## 2022-02-14:

Happy Valentines Day!

Kick off the Traffic Lights project!

First task:

- find updated drafts here: http://win-router/ --> home / Programming Veronika - HTML and JS projects / 2021-07-20 - Traffic Lights.pdf

Direct link:

<http://win-router/api/content?path=home/Programming/Veronika%20-%20HTML%20and%20JS%20projects/2021-07-20%20-%20Traffic%20Lights.pdf>

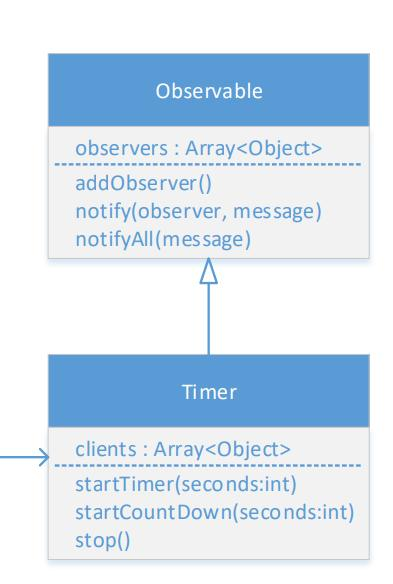
- explore working diagram at p.2 and class diagram at p.3

- create Timer class that implements Observable Pattern in JavaScript (Timer extends Observable class)

Helpful links to look at

-- <https://webdevstudios.com/2019/02/19/observable-pattern-in-javascript/>

-- <https://blog.bitsrc.io/the-observer-pattern-in-javascript-the-key-to-a-reactive-behavior-f28236e50e10>



доделки сегодня:

1. index.html и style.css надо положить в папку "client"

2. твой script.js надо переименовать в TrafficLights и положить в папку server

3. в папке server создай server.js файл и добавь в него только то, что нужно для запуска node back-end server:

- express

- cors

- app.get() - хотя бы один end-point "GET /" который вернет response "Hello!"

- app.listen(.....)

если нужна подсказка - смотри в server.js в albums-app

4. мувни this.stop() из startTimer() в startCountDown()

## 2022-02-17:

Очень подробно, надеюсь это очень сольно облегчает выполнение задачи, и Вероника ценит мои старания для нее.

а то я тоже трачу на это очень много времени чтоб так подробно все описать. о да еще и на понятном языке (wink)

- use debugger when needed

- pull branch (i did couple of amendments)

- Rename TrafficLights.json into config.json

- create Channel class:

this is a "data-structure" class to hold data

that comes from the "channel" part in config.json:

{

"name": "Glasgow Road",

"trafficLights": ["T1", "T3"]

}

attribute fields in the Channel class must be private

so that no one could change them in

the existing instance of the class, once it's been created.

Channel class must have:

- appropriate constructor to set its private fields

- getters for both of its attributes

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get#get_vs._defineproperty>

- setters for both of its attributes

setter is similar to getter, but it receives value and checks if value qualifies.

-- for `name` value must be any non empty String.

-- for `trafficLights` must be non empty Array

examples for getters and setters in js Class here:

https://stackoverflow.com/questions/49895080/javascript-class-getter-setter/49895210#49895210

- inside Controller class

this.config field must be private - no one must have access to it

once config.json file is loaded

read `channels` from config and

add channels (correct Channel instances) into this.channels Map

with their keys as they go in config.json

----------------------------

result of all above:

after you've done below:

const controller = new Controller();

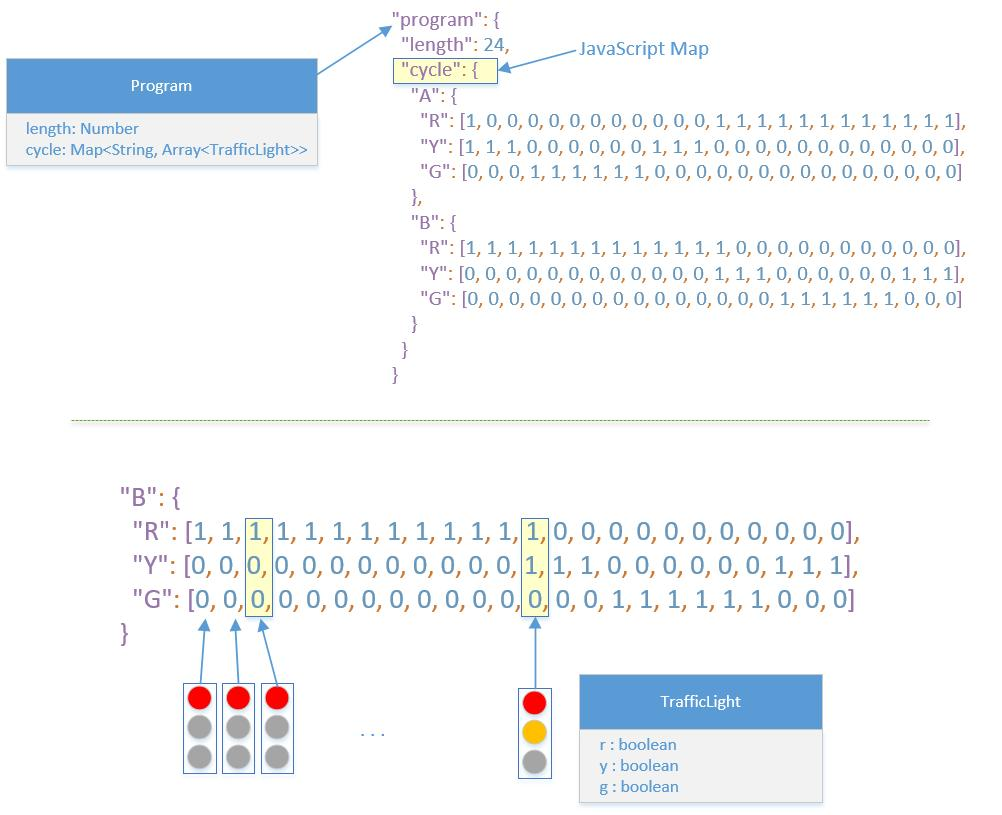
controller obj. should have the `channels` attribute

type of Map, and it should be filled in with the 2 channels:

controller.channels.get("A") --> should return object type of Channel

with name = "Glasgow Road"

## 2022-02-20:



## 2022-02-21:

текста много (потому что расписываю все очень детально),

но работы меньше чем вчера.

Иди по порядку сверху вниз, не перепрыгивай.

некоторые пункты связаны между собой.

- check and merge PR #3 at github in

Traffic-Light-System

this is a PR from my branch into your branch (feature/controller),

what do you need to do then, after merging it?

- for the below 2 methods:

Controller.loadChannels

Controller.loadProgram

1. make private (nobody can't use them from outside

if they want to replace channels and program with new values)

2. make them return value and assign that to

this.channels and this.program accordingly.

- make channels and program also private.

no need in getters. these are internal

only for Controller class.

no one else supposed to use them

- in class Program create method getCycle()

that accepts parameter i (integer Number)

and returns a Map of the same kyes that

already exist in Program.cycle and

each value of the Map entry should have one single Traffic Light,

taken from that i-position of the Program.cycle Array

(look at the UML diagram, or debug if needed)

- in Model.js create State class

State has 2 private properties: "timestamp" and "state"

create getters for them, no setters needed,

they should be set through the class constructor.

\* timestamp: is a String, which holds Date and Time (from `now`)

in this format: `yyyy-MM-dd HH:mm:ss` (see main.js).

set it in the constructor.

\* state: is going to be a Map that will be created from the 2 constructor parameters.

create a method for that, similar to Controller.load<Something> - make a placeholder

for this method for now, and we'll implement it next time.

so, constructor of the State class accepts 2 params:

- channels - Object of the class Channels

- cycle - Map that is returned from Program.getCycle() method.

## 2022-02-22:

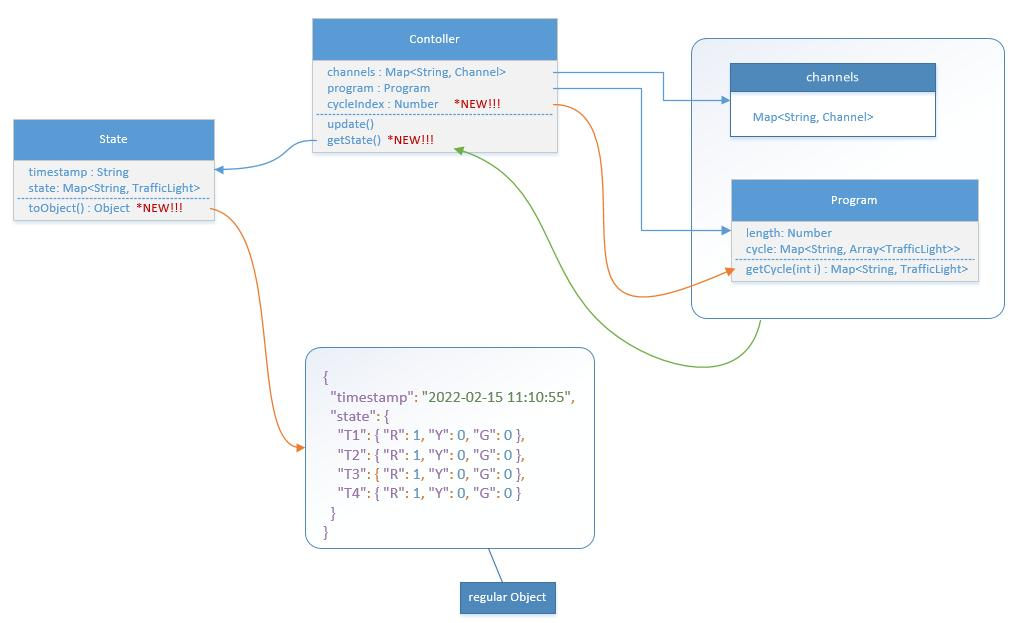
- look at the screenshot below, find "\*NEW!!!" marks - this is what you need to implement today

Controller.cycleIndex should hold a number - current index of the program cycle. Default value is 0

Controller.getState() should take the cycleIndex and use it for getting "Cycle" from program. then result of that along with channels should be used for constructing a new State object, that we have created last night.

- State class should have method toObject() (this is a public method).

this method should create a plain object out of the attributes of the State. see the screenshot.



- you will get £5 for one more if you change also Controller.update() method to make it contain only one single line of code that updates cycleindex. if you do this, here is the free tip: your should use % operator in that line.

requirements:

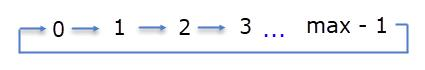
\* min and default value is 0, max value is program.length - 1 (the max available index in the program cycle)

\* every time you call Controller.update() it increases Controller.cycleIndex by 1, up until it reaches the max possible value: program.length, if reached, then it should start with the 0 again.

\* there should not be any "if-else" or "for loops" in Controller.update() method

\* it must contain only one line, approximately 50....60 symbols long. and one of the symbols is %

the purpose of it is: updating Controller.cycleIndex in a cyclic way:



## 2022-03-08:

Traffic Light Controller:

закончить со State классом

объект сласса State ты создаешь примерно так:

new State(channels, cycle)

что такое "channels"?

это Map:

{

A => Channel { name, trafficLights: ["T1, T3"]},

B => Channel { name, trafficLights: ["T2, T4"]}

}

что такое "cycle"?

это тоже Map:

{

A => TrafficLight,

B => TrafficLight

}

а чем должно быть this.#state в объекте State?

тоже Map:

"state": {

"T1" => TrafficLight,

"T2" => TrafficLight,

"T3" => TrafficLight,

"T4" => TrafficLight

}

Бизнес логика:

- проходишь по ключам в channels

- для каждого ключа из cycle достаешь состояние TrafficLight

- из channels для этого ключа также берешь коды [T1, T2, ... ]

и для всех них подставляешь в channels нужной объект TrafficLight который берется из cycle

## 2022-03-11:

- review PR #5

https://github.com/vvp-coding/Traffic-Light-System/pull/5

Tell me what files were changed and what the changes are about?

What is the logic behind the changes?

- add additional logic into Controller.getState()

The Logic:

if controller.status is STOPPED:

use all "red traffic lights" in cycle

for all channels

reminder of what is cycle and how do you use it there:

new State(channels, cycle)

else:

use this.#program.getCycle(this.cycleIndex)

as it's in there right now.

- add console log into Controller.update() method

and log datetime, formatted as `yyyy-MM-dd HH:mm:ss`,

and cycle index number

## 2022-03-16:

- create TrafficLight component - only one component which might have its position parametrized

Direction: N, S, W, E (refer to the screenshot below)

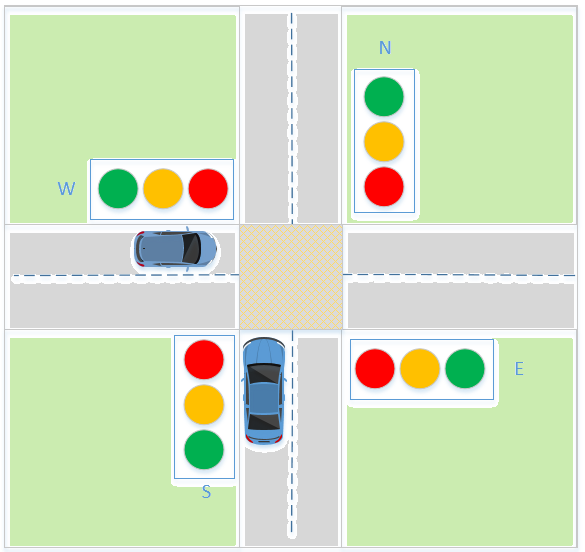
Other requirements:

- Traffic Light's body is always of the same size.

- The Order of colours of its signals should depend on its direction

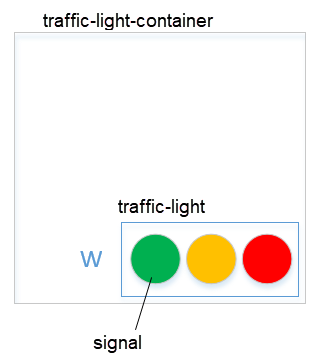
- The positioning inside the parent element should also depend on its direction (use css flexbox for that):

<https://css-tricks.com/snippets/css/a-guide-to-flexbox>



подсказка:

самое важное в решении этой задачи - подумать над LayOut твоего TrafficLight Сomponent:



## 2022-03-18:

- review and merge PR #7

once merged remove .....\Traffic-Light-System\client\node\_modules folder

then from root of the project run `yarn install`

- make body of a traffic light black

- make color of each signal in a Traffic Light grey by default

(but do not delete .red, .green, .yellow classes)

- Font-End changes:

1. Grass and TrafficLight:

Grass - should not have direction.

Traffic light should have the direction,

so refactor the code

so that instead of this:

<Grass direction="W" />

you would have this:

<Grass>

<TrafficLight direction="W" />

</Grass>

again,

look at the last image i've posted yesterday,

there is a huge hint for that.

2. Fetching Data:

when page is loaded start asynchronously Timer (functionality to be followed below).

How you gonna do that?

What makes you know in React that page is fully loaded?

Timer's callback functionality:

- timer's interval = 1000ms

- every time fetch data from server's "GET /state" api.

- render page by sending necessary bits of data to each of the traffic lights

2022-03-19:

- set null as default value for data state

- in fetch check if result returns OK status

if yes then return res.json()

otherwise throw error

- setInterval returns some Id (name it correctly!)

use effect must return a callback that clears that id

- render TrafficLight only if data state is not null or empty

- red is not gray!!!

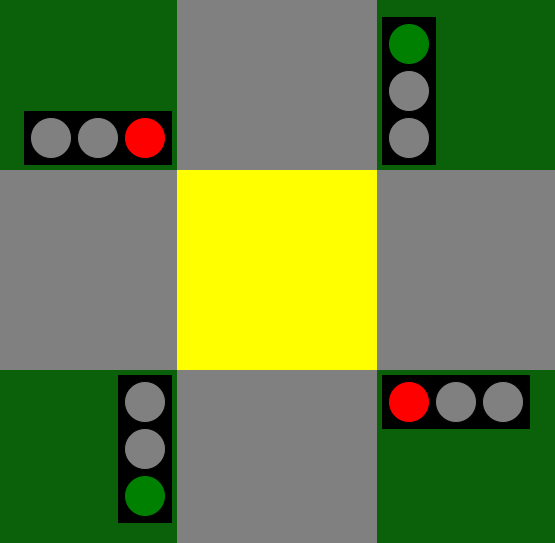
.red {

background-color: grey;

}

but default color of each signal is gray

- correct other small mistakes to make front-end look correct



## 2022-03-21:

- write a read.me file

look carefully into the 1st link below - it explains the structure of you readme file.

use images in your readme, you can take them from TrafficLights.pdf

read me must have:

- Pictures of intersections with cars

- Logical Diagram

- Working Diagram

- UML Class Diagram

it must have sample of json response from back-end server as well.

instructions:

- <https://www.freecodecamp.org/news/how-to-write-a-good-readme-file/>

- <https://www.makeareadme.com/>

- <https://medium.com/@saumya.ranjan/how-to-write-a-readme-md-file-markdown-file-20cb7cbcd6f>

- <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

suggestion/advice on the above:

try to create a content for the readme.md file in some Word document.

then put it into readme.md file using necessary markdown syntax

create a separate branch for that and do everything locally.

when readme file is ready i'll be able to commit and merge it as usual.

VSCode helps you to preview your editing <https://code.visualstudio.com/docs/languages/markdown>