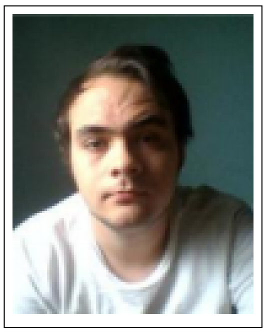
CDIO - 2

Communication with a weight simulator using TCP.

By group 22



Peter El Habr
s165202

Simon Engquist
s143233

Arvid Langsø
s144265

Mikkel Lund
s165238

Jeppe Nielsen
s093905

Mads Stege
s165243

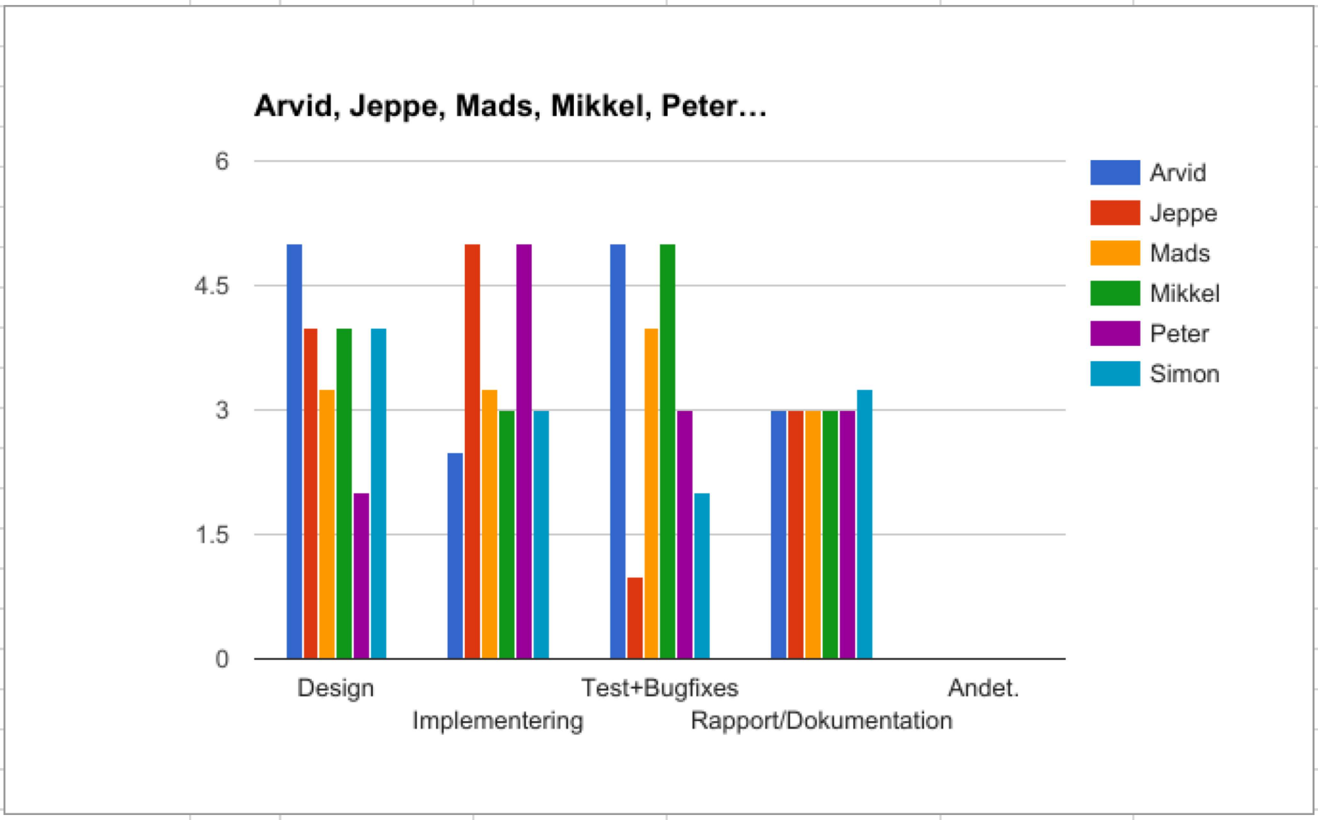
Techincal University of Denmark
DTU Compute

18. of March, 2017 - 05:00 AM

Number of pages (including appendix): 12

Courses: 02324

	Design	Implementering	Test+Bugfixes	Rapport/Dokumentation	Andet.	Total
Arvid	5	2,5	5	3	0	15,5
Jeppe	4	5	1	3	0	13
Mads	3,25	3,25	4	3	0	13,5
Mikkel	4	3	5	3	0	15
Peter	2	5	3	3	0	13
Simon	4	3	2	3,25	0	12,25
Total:	22,25	21,75	20	18,25	0	82,25



Content

1 Introduction

This project describes the design, implementation, and testing of new key features in a distributed weight simulator. The features are meant to function as the commands sent between a user of the weight, and an external server. The inputs tells the weight what functions it should do. The weight in turn prints out commands and confirmations upon these inputs.

Several functions has been implemented in the program as per the assignment description provided.

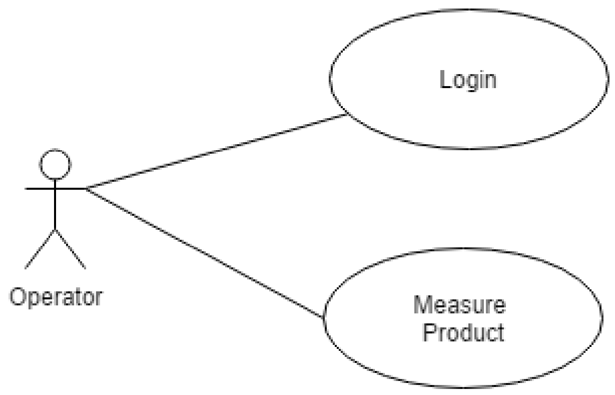
2 Analyses and design

The project was distributed with some of the code for the weight simulator. This code is already somewhat implemented. New code has been added that controls the logic of the simulator. This logic has been implemented so that it behaves like the physical weight. Not all features of the physical weight has been implemented, but all the features the assignment describes has been implemented. The implemented features are sufficient for a simulation of a measurement can be made on the simulator.

Some new features has also been added. You can press the "ZERO" button to restart the current measurement process. You can also press the "EXIT" button to log out of the system.

It has been decided that the socket controller on the weight handles all the incoming inputs from the server. If it can send an answer directly, for instance when receiving a "P111-message, it would answer "P111 A" to the server. In other cases when the weight needs to send information with the answer, the MainController creates the message and send it through the socket controller.

2.1 Use-Cases

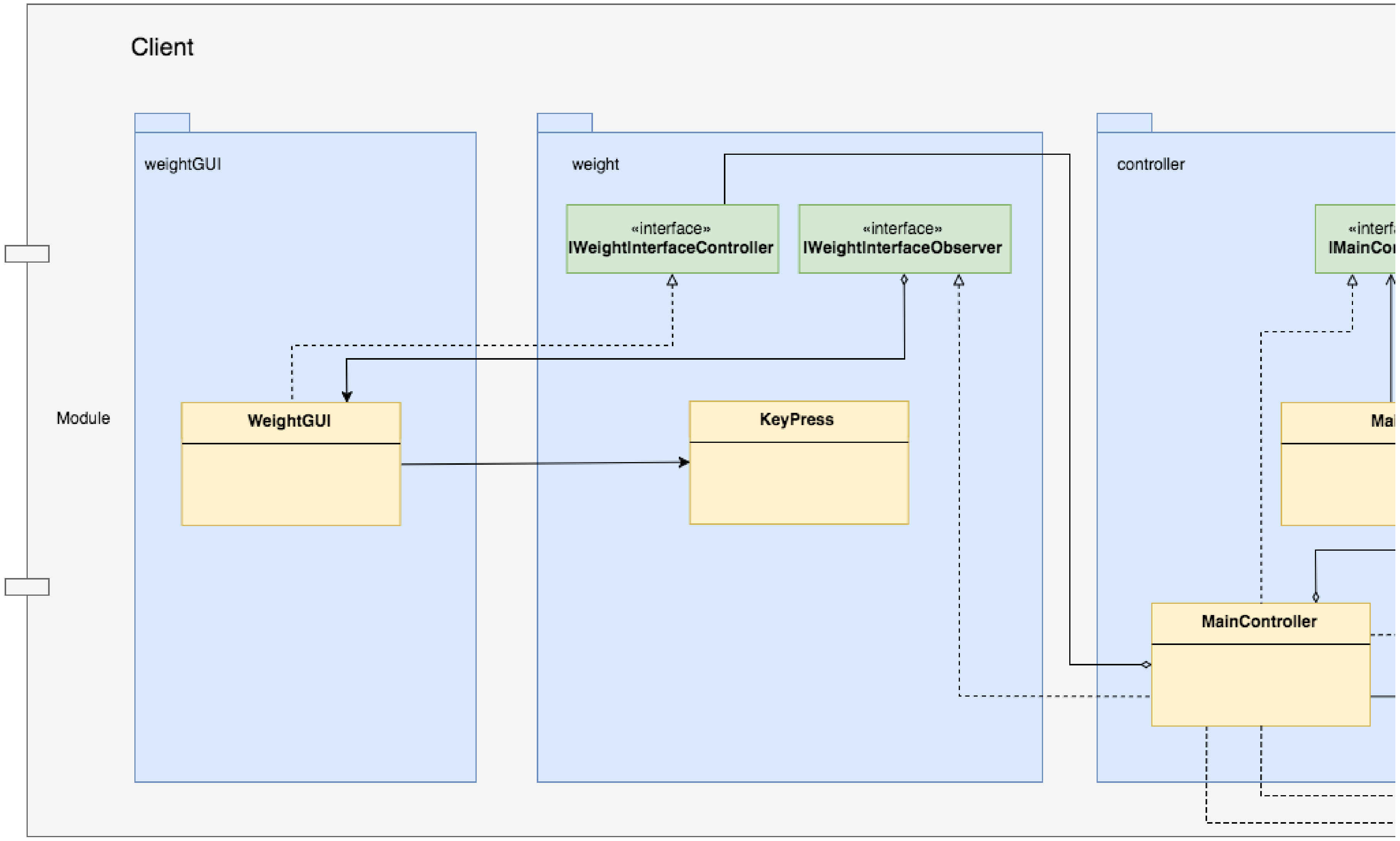


Figur 3: A Use-Case diagram of the weight.

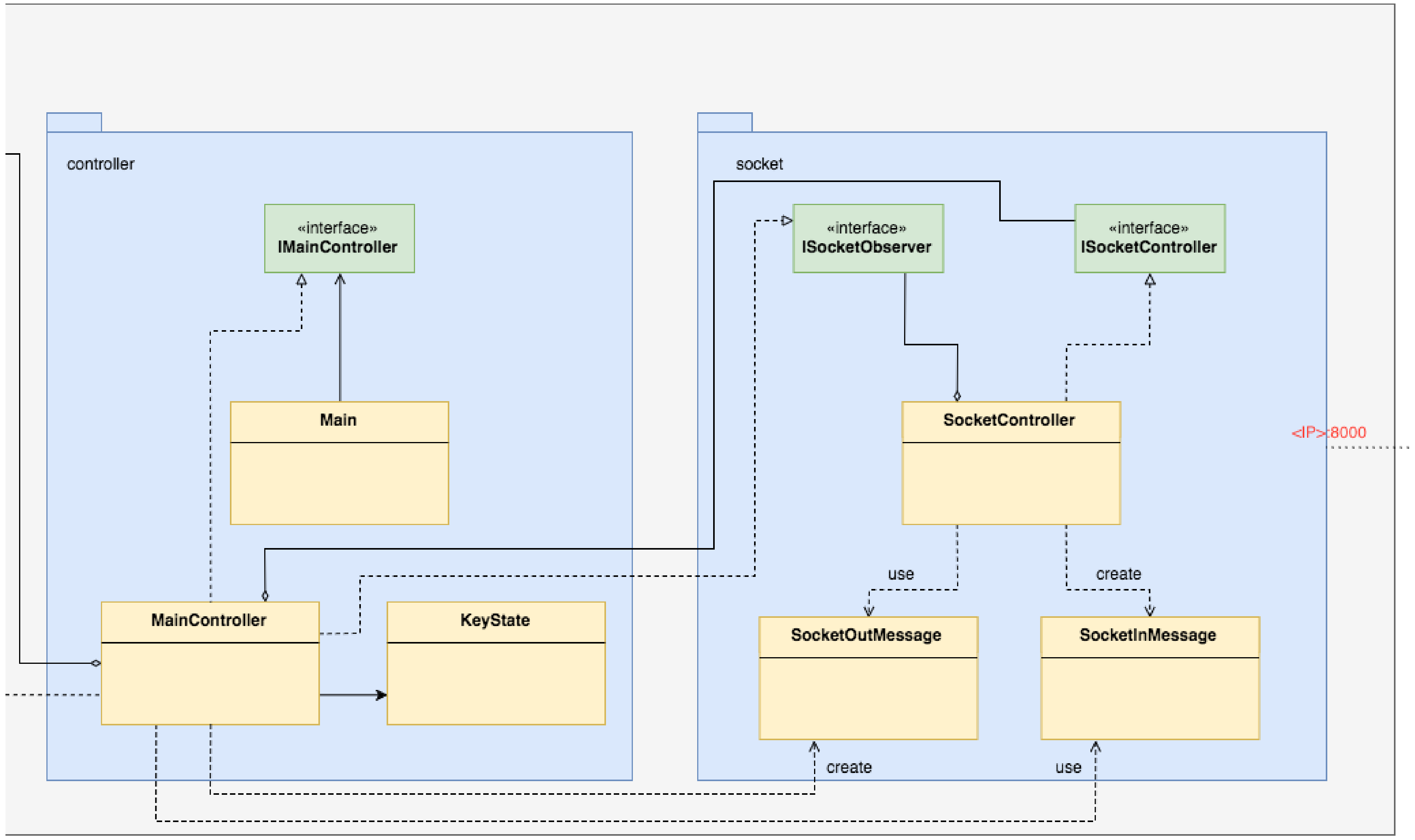
An operator can do two things on the weight. They can log in and they can perform a measurement of a product.

2.2 Class diagram

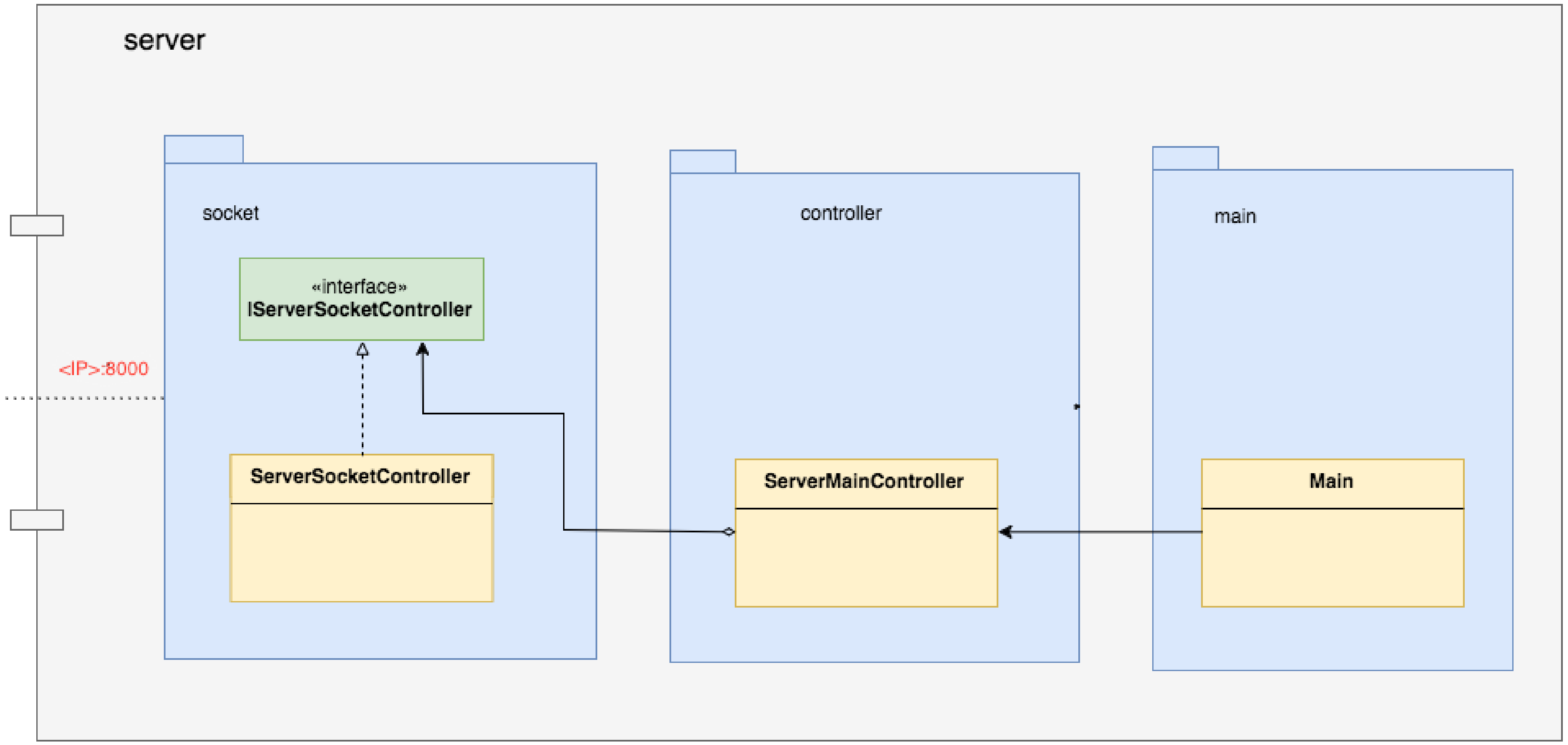
On the following pages you find three different class diagrams:



Figur 4: Class diagram for the GUI of the weight GUI.



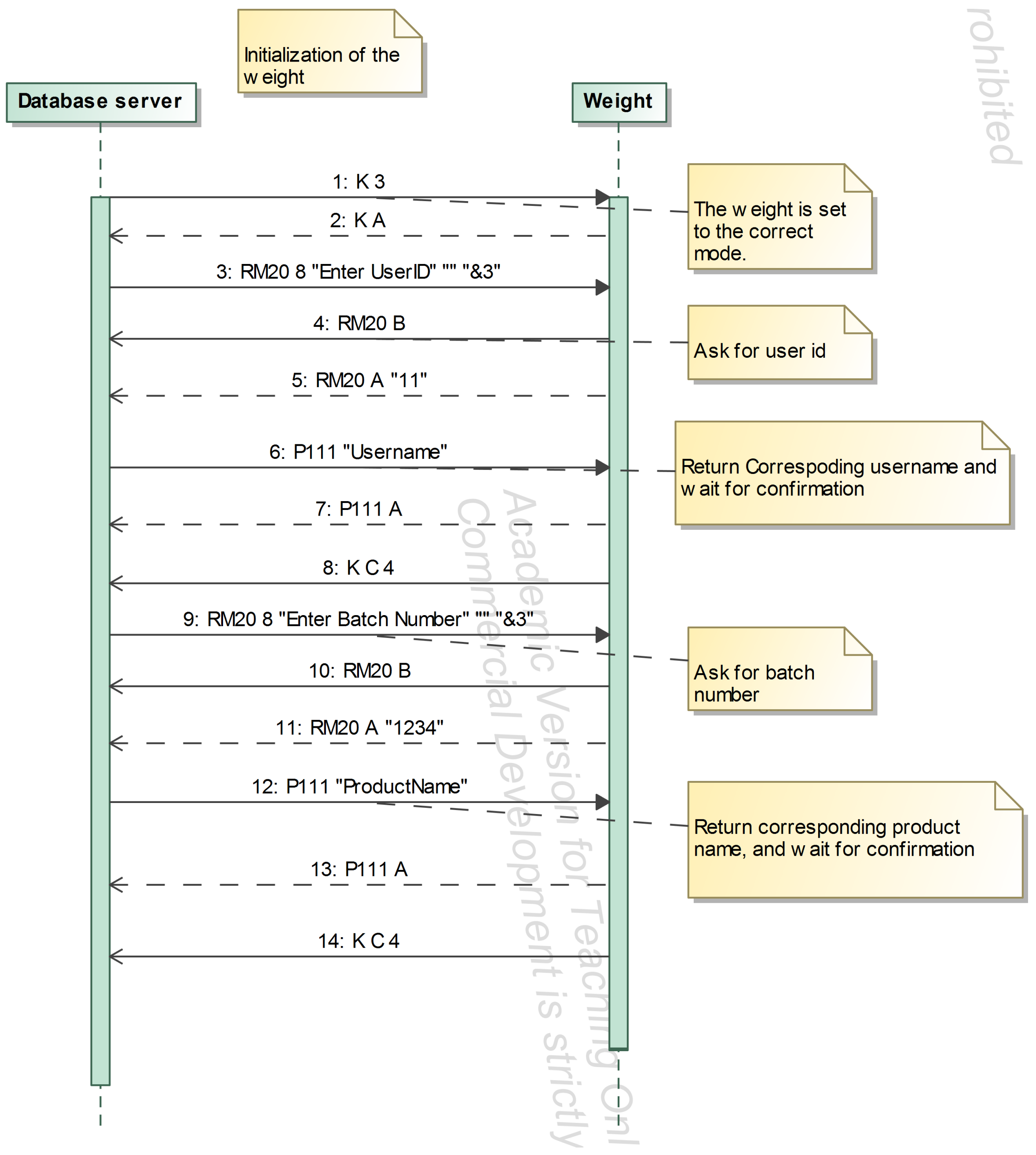
Figur 5: Class diagram for the Main controller and the socket controller of the weight simulator.



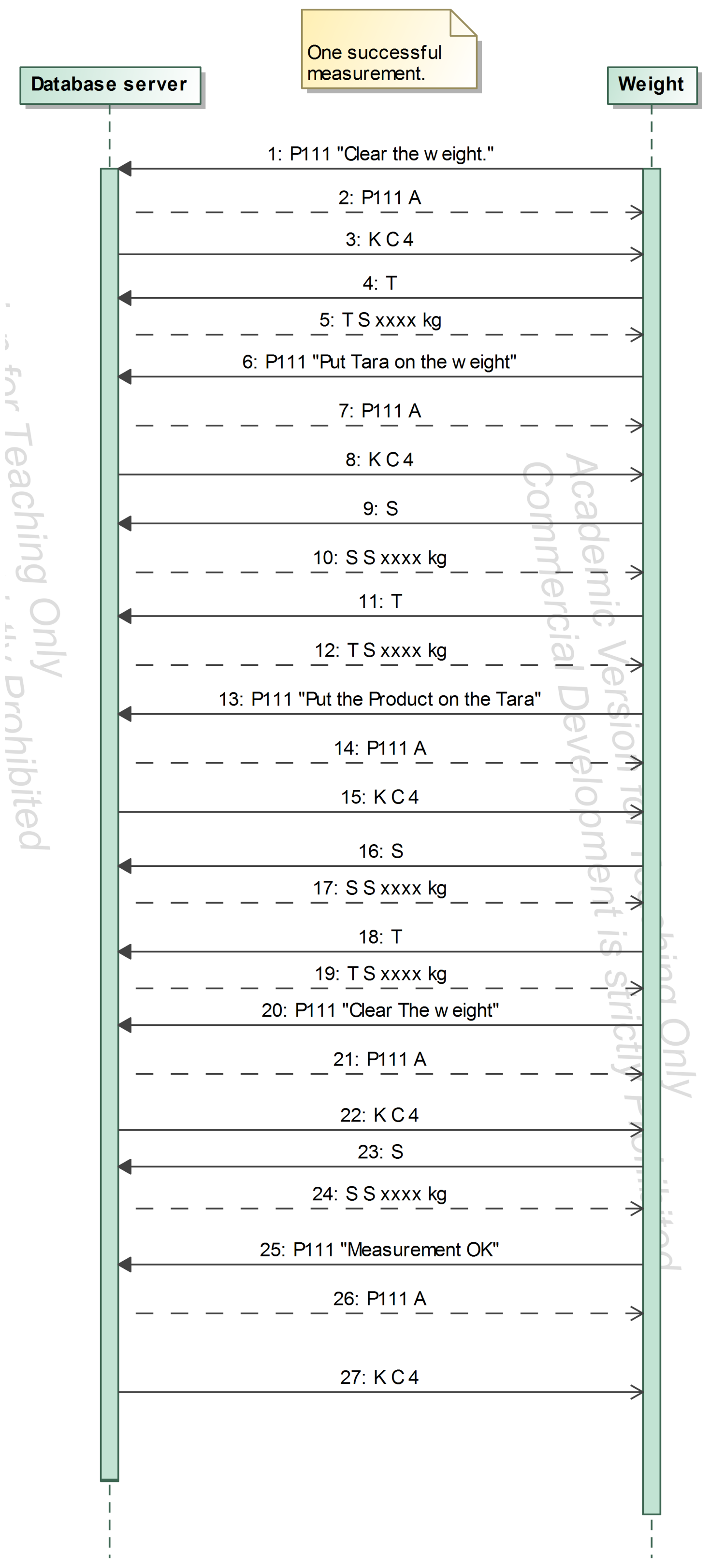
Figur 6: Class diagram for the Server side.

2.3 Measurement protocol

The following diagrams describe the sequences of information sent between the server and the weight. The diagram depicts a successful measurement of a product.


Initialization of the w eight
Database server
Weight
1: K 3
2: K A
3: RM20 8 "Enter UserID" "" "&3"
4: RM20 B
5: RM20 A "11"
6: P111 "Username"
7: P111 A
8: K C 4
9: RM20 8 "Enter Batch Number" "" "&3"
10: RM20 B
11: RM20 A "1234"
12: P111 "ProductName"
13: P111 A
14: K C 4
The w eight is set to the correct mode.
Ask for user id
Return Correspoding username and w ait for confirmation
Ask for batch number
Return corresponding product name, and w ait for confirmation

Figur 7: A sequence diagram describing the communication between weight and server, when logging in.



Figur 8: A sequence diagram describing the communication between weight and server, when weighing a product.

3 Test

The program has been tested with both the weight GUI and the physical weight. It has been tested that all the given commands works in both cases and that the correct messages are shown on the weight. It is tested that when the ”Exit-button is pressed on the weight(gui) the weight logs out the user and returns the weight to the login screen. Furthermore it is tested that when the ”Zero-button is pressed the weight cancels the current weight measurement and returns to the start of a new measurement.

3.1 Socket Packet error

The program encountered a significant error with the physical weight when the user pressed several buttons in quick succession, while the server side still waits for a new packet to be received. The Packet Query is the queue of TCP packets, including commands, that are not yet processed by the server. These accumulate infinitely if an “RM20 I”-error is started by the weight. This error is due to the fact that the weight receives the request but cannot process it. The server side receives this information of “RM20 I” and restarts the loop that returns “RM20” while the weight fails to follow the instructions. This error cause an overload of instructions to the program and the weight, which then run in a never ending loop, preventing the process from continuing to the next step.

The implementation of the flush() method in the ServerSocketController class solves this problem :

- It sends ”RM20 0”and cleans the Packet Query if an error packet is caught from the weight.
- It also sets a delay of two seconds using the following code: ”TimeUnit.SECONDS.sleep (2);”

This solution have been implemented in the program to make it function as expected.

4 Conclusion

During this project, a weight simulator and a command server were successfully implemented. The given code, including the Server Socket, the Controllers and the Graphic User Interface, enabled the development of the weight simulator that approximately imitates the behaviour of a physical weight on the client side.

Two large tests have been carried out, one linking the simulator with the server side responsible for the Client Socket using TCP commands sent to the weight and the other linking the physical weight with the same server side. It has been confirmed during testing that the simulated weight offers similar behaviour to the real weight.

In terms of use of Sockets, the program had to deal with potential errors when receiving packets and the possibility of interruption of the Client-Server connection. The Server part including the Client Socket is therefore equipped with a part of code handling the errors when receiving the answers from the Server Socket, which is the weight. Some errors were very paralysing for the system, like the packet query error. This has been corrected using a system of delays to avoid packet overload.

This project has made it possible to learn more about the use of Sockets and Client to Server communication.