**The situation in a higher altitude:**

* We have a taxing calculator engine that gets some **rules** from outside and applies them into arbitrary vehicles’ transportation logged in the tolling stations.
* The set of rules are based on:
  + The vehicle’s type,
  + The date of transportation,
  + The time of transportation, and
  + The frequency of transportation.
  + Max charge amount.
* The **Rule-Set**s should be used in a factory principal, so that every city can apply its own rules.

For the simplicity, some other special rules are ignored for now, but we provision its place so that they can be easily plugged in.

**Presumptions:**

* The given dates should fall within the 2013, hence, we filter the other dates.
* The rules are sat in a relational data store and will be fetched from there once we create the **rule-set** for each city.
* I assume that my colleague not only won’t get mad I change most of his/her code, but he/she would be glad to learn more (as a junior) how to write a more verbose and reliable code. I understand that if I change his/her code drastically, he/she would be disappointed. Hence, I have checked with him/her and he/she allowed me to do that.

As I’m not quite sure that what is the holidays in the Gothenburg, I assume that the provided dates in the code are the holidays: 1/1, 3/28, 3/29, 4/1, 4/30, 5/1, 5/8, 5/9, 6/5, 6/6, 6/21, 11/1, 12/24, 12/25,12/26 12/31.

**A closer look:**

* Some rules are self-satisfied means that they only need to have the transportation time, whilst some others need its previous transportation information to be able to calculate the tax (e.g., the ones that are based on the frequency transportation).
* Some rules short-circuit the process (e.g., the ones that passes special vehicles with no charge or max charge amount).
* If an interval frequency falls into the next day, we won’t consider it in the calculation for the single charge rule.

**My approach:**

* Whereas I’m not quite dominant on the subject (a.k.a., domain problem), I won’t use TDD approach as it needs me to have a firmed understanding on the domain. Moreover, it is not a green-field app as TDD recommend to start writing tests before you start writing code implementations.
* First step:
  + First off, I’ll simplify the code so I have a better look of what is happening in the already existing solution. I also change some variable names and move the enum into a separate file for a better understanding.
  + Then, I’ll refactor that to fix some bugs regarding the rules and also change the method logic.
* At this stage I’ll add some test to make sure the code is fully functioning (maybe change my code to get the desired result).
* Finally, I’ll change the layout (files and folders) to make it easy to follow.
* For the sake of not doing over engineering I just keep the solution simple, whilst there are some rooms to augment it even more. For instance, we can apply Factory design pattern along with DI so that every city loads its own rule-set dynamically.

**Some bugs of the current solution:**

* A corner-case for the 60min (single charge rule) is that a vehicle may pass more than two times, still we need to pick the highest charge between these tax charges.
* Regarding the previous note, the developer comparing the initial entrance date-time with the rest of the entrance.
* The “TollFreeVehicles” enum does not quite cover all the exempted vehicle types. Writing them in the same order in the “enum” can help to don’t miss any. BTW, “Tractor” should be omitted.
* The maximum charge is 60 SEK per day, whilst the code calculates the total charge all in all and only then compares it against 60 SEK.
* Based on my presumptions for the holiday’s dates, looks like the developer forgot to consider the day-after-and-before-exemption rule.
* As the assignment states, we are “limit the scope to the year 2013”. However, the developer writes a code that handles only one day.