```
package model;

import data.Database;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * This class functions as the Car model of the application. Car can be
    constructed two-fold:
 * <br>
 * <br>
 * <ul>
 * <li><strong>1: Construct the Car with all variables as parameters.</
    strong>
 * <br> This is needed for the initialization of Cars read from the database
    .</li>
 * <li><strong>2: Construct a Car with limited parameters.<strong>
 * <br> This is needed for the construction of new Cars within the
    application.</li>
 * </ul>
 */
public class Car {
    private int id, seats, owner;
    private String name, transmission, description, brand, type, location;
    private double rate;
    private static int counter;

    /**
     * Constructor with all variables available as parameters.
     *
     * @param newId ID of the car
     * @param newOwner User ID of the car's owner
     * @param newName Name of the car
     * @param newType Type of the car
     * @param newBrand Brand of the car
     * @param newTransmission Transmission of the car
     * @param newSeats Number of seats of the car
     * @param newRate Rate of the car
     * @param newDescription Description of the car
     * @param newLocation Location (street name) of the car
     */
    public Car(int newId, int newOwner, String newName, String newType,
        String newBrand, String newTransmission, int newSeats, double
        newRate, String newDescription, String newLocation) {
        id              = newId;
        owner           = newOwner;
        name            = newName;
        type            = newType;
        brand           = newBrand;
        transmission    = newTransmission;
        seats           = newSeats;
        rate            = newRate;
        description     = newDescription;
        location        = newLocation;
        counter++;
    }
```

```
    /**
     * Constructor with limited variables available as parameters.
     *
     * @param owner2 User ID of the car's owner
     * @param name2 Name of the car
     * @param type2 Type of the car
     * @param brand2 Brand of the car
     * @param transmission2 Transmission of the car
     * @param seats2 Number of seats of the car
     * @param rate2 Rate of the car
     * @param description2 Description of the car
     * @param location2 Location (street name) of the car
     */
    public Car(int owner2, String name2, String type2, String brand2, String
        transmission2, int seats2, double rate2, String description2, String
        location2) {
        id              = ++counter;
        owner           = owner2;
        name            = name2;
        type            = type2;
        brand           = brand2;
        transmission    = transmission2;
        seats           = seats2;
        rate            = rate2;
        description     = description2;
        location        = location2;
    }

    /**
     * Writes a query to add a Car to the database
     */
    public void toDB() {
        Database.write("cars",
            this.id+";"+this.owner+";"+this.name+";"+this.type+";"+this.
            brand+";"+this.transmission+";"+
                this.seats+";"+this.rate+";"+this.description+";"+this.
                    location+" ");
    }

    // Start getter methods
    public int getId() {
        return this.id;
    }

    public int getOwner() {
        return this.owner;
    }

    public String getName() {
        return this.name;
    }

    public String getType() {
        return this.type;
    }

    public int getSeats() {
        return this.seats;
    }
```

```
    public String getDescription() {
        return this.type;
    }

    public String getLocation() {
        return this.location;
    }

    public double getRate() {
        return this.rate;
    }

    public String getBrand() {
        return this.brand;
    }

    public String getTransmission() {
        return this.transmission;
    }
    // End getter methods

    /**
     * Delete this car from the database
     */
    public boolean delete() {
        return Database.deleteById("cars", this.id);
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-2s | %-10s | %-11s | %-20s | %-12s |
            %-15s | %-13s | %-12s | %-30s |\n",
                "ID", "Name", "Type", "Brand", "Transmission", "Seats",
                    "Rate per hour", "Description", "Location");
    }

    public static String printLine(Car car) {
        return String.format("\t  | %-2s | %-10s | %-11s | %-20s | %-12s |
            %-15s | %-13s | %-12s | %-30s |\n",
                car.getId(), car.getName(), car.getType(), car.getBrand(),
                    car.getTransmission(), car.getSeats(), car.getRate(),
                    car.getDescription(), car.getLocation());
    }

    /**
     * Static method to print multiple cars
     *
     * @param headline Headline the table has
     * @param cars List of cars that will be printed
     */
    public static String toString(String headline, List<Car> cars) {
        String out = String.format("\n\n\t"+headline+":\n");
        out += printTableHeader();
        for ( Car car : cars ) {
            out += printLine(car);
        }
        return out;
    }

    /**
```

```
     * Static method to print a single car
     * @param car
     */
    public static String toString(Car car) {
        String out = String.format("\n\n\tCAR:\n");
        out += printTableHeader();
        out += printLine(car);
        return out;
    }

    /**
     * Print bookings when parameter is a list of cars
     * Note: static
     *
     * @param cars List of cars which will be printed
     */
    public static String toString(List<Car> cars) {
        String out = String.format("\n\n\tCARS:\n");
        out += printTableHeader();
        for ( Car car : cars ) {
            out += printLine(car);
        }
        return out;
    }

    public String toString() {
        String out = String.format("\n\n\tCAR:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }
    // End print methods

    /**
     * Static method to filter cars.
     *
     * @param cars The list of cars which will be filtered
     * @param key Key (type, brand, transmission, seats, rate, or owner) by
         which to filter
     * @param value Value of that key by which to filter
     * @return The filtered List
     */
    public static List<Car> filter(List<Car> cars, String key, String value)
        {
        List<Car> filtered = new ArrayList<Car>();

        for ( Car car: cars ) {
            boolean check = false;
            if ( key == "type" ) {
                check = car.getType().equals(value);
            } else if ( key == "brand" ) {
                check = car.getBrand().equals(value);
            } else if ( key == "transmission" ) {
                check = car.getTransmission().equals(value);
            } else if ( key == "seats" ) {
                check = car.getSeats() == Integer.parseInt(value);
            } else if ( key == "rate" ) {
                check = car.getRate() <= Double.parseDouble(value);
            } else if ( key == "owner" ) {
                check = car.getOwner() == Integer.parseInt(value);
```

```
        }

        if ( value == null ) {
            // Add all cars if search value equals null
            filtered.add(car);
        } else if ( check ) {
            // Add car to filtered list if above applies (check == true)
            filtered.add(car);
        }
    }

    return filtered;
}

/**
 * Get a car by its ID.
 *
 * @param searchId ID of the car which is to be found
 * @return The found car (or null if 404)
 */
// to db
public static Car getById(int searchId) {
    Car found = null;
    for ( Car car : Database.getCars() ) {
        if ( car.getId() == searchId ) {
            found = car;
        }
    }
    return found;
}

/**
 * Make a report of all cars. Make two lists, one of booked cars the
     other
 * one of not-booked cars. Print both lists separately.
 */
public static String report() {
    List<Car> bookedCars    = new ArrayList<Car>();
    List<Car> notBookedCars     = new ArrayList<Car>();

    for ( Car car : Database.getCars() ) {
        for ( Booking booking : Database.getBookings() ) {
            if ( car.getId() == booking.getCarId() && !bookedCars.
                contains(car) ) {
                bookedCars.add(car);
            }
        }
    }

    for ( Car car : Database.getCars() ) {
        if ( !bookedCars.contains(car) && !notBookedCars.contains(car) )
            {
            notBookedCars.add(car);
        }
    }

    String out = Car.toString("BOOKED CARS", bookedCars);
    out += Car.toString("NOT-BOOKED CARS", notBookedCars);

    return out;
```

```
        }
    }
```