Project – Car Rental System

Based on the UML diagram in Figure 1, build a car rental system that allows users to add customers, rent and return cars.

Requirements

1. (Weight: 20%) **Business Logic:** Build classes that correspond to the UML diagram in Figure 1. Of course you will need to add methods to some of the classes, for instance, methods that allow a customer to be searched. The `CarSpec` class represents the specifications of a car. Many cars could have the same specs. The size of a car could be small, midsize, or large. The `Car` class represents an actual physical car that can be rented. The car has an ID. A customer can rent cars. When a car gets rented, the rental information is stored in `Rental` objects. The status of the rental can be either rented or returned. When the customer returns the car, the status becomes returned, and the returnDate is recorded.

2. **Graphical User Interface:** Build the following windows (`JFrames`):

   2.1. (Weight: 20%) Build a window allows users to search for customers (Figure 2). The user can type in anything about in anything about the customer (e.g. phone, last name). When the user clicks "Search", the system will find any customer records that match what's in the text field, and display them in a grid. When the user selects a customer (in the grid) and clicks "Rent Car", the interface in Figure 3 will show up (tab "Find Car" will be displayed). However, if the user clicks "Rented Cars", the system will bring up the interface in Figure 3, but tab "Rented Cars" will be displayed. The window in Figure 3, 4, or 5 can do things (e.g. rent a car) only for the customer that has been selected in the window in Figure 2.

   2.2. (Weight: 20%) **Find Car** – This panel allows the user to search for a car to be rented (Figure 3). As a result of searching, the system displays information about the cars available to be rented. The system shows information about available cars from the `CarSpec` class. The user can select multiple cars to rent them. Clicking "Rent car" will bring up a dialog box that allows the user to type in the rental date. For each rental, a corresponding `Rental` object will be created.

   2.3. (Weight: 20%) **Rented Cars** – This panel displays the car that the customer has rented (Figure 4). This panel gets updated whenever the user has rented a car. The user can return the cars by simply selecting them and clicking return. When the user clicks "Return Car", a message box will show up asking the user to enter the return date. Once the user confirms the return, the system will update the Rental object (in particular, the status and return Date).

2.4. (Weight: 10%) **Returned Cars** – This panel displays the car that the customer has returned (Figure 5). This panel gets updated whenever the user returns a new car.

3. (Weight: 10%) **Deployment:** Make a class "CarRental" that runs the application. Add a few customers, car specifications, and cars manually.
4. (Optional – Weight: +10% within the scope of the project) **Database:** It is okay to develop the application based on temporary data within the system itself. However, if you develop a persistent database for the system, and write queries that interact with the database to retrieve and modify data, you get 10% extra credit within the scope of the project.

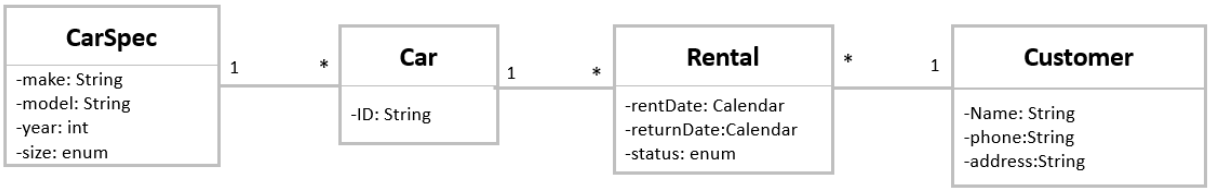

Figure 1

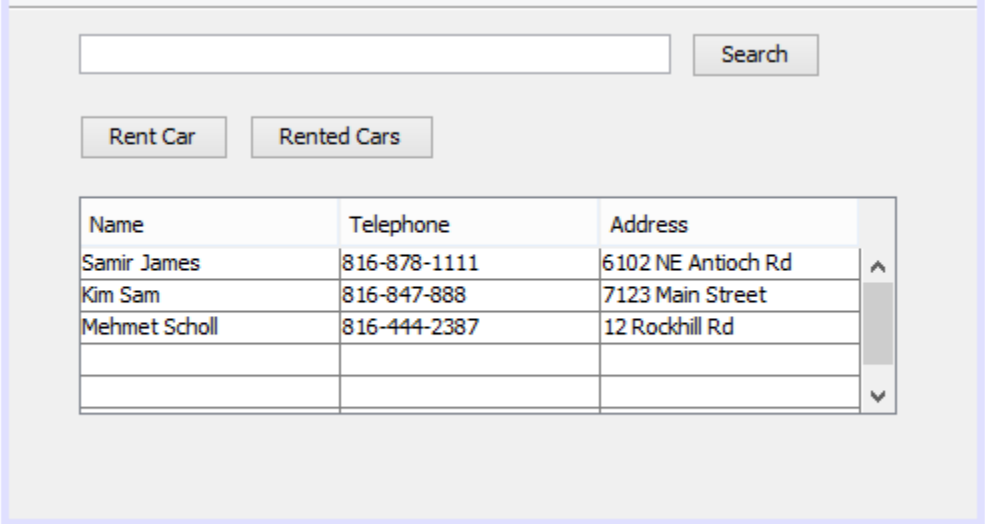

Figure 2: Customers frame

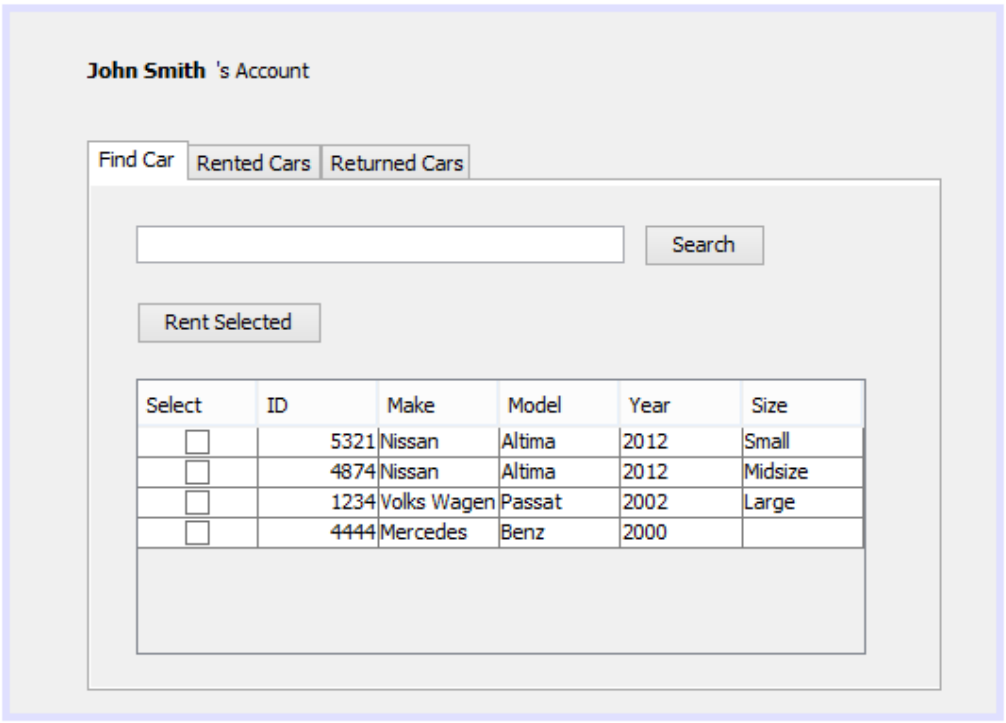

Figure 3: Customer's frame – Find Car tab

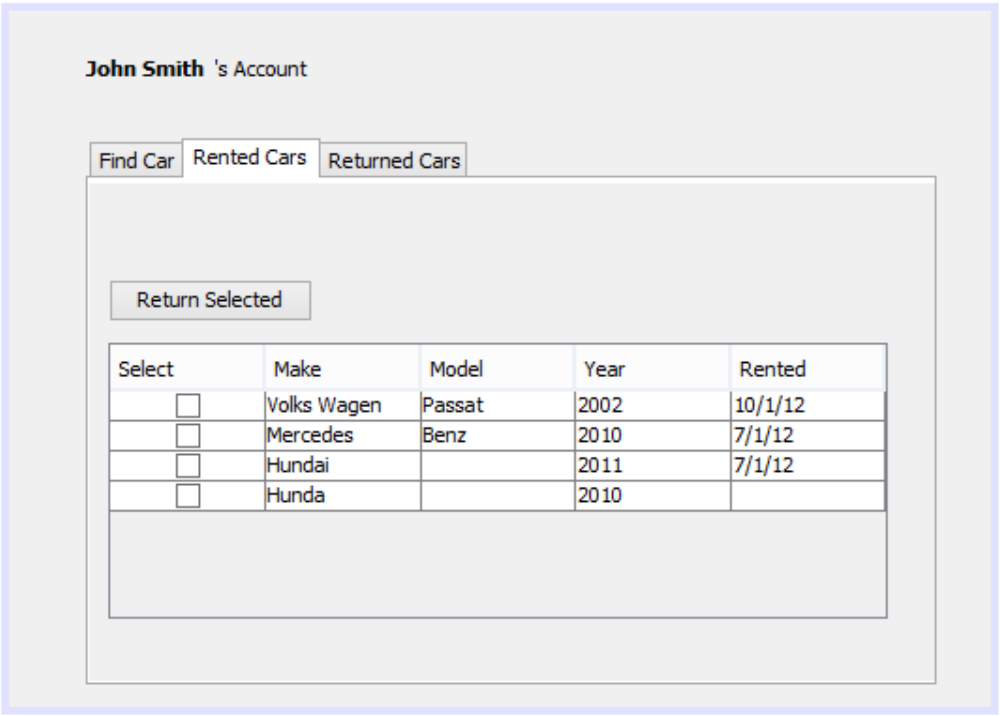

Figure 4: Customer's frame – Rented Cars tab

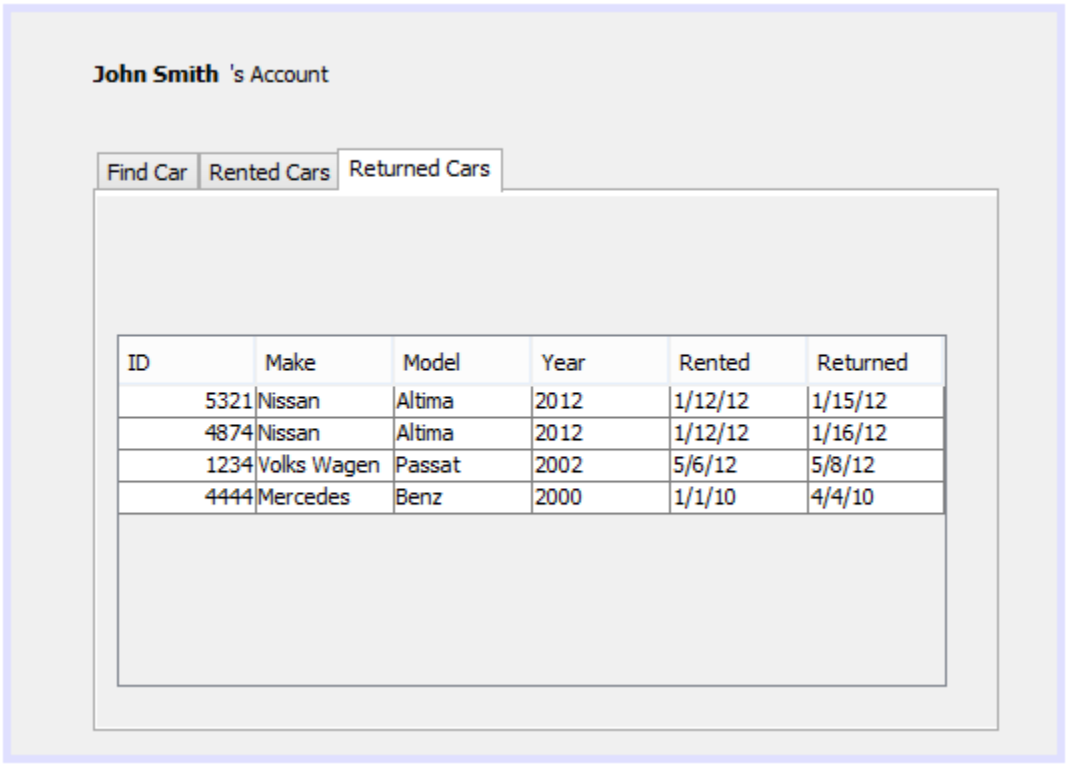

Figure 5: Customer's frame – Returned Cars tab