Traffic Lights, assignment 3/4/5

2018-09-19

Contents

[1. Requirements 2](#_Toc525160400)

[2. Design 2](#_Toc525160401)

[2.1 Considerations 2](#_Toc525160402)

[2.2 Idea preview 2](#_Toc525160403)

[2.3 Technology overview 4](#_Toc525160404)

[2.4 Run application 5](#_Toc525160405)

[2.4.1 Run as is 5](#_Toc525160406)

[2.4.2 Manually in IDE 5](#_Toc525160407)

[2.4.3 Assignment 3 5](#_Toc525160408)

[2.4.4 Modified configuration 5](#_Toc525160409)

[2.4.5 Rebuild 5](#_Toc525160410)

[2.5 DB structure 6](#_Toc525160411)

[2.6 Classes interaction 8](#_Toc525160412)

# Requirements

3.Write a small java program, based on the spring stack, to simulate a single set of traffic lights, along with state transitions. The state transition should happen automatically every two seconds. To display the state, you can use log4j statements or other means. Please include a test suite and treat as if this is production code

5.Enhance solution from 3 "traffic lights" to operate from an in-memory database, for example H2. The enhanced program should include a programmable duration of light phases based on time of day, and day of week. Please include a test suite and treat as if this is production code.

Note: You can complete either Question 4 or 5 above, if you lean more towards front end development or backend development, or you complete both if you feel equally proficient. We won’t hold it against you either way.

# Design

## Considerations

1. What do you mean by “single set of traffic lights”.

Is it 1 semaphore with lights: red/yellow/green

Or set of semaphores in crossroad

I assumed, this is about set of semaphores in crossroad,

As this task looked more difficult :P

I will leave the easier one for juniors

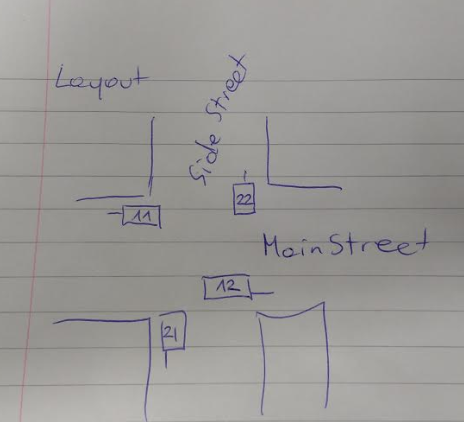
## Idea preview

MainStreet + SideStreet have Semaphore instances connected,

MainStreet11 + MainStreet12 semaphores are MainStreetType

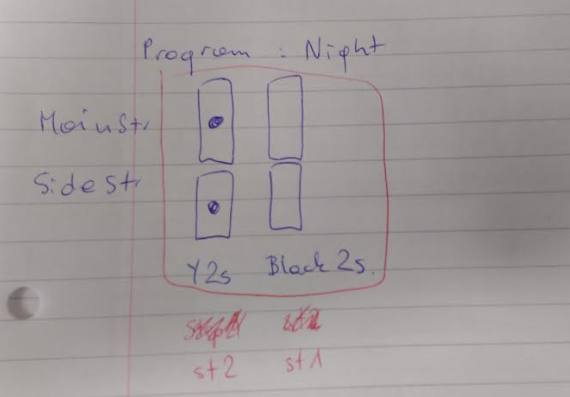
SideStreet21 + SideStreet22 semaphores are SideStreetType

(more instances of the same SemaphoreType can be added easily)



TrafficLightsApp is initialized with DefaultProgram, representing NightMode.

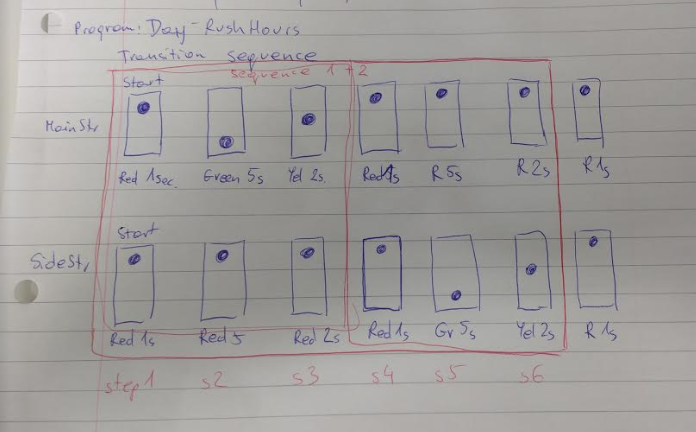
NightMode iteration contains 2 steps (Yellow + Black)



Other Programs are loaded from database at startup (caching mechanism will be added in Phase2)

RushHoursProgram contains 6 steps

WeekendProgram contains 6 steps





Application has to run the full iteration, then Program can be changed.

This means there might be a delay of a full iteration length,

Then new Program starts from Step1.

Changing Program follow Schedule configured in database

## Technology overview

* Spring Boot + Tomcat
* Configuration in database H2-server
* Configuration is loaded in **StartUpInit** class
* Semaphore instances are registered in **SemaphoreRegistry**
* Light change is maintained in **LightManager**, pushed to semaphore instances using Observer pattern, implemented by **PropertyChangeListener**
* Current light status is printed to logs independently, configured in **SemaphoreVisitor**, using Visitor pattern

## Run application

### Run as is

To run application use scripts prepared:

* 1\_runDatabase.bat (Keep database running in background)
* 2\_runTrafficLightsApp.bat

This should run without any changes, as checked out from repository.

### Manually in IDE

Open IDE in trafficlight/sourcecode/trafficlights5 folder

Run TrafficlightsApplication class

And watch console

Logs are stored in **logs** folder

### Assignment 3

Task3 sample run is configured in **TrafficLightsTask3Test**,

run as Junit and watch console

### Modified configuration

Configuration can be modified in database (use your favourite tool)

**jdbc:h2:tcp://localhost/c:/iwona/trafficlight/h2\_database/dbcontent**

user: “sa”, password: empty

Then rerun app

### Rebuild

Go to trafficlight/sourcecode/trafficlights5 folder

Run commandLine: mvn clean install

## DB structure

Full configuration is kept in database.

Please review the table structure, it should be intuitive.

**Testing different options:**

1. To configure new program, populate Program + Config + Steps accordingly.
2. Add Schedule record with priority.

Priority 1 is the most important. Assuming they don’t overlap.

(Configuration validation will be added in Phase2).

PROGRAM

Id

…

STEPS

Id

ProgramId

Sequence

Duration

STEP\_CONF

Id

StepId

SemaphoreTypeId

ColourId

SCHEDULE

Id

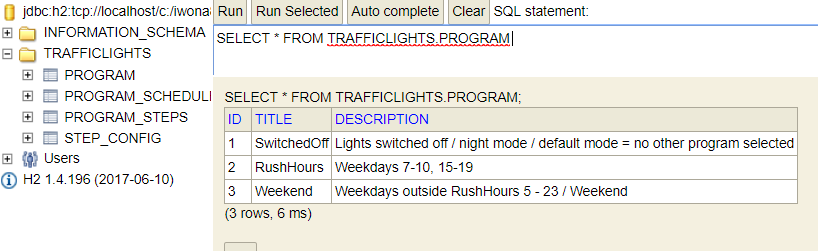
ProgramId

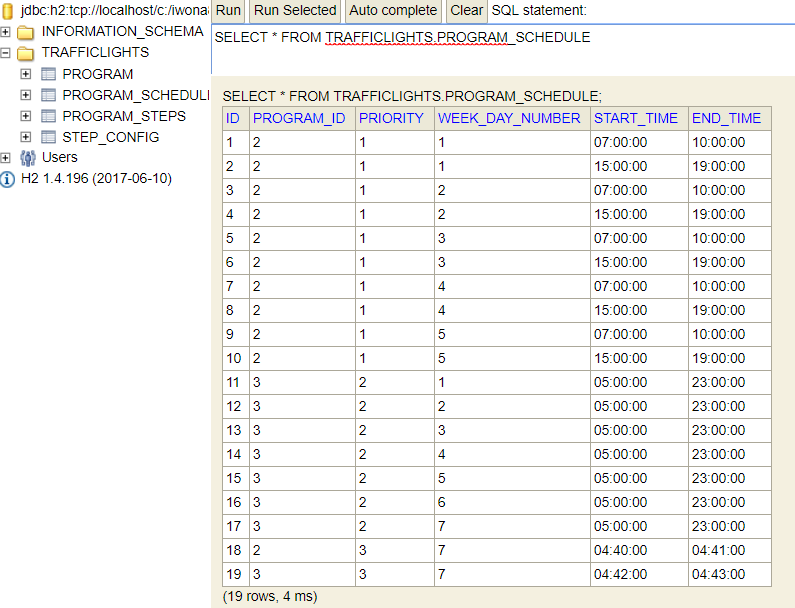
Priority

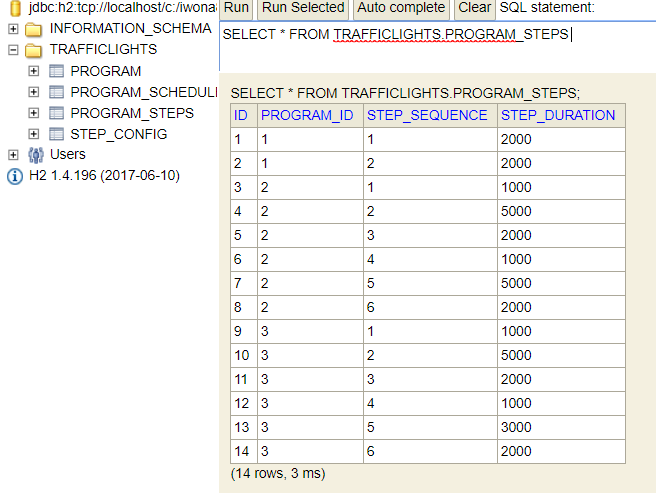
WeekDayNumber

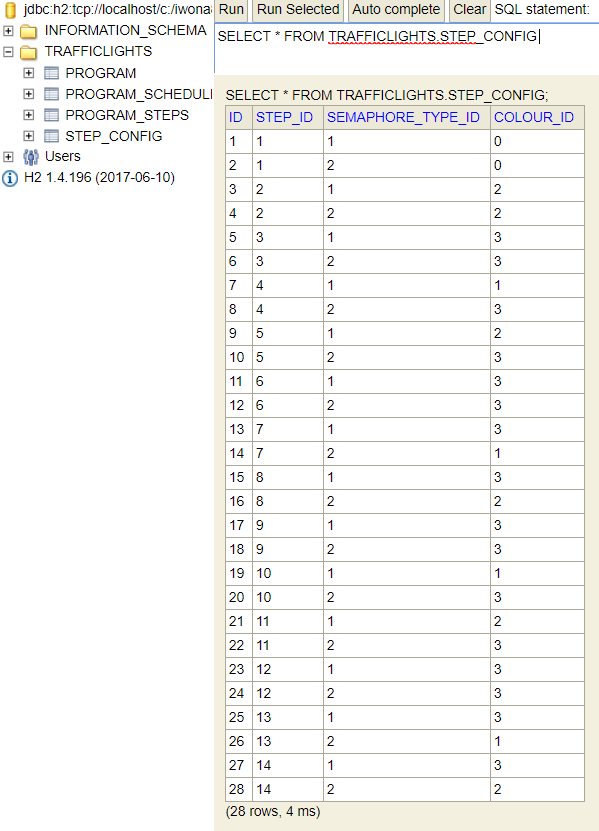
StartTime

EndTime









## Classes interaction

Sxdfsdf

StartUpInit

loadConfig

initialize

register Semaphores to observe LightManager

register SemaphoreWatcher

SemaphoreRegistry

Init instances

Program

Configuration

SemaphoreVisitor

Watch Semaphore status

LightManager

Setting CurrentProgram

Push to instances

Semaphore Instances

ProgramScheduler

Set NextProgram

Semaphore Instances

Semaphore Instances