Transportation company has Drivers, Vehicles, Events and Trainings.

The company has 3 types of events: Accidents, Parking Tickets, Traffic Tickets.

Each Event has the driver & vehicle that related to this event, you want to know, when the event occurs and where it took place (city, street).

Accident need to have information regarding the other vehicle and driver involved, Parking tickets the amount of the fine, and for Traffic Tickets the cause (speeding, red light etc…) and the fine.

Each driver has a list of training he finished in the company.

The managers want to calculate the bonus for each driver in a specific time frame of their choice as will be describe later, bonus can be positive or negative, if it is positive the it is actually a bonus, if it is negative it is a fine the driver should pay the company.

When you design your software take under consideration that changes can occur during the development, we may ask about new features and how you can add them into your code.

You don't have to make it interactive (only if you want), what matters here is the way you get the information from the entities and the relationship between them.

I would like to have End-2-End Implementation I am less interested in the frontend aspect, API description is enough (Rest + Json)

Requirements:

* Use relational DB (you may choose any type you want, but try not to use in-memory DB)
* Use Spring projects and Spring Boot like: spring-data (for persistence), spring-mvc (for Restful application), etc.
* Use Hibernate Annotations (requirement is to use annotations), and Hibernate core as the Spring-Data provider (it’s should be the default provider for spring-data)
* Use Tomcat 8 for deployment (should be embedded Tomcat with Spring Boot)

Use Cases:

* Show all Drivers (also add & delete)
* Show all Trainings (also add & delete)
* Show all Vehicles (also add & delete)
* Show all Events of Type X (x is a parameter, also add & delete)
* Show all trainings for a specific driver
* Show events of type X and driver Y (x,y are parameters)
* Add new training for a driver
* Show the Breakdown of events in percentages, meaning the percentage of accidents, parking tickets & traffic tickets.
* Show the bonus for each driver in a time frame (start-end dates are a parameters)
* Calculate the balance of bonuses in a time frame (start-end dates are a parameters)

Additional Requirements:

* Dates are Date Time format
* Names (drivers, trainings etc) should not be longer than 15 characters.
* Description of events should not be longer than 1000 characters.
* Cause of Traffic tickets should be defined enum.
* Calculate bonus:
  + Calculate driver points:
    - Training is 1 point
    - Accident is -3 points
    - Traffic tickets is -2 points
    - Parking tickets is -1 points.
  + Bonus\fine is calculated with the following equation: no. of point \* 100

Bonus (Only if you have finished all the tasks above)

* Logging - please add logs to your system, in addition to your logs add TRACE logs so that on each function in your code I can know when the function is started and when it’s finished.
* Exceptions – when a constraint is broken, please throw exception and handle it as well.
* Testing – there is no such thing as code that is not being tested, as part of your work please test your system (automation testing), it doesn’t matter with which library you will use but you will be asked about the technology and the design of you tests.
* Create a background task that on every first of a month will calculate the bonuses of the last month for each driver and write it to a file (very simple csv file with driver id, name and the bonus), the name of the file is the month of the bonuses.