**F21 - ST4SWT**

**Obligatorisk handin 2**

**Chargingstation**

**Afleveret:** 18-03-2021

**Afleveret af**: Gruppe 14

**Deltagere i afleveringen**

|  |  |  |
| --- | --- | --- |
| Studienummer | Navn | Studieretning |
| 201907690 | Lasse Gjetrup Eskildsen | ST |
| 201611717 | Jacob Jungersen Hansen | ST |
| 201905378 | Signe Gram Sand | ST |
| 201908197 | Lars Mølgaard Pedersen | ST |

**Kontaktperson**

|  |  |  |  |
| --- | --- | --- | --- |
| Studienummer | Navn | E-mail | Studieretning |
| 201908197 | Lars Pedersen | 201908197@post.au.dk | ST |

**URL Jenkins:**

<http://ci3.ase.au.dk:8080/job/Team14ObligatoriskOpgave2/>

**URL GitHub:**

<https://github.com/JacobJH/LadeskabTeam14>

# Arkitektur og design:

## Overvejelser ift. arkitektur og design:

* Der er implementeret interfaces til boundary klasserne, som repræsenterer den manglende hardware. Dette gør det muligt at anvende fakes, og derudover giver det mulighed for fremtidigt at kunne implementere hardware-klasser
* Programmets arkitektur vil blive bygget op i 3 lag, efter boundary control entity modellen. Dvs et lag til boundary-klasser, et lag til controller klasser og et lag til entities. Dette gøres for at sikre et naturligt flow af data samt give et godt udgangspunkt for at kunne designe systemet med en lav kobling. Den lave kobling vil give en stor fordel når det kommer til testbarheden af systemet.
* Vi designer systemet så der indgår et Observer Pattern og til dette anvendes der events. Det vil sikre en løs kobling imellem klasserne.

Vi anvender events følgende steder:

* + Håndtering af lukning samt åbning af døren
  + Håndtering af RFID-læseren
  + Udmelding omkring strømtrækniveauet på telefonen

# Strukturering af github:

Da hele systemet bliver forbundet til en Jenkins server, har vi valgt at arbejde med branches når vi implementere ”nye features”. Det betyder i praksis at når vi starter en ny feature branch hver gang vi starter en ny dags arbejde / lektion. Det betyder at vi ikke behøver at få alt uploadet på Jenkins før det er skrevet færdigt og er testet. Når vores feature branch er færdig eller når dagen er slut, vil vi merge branchen med vores Main-branch, og derved få Jenkins til at køre Coverage test, interaktionstest og unit test på hele systemet.

# Opdeling af ansvar:

Udgangspunktet for uddelegeringen af ansvaret bygger på et ønske om at skrive test først og derefter kode klasserne. Derfor vælges det at uddele hver boundary klasse og dertilhørende test og interface til et gruppemedlem. På den måde får alle mulighed for at skrive test til en klasse, som de derefter kan implementere. Efter dette vil vi sammen skrive testene til kontroller klasserne, da disse har behov for interaktionstest såvel som State baseret test, og det vil derfor give god mening at lave disse når boundary klasserne og deres interfaces er lavet.

Fordelen ved at dele opgaverne ud på denne måde er at det sikres at vi ikke ved et uheld kommer til at skrive kode oveni hinanden og derved har lavet dobbeltarbejde. Derudover vil det gøre det lettere at skrive klasserne, hvis man selv har skrevet testen som hører til, i modsætning til hvis man skulle implementere klasser ud fra test som man ikke selv har skrevet.

Vi ønsker at øve os i testdriven Development og derfor er vores vigtigste prioritet at testene skrives først og koden derefter. Denne måde at opleve arbejdet på sikrer at det er muligt.

Begrænsningerne ved denne måde er blandt andet at nogle af gruppemedlemmerne muligvis vil ende med at større arbejde end andre, fordi de enten koder hurtigere eller har fået mindre klasser som ansvarsområder.

# Refleksion over arbejdsfordeling og continuos integration: