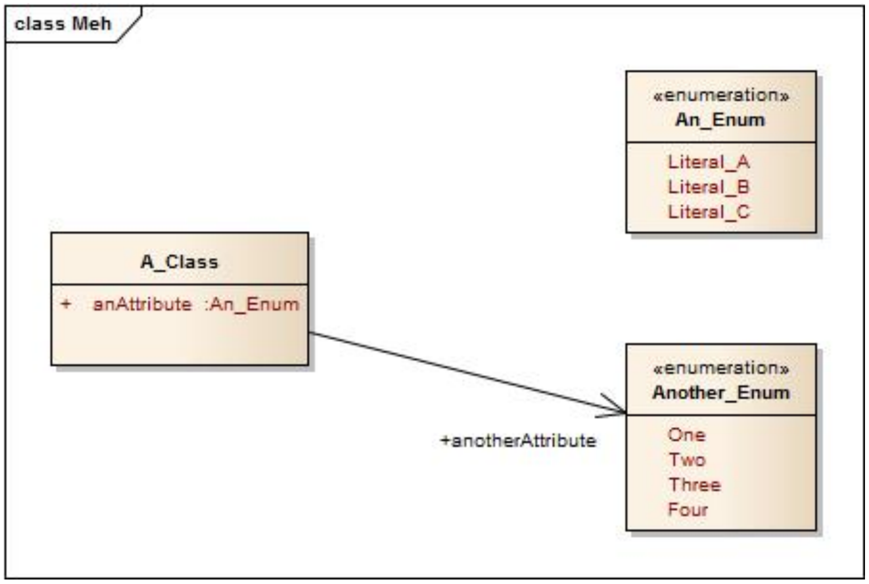
Mobile App

- I think on the Entitlement piece he said it was okay if just use the old Entitlement package and abstract a lot of those detail, kudos to you for still building something out, but I don't think you need to spend too much more time here in your designs.
- I was thinking from a high level architectural perspective you might want to move the entitlement services to the central SmartCar system that way you can manage access from a central point. I don't think you'd have to change much to do it because you can keep all the components maybe just add a call to the central service to check access.
- Might be an opportunity to separate concerns more by adding some classes to handle specific functionality like User Profile, Preferences, Ride Requests, etc.

SmartCar Service

- I think you might want to adjust your use of enums. Enums are great for typing things, such as days of the week, but don't really work for things like time, speed, direction where values aren't a specific set of possibilities. Also, for the enum classes you'll want to provide just a list of the possible values for that enum, not so much fields. I think your use of VehicleType is spot on for enums, but your LocationType is actually a class, which is an easy change. Here is a good example of an enum UML:



- For vehicles I like the approach of using enums instead of immediately jumping to classes for purposes of reuse, but subclassing can help when you want different functionality based on the type. Vehicles will probably have a lot of different behavior and properties about them that might make the enum approach cumbersome to maintain. E.g. how routes are calculated, different features of the vehicle, etc. I used the strategy pattern heavily here just to have the flexibility for different types of vehicles to easily have different behavior. You could get away with using enums, but I think you'll end up with lots of case statements.

Vehicle System

- Like a command approach here
- I'm not sure if you need the CommandImplementation abstract class. This looks like an extra layer added to the command pattern, versus just implementing the command interface directly, but maybe you had some idea in mind as to how you wanted to use it, such as you want all commands to have some similar piece of functionality, even then it might be worth looking at composition.

command interface directly, but maybe you had some idea in mind as to how you wanted to use it, such as you want all commands to have some similar piece of functionality, even then it might be worth looking at composition.

- Similar to Mobile App, it might be easier on you if you separate concerns more, it'll make it easier to manage.