

P1 Proposal: Car Organizer

Brief Project Description:

This app enables users to quickly search through and list the used car inventory of a dealership by each car's mileage. The app makes use of a Red-Black Tree to efficiently insert the car data and retrieve it in a sorted manner. The data is loaded from a local data file that the user specifies.

Representative Tasks Performed Using this Application:

1. Search for and list the cars with the lowest mileage in the inventory.
 2. Search for and list all cars that have 900,000 or more miles.
-

Backend Developer (BD) Role

The backend developer writes code that reads in car inventory data from a CSV file, inserts it into a Red-Black Tree, and accesses it based on commands from the frontend. To read in the data, the backend developer develops a java class that defines a single car object and that can be inserted into the Red-Black Tree using each car's mileage as the key. The backend supports data sets formatted in the same way as this [car dataset from kaggle](#) in CSV format. The frontend accesses the backend's functionality through method calls on the backend object.

Interface Design Responsibilities:

- An interface for a class that defines a single car and exposes car properties required by the frontend: brand, model, year, price, and mileage.
- An interface for the backend that exposes the required functionality to the frontend: read data from a file, get a list of cars with the minimum mileage, get a list of cars with mileage at or above a specified threshold.

Presentation Responsibilities:

After integration, the backend developer demonstrates a search with the app to find the cars with the lowest mileages in the data set.

Frontend Developer (FD) Role

The frontend developer writes code that drives an interactive loop of prompting the user to select a command, then requests any required details about that command from the user, and then displays the results of the command. The commands to include are: a command to specify (and load) a data file, a command to list the vehicles with the lowest mileage in the data set, a command to list all vehicles with a mileage at or above a specified threshold, and a command to exit the app. The results are computed with the help of the BD's code. When the user enters invalid input, instructive feedback about what they should enter is displayed.

Interface Design Responsibilities:

- An interface for a class that implements the functionality of the frontend for the app. The class has a constructor that accepts a reference to the backend and a `java.util.Scanner` instance to read user input. It also has a method that starts the main command loop for the user. To allow for easier testing of the frontend, the command loop is broken down into a separate method for the main menu and each of the sub menus.

Presentation Responsibilities:

After integration, the frontend developer will record a video demonstration of the task: Search for and list all cars that have 900,000 or more miles.