Question 1 [4 points]

We call a vehicle **VIP** if its identifier ends with a special sequence of characters (given as parameter **"viptag"** -- see below). For example, if this special sequence of characters is **"_vip"**, then vehicles with identifiers like **"v1_vip"** and **"car_vip"** are **VIP** vehicles.

Add a ***LightSwitching Strategy*** to the simulator such that in each call to **chooseNextGreen** it gives green light to the first incoming road whose queue has **VIP** vehicle(s), using a circular search starting from **the position of the current road** with a green light. If all lights are red, then the search starts from position 0. In the case that no incoming has a **VIP** vehicle, it behaves **RoundRobinStrategy** with a given *time slot*. Note that the *time slot* is not used if there are **VIP** vehicle(s), it is only used with the **RoundRobinStrategy** when there are no **VIP** vehicle(s).

The **JSON** input corresponding to this strategy has the following syntax:

```
{
  "type" : "vip_lss",
  "data" : { "viptag" : "_vip", "timeslot" : 5 }
}
```

where

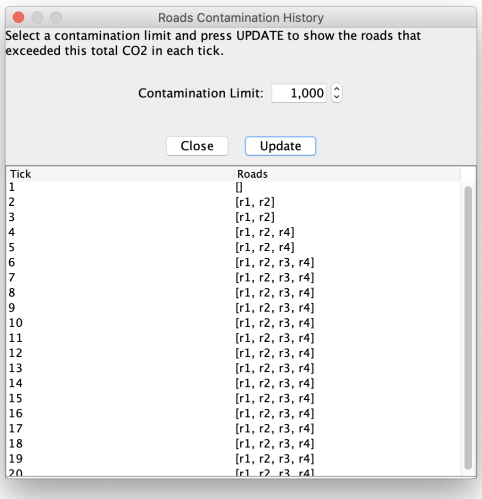
1. The value of **"viptag"** is the special sequence of characters that identify **VIP** vehicles. It is optional, with a default value **"vip"** (without the underscore).
2. The value of **"timeslot"** is a positive integer. It is optional, with default value 1.

Together with this **PDF**, you have a **JSON** file that you can use to test your implementation. You can use `s1.endsWith(s2)` to check if `String s1` ends with a `String s2`.

In your solution, you can add new classes but you are not allowed to modify existing ones except the `Main` class. A solution that does not respect this requirement will be given 0 points.

Question 2 [6 points]

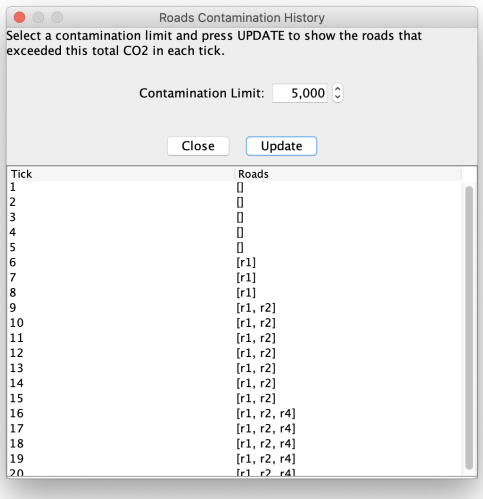
Add a new button to the **`ControlPanel`**, such that when clicked it opens a Dialog with the following aspect:



Select a contamination limit and press UPDATE to show the roads that exceeded this total CO2 in each tick.

Contamination Limit: 1,000

Tick	Roads
1	[]
2	[r1, r2]
3	[r1, r2]
4	[r1, r2, r4]
5	[r1, r2, r4]
6	[r1, r2, r3, r4]
7	[r1, r2, r3, r4]
8	[r1, r2, r3, r4]
9	[r1, r2, r3, r4]
10	[r1, r2, r3, r4]
11	[r1, r2, r3, r4]
12	[r1, r2, r3, r4]
13	[r1, r2, r3, r4]
14	[r1, r2, r3, r4]
15	[r1, r2, r3, r4]
16	[r1, r2, r3, r4]
17	[r1, r2, r3, r4]
18	[r1, r2, r3, r4]
19	[r1, r2, r3, r4]
20	[r1, r2, r3, r4]



Select a contamination limit and press UPDATE to show the roads that exceeded this total CO2 in each tick.

Contamination Limit: 5,000

Tick	Roads
1	[]
2	[]
3	[]
4	[]
5	[]
6	[r1]
7	[r1]
8	[r1]
9	[r1, r2]
10	[r1, r2]
11	[r1, r2]
12	[r1, r2]
13	[r1, r2]
14	[r1, r2]
15	[r1, r2]
16	[r1, r2, r4]
17	[r1, r2, r4]
18	[r1, r2, r4]
19	[r1, r2, r4]
20	[r1, r2, r4]

where the user can select a **Contamination Limit** (between `0` and `Integer.MAX_VALUE`), and when **`UPDATE`** is clicked it shows in each row of the table: (1) a tick, i.e., a time; and (2) the identifiers of roads whose total `CO2` exceeded that limit at that time (at the end of the corresponding simulation step). For example, suppose we have executed the simulation `20` ticks (using the **`JSON`** file provided with this question):

1. If we choose `1000` as a limit and click on **`UPDATE`**, the table will be as the one shown on the left, which indicates that: in the first tick no road exceeded the limit; from tick `2` to `3` only roads `r1` and `r2` exceeded the limit; from tick `4` to `5` only roads `r1`, `r2` and `r4` exceeded the limit; and from tick `6` to `20` all roads exceeded the limit.
2. If we choose `5000` as a limit and click on **`UPDATE`**, the table will the one on the right.

When the dialog window opens, the table can be empty or already filled with information for the selected limit (it depends much on how you decided to implement this question, do not worry about this, both solutions are valid).

The new button in the **`ControlPanel`** should be disabled during the simulation and enabled again when the simulation is over or stopped. Together with this **`PDF`**, you have an icon that should be used for the new button, and a **`JSON`** file that you can use to try your implementation (it corresponds to the tables above).

- **When you implement this question, think well of the information you need to store in order to generate the table. Be careful, as if you do not think of this very well you will end up showing the information of the last tick in all rows.**

- **You are not allowed to modify or add any class the model, you can only modify or add classes to the view. A solution that does not respect this will be given 0pt.**