Frank Ziegler Homework 2 (OOD)

This UML diagram models both the `Model` and `Service` layers of an MVC registration (as well as titling, licensing) system.

It is unfinished.

A customer initiates a transaction, which is composed of customer information, vehicle information, existing registration information, insurance information, etc. The service layer validates this transaction first with a `TransactionService`, which uses a validator factory to determine whether all the rules/needs of the MVC are met, and a `FeeCalculator` to determine what fees to apply.

The Validator Factory Pattern

A `ValidatorFactory` returns a `CompositeValidator` to the `TransactionService`. A `CompositeValidator` maintains a list of `LeafValidators`, which are validators used to handle any and every particular type of `Transaction`.

`CompositeValidator` is implemented by `RegistrationValidator`, `TitlingValidator`, etc. These are the different kinds of "MVC services." So, a particular `RegistrationValidator` will utilize the data behind the transaction (the kind of vehicle it is, the kind of transaction it is, the fuel type, if its commercial, etc.) to choose the correct `LeafValidator`s, which will validate any of their respective cases.

The Fee Calculator

Based upon the information given by the model (the transaction object and the fields inside it like vehicle, customer, etc.), the fee calculator will inspect it and assign fees appropriately.

This part I didn't finish. But, the idea here is that there is a bunch of fees, i.e.

```
static int historicFee = 25;
static int commercialFee = 25;
static int gvwrFee = 25;
```

or whatever.

The General Idea

The model is simple: it is the facts. The Customer. The Vehicle. The License Plate. The Registration.

The service layer can be broken into two components:

1. The validation
    - Are all the necessary documents here? Is the inspection valid? Is the registration valid?

2. The fee calculation

The central hub of it remains in the `TransactionService`, which is an interface that is implemented by the services the MVC has like `RegistrationService` where you can renew, create an initial registration, transfer, etc.