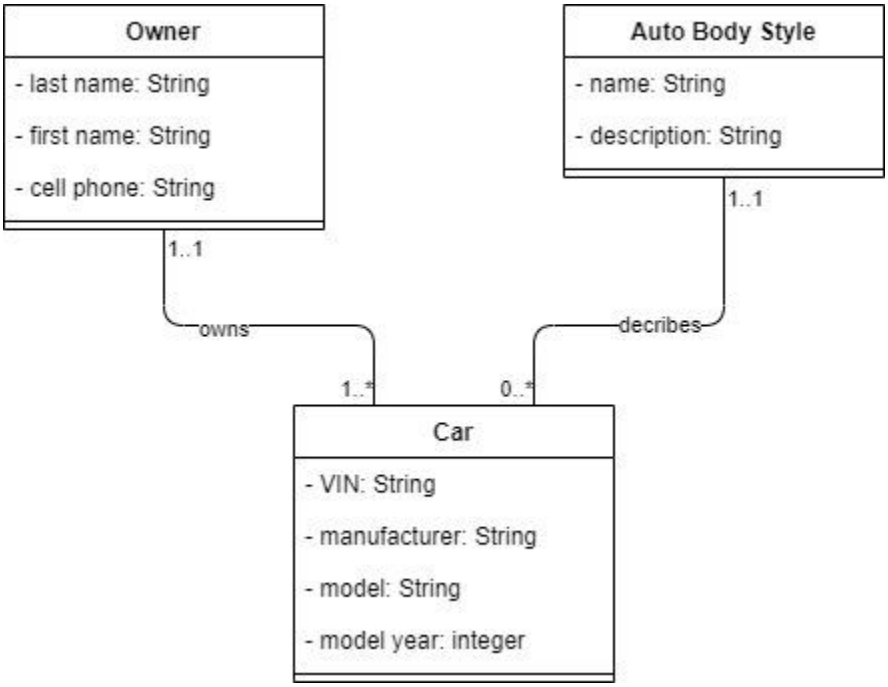
CECS 323 LAB JPA

OBJECTIVE:

Gain some first-hand experience using JPA and test out the technology stack along the way.

INTRODUCTION:

Your neighborhood car club wants to set up a database to track information about owners and their vehicles. Of course, one owner can own many vehicles, but a given vehicle can only have one owner (at least for our purposes). Each vehicle has a particular body style that we associate with it. The UML for this little database looks like this:



One could easily make the case that there are only so many manufacturers of automobiles, so we could have made a lookup table for manufacturer. One could also argue that there are only so many possible models as well. In fact, we **could** have made a Model class that is a child to the Manufacturer class. But this is meant to be a simple intro.

You will find the **annotated** code for Owner and Auto Body Style here. I also supply you with the **POJO** for the Cars class. It will be your responsibility to properly annotate the Cars class. The project has a seed.sql in it which will populate the auto_body_styles table when you start the application so that you do not have to worry about that. The persistence.xml references the seed.sql and will cause JPA to run seed.sql prior to running your application. You also have a skeleton main program that you will have to update to test persisting the Cars entity.

Note that the Vehicle Identification Number (VIN) of Car uniquely identifies all cars.

PROCEDURE:

1. If you have not already done so, please watch the lab introduction video here.

2. Start with the Cars POJO supplied and annotate it.
    a. Let JPA know that VIN is the primary key for Cars.
    b. You will have a many to one relationship to Owners.
        i. Be sure to use the join column annotation to both make sure that the migrated foreign key is never null. See here for how to use the nullable=false parameter in the join columns annotation to make sure that we never create a Cars instance without an owner.
        ii. "Role name" the owner_id in Owners to owner_id in Cars.
    c. You also need to annotate the many to one relationship to auto_body_styles.
        i. The primary key of auto_body_styles is an attribute called "name". You will want to role name that to "auto_body_style_name" in Cars.
        ii. See here for how to use the join column annotation to give a role name to a migrating foreign key.
    d. Be sure that each attribute (that is not part of an association) is specified NOT NULL and given a maximum length using the @Column annotation. I have a comment in the Cars code, telling you the maximum length for each attribute.
    e. Write a public constructor that takes in all the attributes of the Cars class and a public default constructor that takes no arguments and does nothing.
3. Update the CarClub source in the app folder under src to create several owners, and give each of those owners at least two cars, and persist the new owners and their cars.
4. Once you have the code up & running get a screen shot from the Derby database showing a list of the rows in the Cars and Owners tables.
5. We do not need to change the auto_body_stypes entity at all. Leave that alone.

WHAT TO TURN IN:

- Your annotated Cars class java file.
- The updated Owners class java file reflecting the one to many relationship that you added.
- The updated CarClub.java with the changes in main() to create and persist Owners and Cars instances.
- The screen shots of the Owners and Cars tables showing the records that you have successfully added by running your code.
- Your team's filled out collaboration document. You can find a template for that here.