, after receiving each Security's response presents the Student with the best option. Can only communicate with one student at a time.
- Security has a Listen Behaviour (SecurityListenBehaviour) that is also cyclic. This behaviour receives the Librarian's request, contacts the Tables in order to see if there is free space to accept the Student, calculates it's satisfaction in accepting the student in that floor and sends that information to the Librarian agent.
- The Tables also have a cyclic Listen Behaviour (TableListenBehaviour) that uppon receiving a request from the Security responds with it's status (free of not). This Behaviour is also responsible for the interaction Student-Table.

5. Communication

Unlike the other agents, the student has 3 associated behaviours:

- StudentRequestBehaviour that is OneShot, and sends the initial request to the Librarian;
- StudentListenBehaviour that is Cyclic. This behaviour is responsible for accepting the floor recommended by the librarian, contacting the tables of that floor in order to find a seat and handling the Security ‘kick’ orders.
- StudentNoiseBehaviour that is a Ticker Behaviour. This behaviour handles the noise generation from the student at each tick and informs that floor’s Security.

Each behaviour outputs it’s messages in the logs folder.

5. Results and Conclusion (1/2)

student noise \ security n. tolerance	2	5	8
2	Kicks: 169	134	67
	Users: 269	234	167
	Sat: 72%	68%	80%
	Occ: 87%	87%	88%
5	Kicks: 408	278	100
	Users: 500	378	200
	Sat: 92%	63%	72%
	Occ: 74%	88%	88%
8	Kicks: 439	378	82
	Users: 500	478	182
	Sat: 100%	76%	79%
	Occ: 49%	80%	87%

Analysing this table (an experiment run for 30 seconds with 5 floors, 20 tables per floor and 500 students) we can see that if Securities are less tolerant to noise (1st column) more Students can use the Library and with higher satisfaction (based on the floor's course). However this generates a lot of empty tables.

On the other hand, if the Securities are more tolerant (3rd column) students aren't kicked at a fast enough rate to ensure a satisfactory fluxe of Students in the Library entering a "first come - first served" policy. The Option with a medium Security tolerance and a high rate of student noise ensures an healthy fluxe of new students keeping the quiet ones and sending the loud ones away.

In the end we can say that the best policy is for the Securities to have a moderated noise tolerance!

5. Results and Conclusion (2/2)

number students \ security n. tolerance	2	5	8
100	Kicks: 78	59	68
	Users: 100	100	100
	Sat: 97%	97%	97%
	Occ: 11%	25%	23%
300	Kicks: 250	177	118
	Users: 300	277	218
	Sat: 97%	87%	81%
	Occ: 39%	68%	80%
500	Kicks: 408	278	100
	Users: 500	378	200
	Sat: 92%	63%	72%
	Occ: 74%	88%	88%

Changing the table (an experiment run for 30 seconds with with 5 floors, 20 tables per floor and an average probability of students making noise of 50%) in order to analyze the relation between Number of Students and Security Tolerance, as we can see, a greater number of students gets better results from a lower tolerance, ensuring a flux of students that allows for most of them to achieve a positive result to their table request. On the other hand, having a lower tolerance with a low number of students makes so that a lot of tables remain empty as students are kicked at a greater frequency than needed.

In conclusion we can verify that optimally the tolerance would decrease as the number of students increases to ensure the better flux.

AIAD

Library Management Simulation with Distributed Agents

Additional Information

GROUP 01:
Ricardo Cardoso - 201604686
João Lemos - 201000660
Simão Silva - 201605422

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master's Degree in Informatics and Computing Engineering

1. Detailed Example

The execution's initial configuration is obtained from a text file with the following format:

```
15
5 8
4 MIEIC
4 MIEM
4 MIEGI
4 MIEC
4 MIEQ
100 4
rui MIEIC
ricardo MIEIC
ana MIEIC
francisca MIEM
carolina MIEM
antonio MIEGI
jose MIEC
carlos MIEC
barbara MIEQ
joana MIEQ
```

- The first line is the simulation time, in seconds
- The second has the number of floors and the mean noise tolerance of the floors' securities. It is followed by the description of each floor (number of tables and course)
- Finally, the number of students that will be created, with the mean noise factor is provided. The next lines detail each student's name and course

2. Detailed Example

1. The Student starts by requesting the Librarian for an available seat and receives an AGREE meaning that the agent will start searching:

```
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE SENT REQUEST:
(REQUEST
 :sender  ( agent-identifier :name "003_francisca@172.17.0.1:1099/JADE"  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name librarian@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "MIEM"
 :ontology  TABLE )
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE RECEIVED AGREE FROM ( agent-identifier :name librarian@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
```

2. The Librarian asks all the securities if there are any free tables on their floors, also sending the Student's course. He receives an AGREE in case the Security can process the request:

```
[19:30:19] :: librarian@172.17.0.1:1099/JADE SENT REQUEST:
(REQUEST
 :sender  ( agent-identifier :name librarian@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name security_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name security_0@172.17.0.1:1099/-
JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name security_4@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-
identifier :name security_2@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name security_3@172.17.0.1:1099/JADE  :addresses (sequence http://-
SIMAO-DESKTOP:7778/acc )) )
 :content  "MIEM"
 :ontology  TABLE )

[19:30:19] :: librarian@172.17.0.1:1099/JADE RECEIVED AGREE FROM ( agent-identifier :name security_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
```

3. Detailed Example

3. Each Security asks the Tables from the floor he is responsible whether they are free or not, receiving a CONFIRM or a DISCONFIRM:

```
[19:30:19] :: security_1@172.17.0.1:1099/JADE SENT REQUEST:
(REQUEST
 :sender  ( agent-identifier :name security_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name table_1_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name table_1_0@172.17.0.1:1099/-
JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name table_1_2@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-
identifier :name table_1_3@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "isFree?"
 :ontology  TABLE )
```

```
[19:30:19] :: table_1_0@172.17.0.1:1099/JADE SENT REPLY (CONFIRM
 :sender  ( agent-identifier :name table_1_0@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name security_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "free"
 :reply-with  security_1@172.17.0.1:1099/JADE1605468619442  :ontology  TABLE )
```

4. After that, he will send back to the Librarian its satisfaction on receiving the specified Student, along with the floor he belongs to:

```
[19:30:19] :: security_1@172.17.0.1:1099/JADE SENT REPLY:
(INFORM
 :sender  ( agent-identifier :name security_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name librarian@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "100 1"
 :ontology  SATISFACTION )
```

4. Detailed Example

5. The Librarian processes all the replies from the Securities, chooses the best (the one with higher satisfaction) and informs the Student with the floor where he has to go:

```
[19:30:19] :: librarian@172.17.0.1:1099/JADE SENT REPLY:
(INFORM
 :sender  ( agent-identifier :name librarian@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
 :receiver  (set ( agent-identifier :name "003_francisca@172.17.0.1:1099/JADE"  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "1"
 :ontology  BEST_FLOOR )
```

6. The Student will then request all the table from the best floor if he can seat in one of them, receiving from each a CONFIRM if positive or a DISCONFIRM if negative:

```
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE SENT REQUEST:
(REQUEST
 :receiver  (set ( agent-identifier :name table_1_1@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name table_1_0@172.17.0.1:1099/-
JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-identifier :name table_1_2@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) ( agent-
identifier :name table_1_3@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :ontology  TABLE )
```

```
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE RECEIVED CONFIRM FROM ( agent-identifier :name table_1_0@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
```

5. Detailed Example

7. After that, the Student requests to seat in one of the available tables, receiving a final AGREE from it:

```
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE SENT REQUEST:
(REQUEST
 :receiver  (set ( agent-identifier :name table_1_0@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :content  "MIEM"
 :reply-with  table_1_0@172.17.0.1:1099/JADE1605468619450  :in-reply-to  "003_francisca@172.17.0.1:1099/JADE1605468619450"  :ontology  SEAT )
```

```
[19:30:19] :: 003_francisca@172.17.0.1:1099/JADE RECEIVED AGREE FROM ( agent-identifier :name table_1_0@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
```

8. When some Student makes noise, an INFORM is sent to the Security responsible for the floor he is in and he may or may not kick the Student, depending on a randomized behaviour. Finally, if the Student is kicked the table receives and INFORM from the student to update its state.

```
[20:40:35] :: 026_ana@172.17.0.1:1099/JADE SENT NOISE INFORM:
(INFORM
 :receiver  (set ( agent-identifier :name security_0@172.17.0.1:1099/JADE  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :ontology  NOISE )
```

```
[20:40:35] :: security_0@172.17.0.1:1099/JADE SENT KICK INFORM:
(INFORM
 :receiver  (set ( agent-identifier :name "026_ana@172.17.0.1:1099/JADE"  :addresses (sequence http://SIMAO-DESKTOP:7778/acc )) )
 :reply-with  "026_ana@172.17.0.1:1099/JADE1605472835674"  :ontology  KICK )
```

```
[20:40:35] :: table_0_2@172.17.0.1:1099/JADE RECEIVED INFORM FROM ( agent-identifier :name "026_ana@172.17.0.1:1099/JADE"  :addresses (sequence http://SIMAO-DESKTOP:7778/acc ))
```

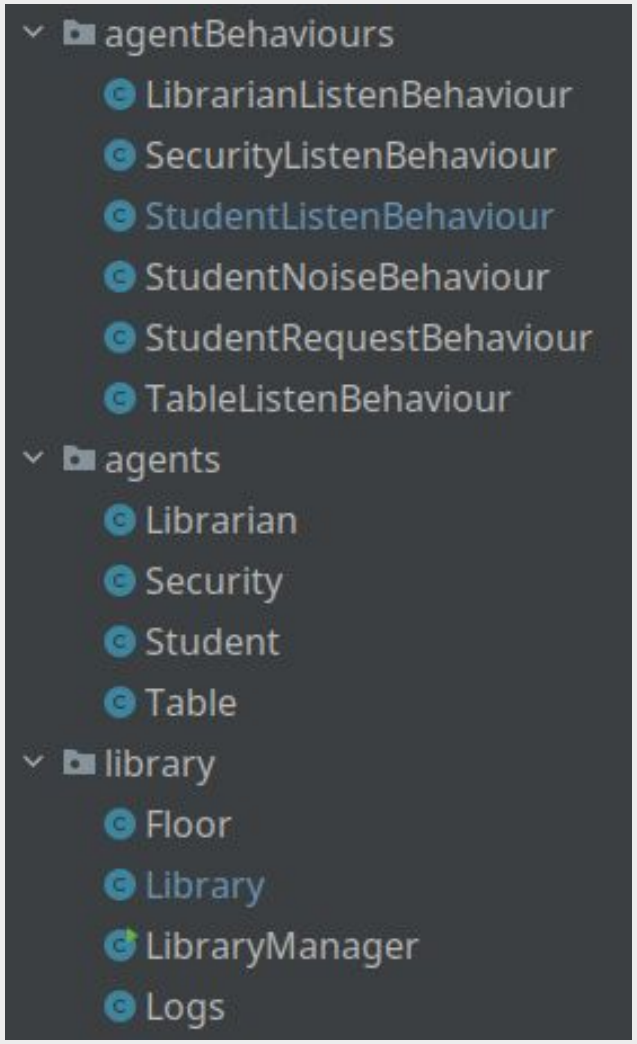
6. Detailed Example

After the execution of the system, a set of statistics is presented to the user for further analysis:

```
========================================
==                                    ==
==              RESULTS               ==
==                                    ==
========================================

-> Working time: 30 seconds
-> Average securities noise tolerance: 5.8
-> Average students noise: 4.886
-> Number of securities kicks: 278
-> Average table satisfaction: 63.45600791714824%
-> Average table occupancy: 87.89108910891085%
```

7. Implemented Classes



In this project we implemented 6 classes of behaviours, 4 classes of agents, and 4 classes of data control.