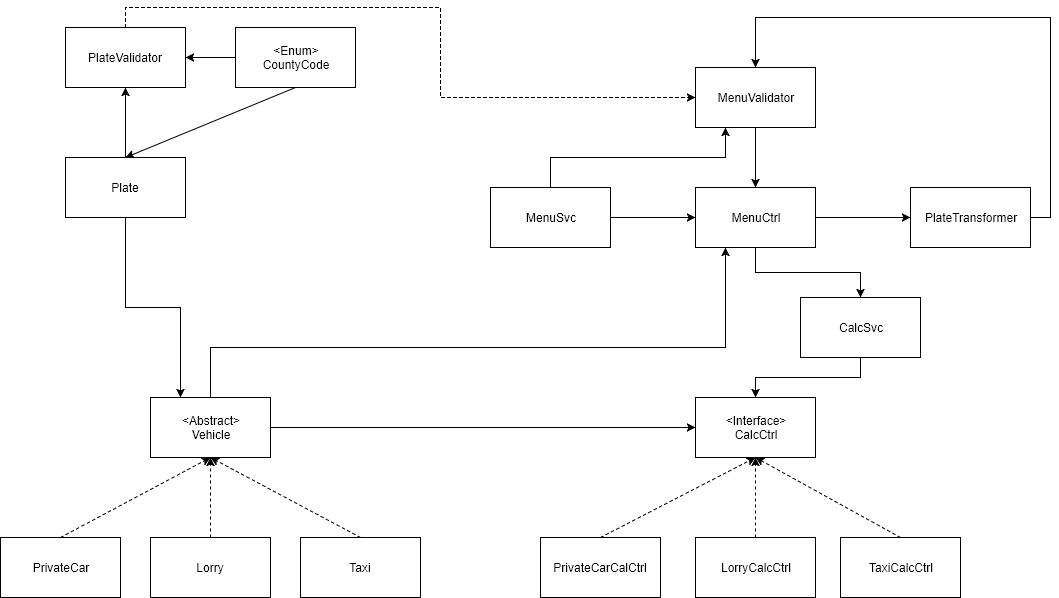
**Structure**

View: MenuSvc

Model: Entity

Controller: calculation services

Others: validator to check the plate number is right format



Flow:

1. User enter vehicle type, plate and engine size in the MenuSvc, data passed to MenuCtrl
   1. Plate string is passed to PlateTransformer and convert into Plate object
      1. MenuValidator extends PlateValidator validates the plate format, if the object meets the plate number format, set validated as true, return true; if not, return false, MenuCtrl will let MenuSvc show error message and ask user to input a valid plate number;
   2. MenuValidator validates the input such as engine size should more than 1000cc or 1L, CO2 emission should be positive number etc. If it doesn’t meets the rule, MenuCtrl will ask user to input again.
2. If the above input data is valid, then MenuCtrl will control MenuSvc and ask user to input further information based on the vehicle type.
   1. All valid input data will be passed to CalcSvc which will then pass it to the relevant calculation controller
3. CalcSvc links menu and calculation controllers, it creates the vehicle objects (private car, taxi, lorry) with validated plate to the relevant calculation controller, and return the calculation result back to menu. CalcSvc acts as an injector.
4. CalcCtrl gets the vehicle objects and other relevant data to calculate the result

**Process:**

Select a type of vehicle

Please enter your registration number

Please enter the engine capacity: select 1 to enter in cc, 2 to enter in liter.

Depending on the vehicle type, different value will be asked to enter:

* (For private cars, if registered in 2008) Was the vehicle registered before July?
* (For private cars) Please enter CO2 emission volume, it should be a positive whole number
* (For lorries) Please enter number of fixed wheels this should be an even number between 4 -12 inclusive
* (For lorries) Please enter the unloaded weight in tonnes with max 2 decimal places
* (For taxi) Please enter the maximum number of passengers

The tax for vehicle <registration plate> will be <amount>

**Objects:**

Entity:

Vehicle: Abstract class, obj plate, double engineCapacity, string type

PrivateCar: extends Vehicle, + int co2Emi, boolean regBeforeJuly08

Lorry: extends Vehicle, + int wheelNum (even number)

Taxi: extends Vehicle, + int passNum (1-10)

Plate: regYear, regHalfYear(overloaded constructor), countyCode, seqNum, boolean validated, override toString(),

CountyCode: enum (<https://en.wikipedia.org/wiki/Local_government_in_the_Republic_of_Ireland>)

Calculation Service:

CalcSvc: obj plate, obj privateCar, obj lorry, obj taxi and all the parameters for these objects

Calculation Controller:

CalcCtrl: interface, double calTax()

PrivateCarCalCtrl: impletments CalcCtrl

LorryCalCtrl: implements CalcCtrl

TaxiCalCtrl: implements CalcCtrl

Validation:

PlateValidator: Plate number is in the right format. Result is a Boolean value. Return true for failure and true for passed, it will be used in a do while loop

MenuValidator: extends PlateValidator, validation result is Boolean which indicates validation success and failure. A do while loop will be used then in the MenuCtrl, so success will return false, failure will return true.

MenuSvc: the text GUI for getting input data and showing calculated result.

PlateTransformer: “ stepIt split the string based on “-“ to

**Core Algorithms:**

PrivateCarCalcCtrl:

* If the registration was before 2008, then it will be calculated based on engine capacity;
* If the registration was between Jan 1st 2008 and June 30th 2008, compare the co2 emission and engine capacity and take the one has less value;
* Else calculate based on CO2 emission.

If engine size is more than 2000cc, result\* = 1.2

LorryCalcCtrl:

0.1\*engineCap + 100\*wheelNum+4\*unloadedWeight

TaxiCalcCtrl:

0.1\*engineCap + 10\*passNum

PlateValidator:

* Sequence number (string) length should be 1 to 6 digits whole number between 1 and 999999 inclusive
* County code is within the county code enum
* Vehicle registered in 2008 and 2013 onwards has Boolean regBeforeJuly not null
* Vehicle registered before 2013 except 2008 has no regBeforeJuly value

**Unit Testing:**

1. PrivateCarCalcSvc: create 3 private cars with registration in 2002, 2005 with 3000cc engine size, first half of 2008, 2012 and check the result
2. LorryCalcSvc: create 4 lorries

Lorry1: Lorry(plate, 8, 2000, 3.0)

Lorry2: Lorry(plate, 4, 15000, 17.5)

Lorry3: Lorry(plate, 12, 10000, 16.0)

1. TaxiCalcSvc: create 5 taxi with passenger number 1, 4, 6, 9 and check the result
2. Plate: create below plate number strings, it should return the expected result. Correct format plate number should pass the testing. Plate testing will be before vehicle testing.
   1. 12D12: MISSING\_HYPHEN
   2. 221-D-12: INVALID\_YEAR
   3. 151-DN-123: INVALID\_COUNTY\_CODE
   4. 121-D-123: WRONG\_FORMAT\_BEFORE\_13
   5. 00-D-1234567: OVER\_BOUNDARY\_SEQUENCE\_NUMBER
   6. 00-D-0001: WRONG\_FORMAT\_SEQUENCE\_NUMBER
   7. 00-D-0: OVER\_BOUNDARY\_SEQUENCE\_NUMBER
   8. 01-D-1: Pass
3. PlateTransformer: Plate should convert String into object
4. CalcSvc:

MenuSvc testing will be the last

**Sequence of Development Tasks:**

Start with Taxi

1. Entities (Plate and then Taxi )
2. Calculation controller
3. MenuSvc + MenuCtrl
4. CalcSvc

Unit testing on the go

Then add Private cars and Lorry.