Creating Class Diagrams in GenMyModel

CSC 2310

In this lab you will continue to practice creating diagrams using the GenMyModel environment. You will use the information contained in the **Problem Description** shown below as context for creating the relevant diagrams(s).

Pre-work

- Download the lab source files using the following command:

```
% git clone https://gitlab.csc.tntech.edu/csc2310-fa22-
students/%userid%/%userid%-lab-03-uml.git
```

replacing `%userid%` with your own TNTech issued userid.

Problem Description

Concept

We are creating a carpooling application for the students and faculty of Tennessee Tech that enables them to get to and from campus in a timely manner without having to worry about searching for a parking spot. It will allow students or faculty to travel to and from campus by hailing a ride and informing other users of their desired destination. Another feature will allow the driver to post information about where they are planning to travel so that other users can request to join them. After a ride has been completed, users will have the ability to rate their rider or driver to ensure that the user experience remains safe and enjoyable. Unlike other applications for similar services, the drivers will not be paid directly but rather through other forms of compensation. This service will only be available to students and faculty of Tennessee Tech so that it can foster safety and community of its users.

Elevator Statement

For students and faculty at Tennessee Tech who seek a solution to the parking problem on campus, the carpooling app is a tool that will make it easier to get to and around campus while also providing incentives to those who use it. Unlike other ride sharing apps, our solution will be designed to provide a service that is only available to students and faculty of participating universities so that it can ensure the safe delivery of users to their destination while also not requiring a direct transfer of money.

Objects, Classes, and Relationships

The two primary types of users for this application are *Drivers* and *Riders*. As such, each has their own variant of the system that can be built from the same core system. However, there is enough variation in the main components of the system that the components can be differentiated using inheritance.

The core app includes the following components:

- Dispatcher - This component connects drivers and riders together; when a rider starts searching for rides, the component connects with a global dispatcher, allowing them to see drivers in the area. Once a driver accepts a fare, the Dispatch object is used by the communication object to support messaging between driver and rider.
- Communication - Manages communication between drivers and riders; drivers can be in communication with many riders (past, present, and future). Riders can communicate with only one driver at a time;
- Map - Displays the map for the area where the drivers or riders are located
- Record - Keeps track of the past rides for drivers and riders; drivers see these records as "fares", including the amount of incentive earned; riders merely have a record of the date and time of the ride. Both driver and rider may add a comment and rating about the driver/rider. The record is associated to a profile (described below).
- Profile - Drivers and riders each have profiles that include an id, name, and reputation score. Drivers also have a picture of their vehicle, it's license plate number, and other attributes that support phyisically identifying their car. The profile is directly associated to the Record object described above.

Note that drivers and riders use **different** apps, while they are built off of the same core system, they are different apps. Your model must depict this separation through the use of inheritance, where appropriate.

Activity

In this assignment, you will create a class diagram based on the description given above. See the following resources for information on class diagrams

- iLearn > Content > Slides
- Teams > Lecture Models and Examples > Posts
- Chapter 1, Python Object-Oriented Programming (Lott, Phillips)

In the model, it is expected that you will use the following modeling constructs:

- Class, including attributes. Operations are optional
- Relationships
    - Inheritance/Generalization
    - Aggregation (collections of classes)
    - Composition (part/whole relations)
    - Association (i.e., less-restrictive relationships)

In your model you should hide "End Names" on non-inheritance relationships, but include multiplicities.

Note on exporting your model: You must change the default color of the **composition** relationship to black in order for the diamond end to appear as black. You can do this by selecting the relationship, choosing the "Properties" tab (lower left hand corner of the screen), and setting Appearance > Color to black.

Turn-in

You must include the following meta-data in your diagram as a text annotation (Right-click->Comment)

- Your name

- T-Number
- Lab Section

Export your model to a `.png` file with the following name as follows: `userid_lab03.png` in the directory created when you executed the `git clone` command above. For example, `jgannod_lab03.png`.

Submit your `.png` file to iLearn using the appropriate link and by using the following commands:

```
% git add userid_lab03.png
% git commit –m "Completed assignment"
% git push –u origin master
```

This laboratory is worth 20 points.

Rubric

- Completeness (7 pts): The components and relationships identified in the description are implemented in the model
- Correctness (11 pts): UML Class Diagram notation is used correctly, including the correct relation types and multiplicities
- Submission (2 pts): File submitted using the specific standard (filename correct). Git submission is expected but will not be assigned a point value in this lab.