One Night Car

(vorher Auto4u) Carsharing Verwaltungssystem

Team

Benito Grauel

Pascal Giese

Ahmad Abo Louha

Alejandro Restrepo Klinge

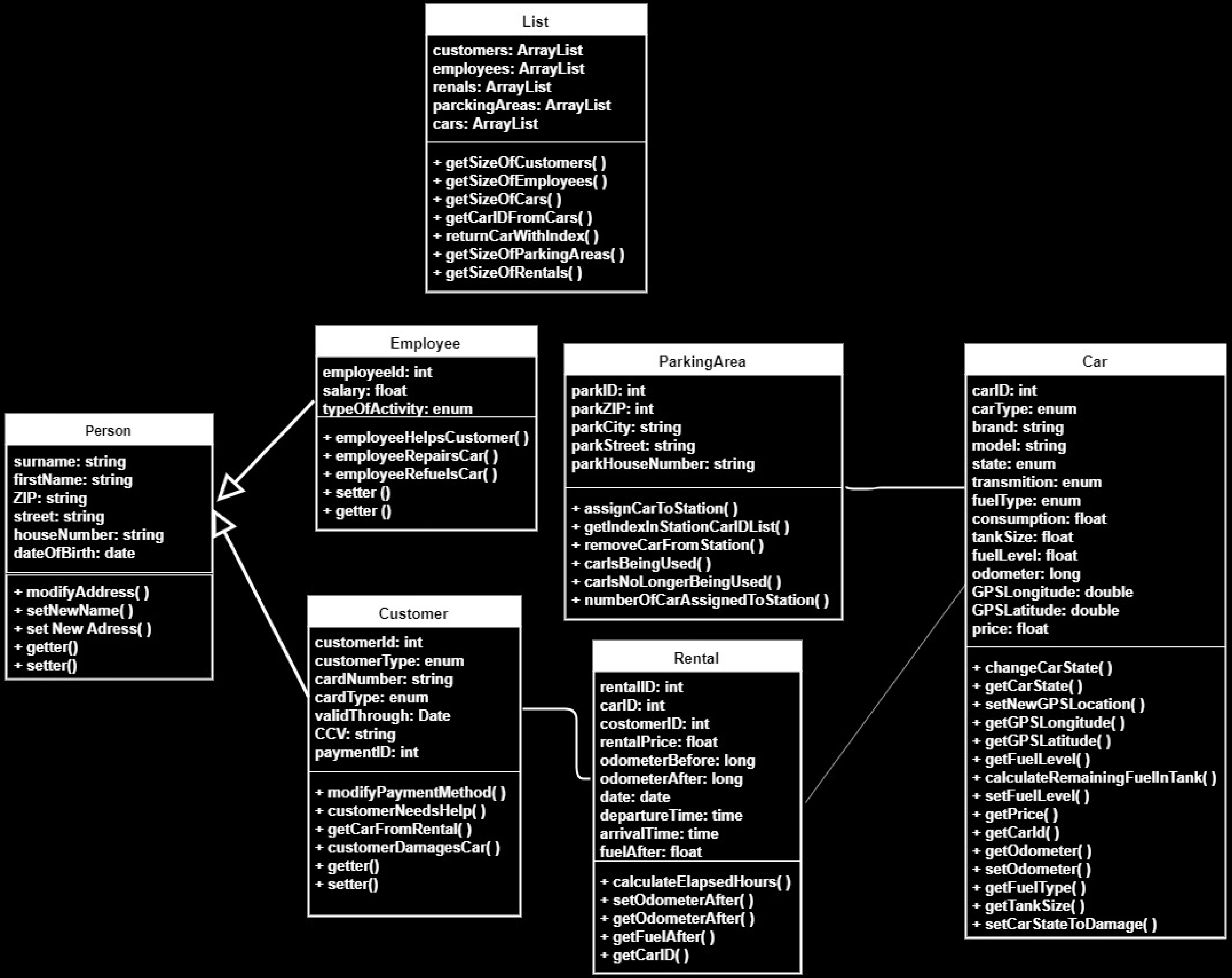
GitHub Repository: https://github.com/fh-erfurt/One-Night-Car

GitHub Team: https://github.com/orgs/fh-erfurt/teams/team-car-sharing

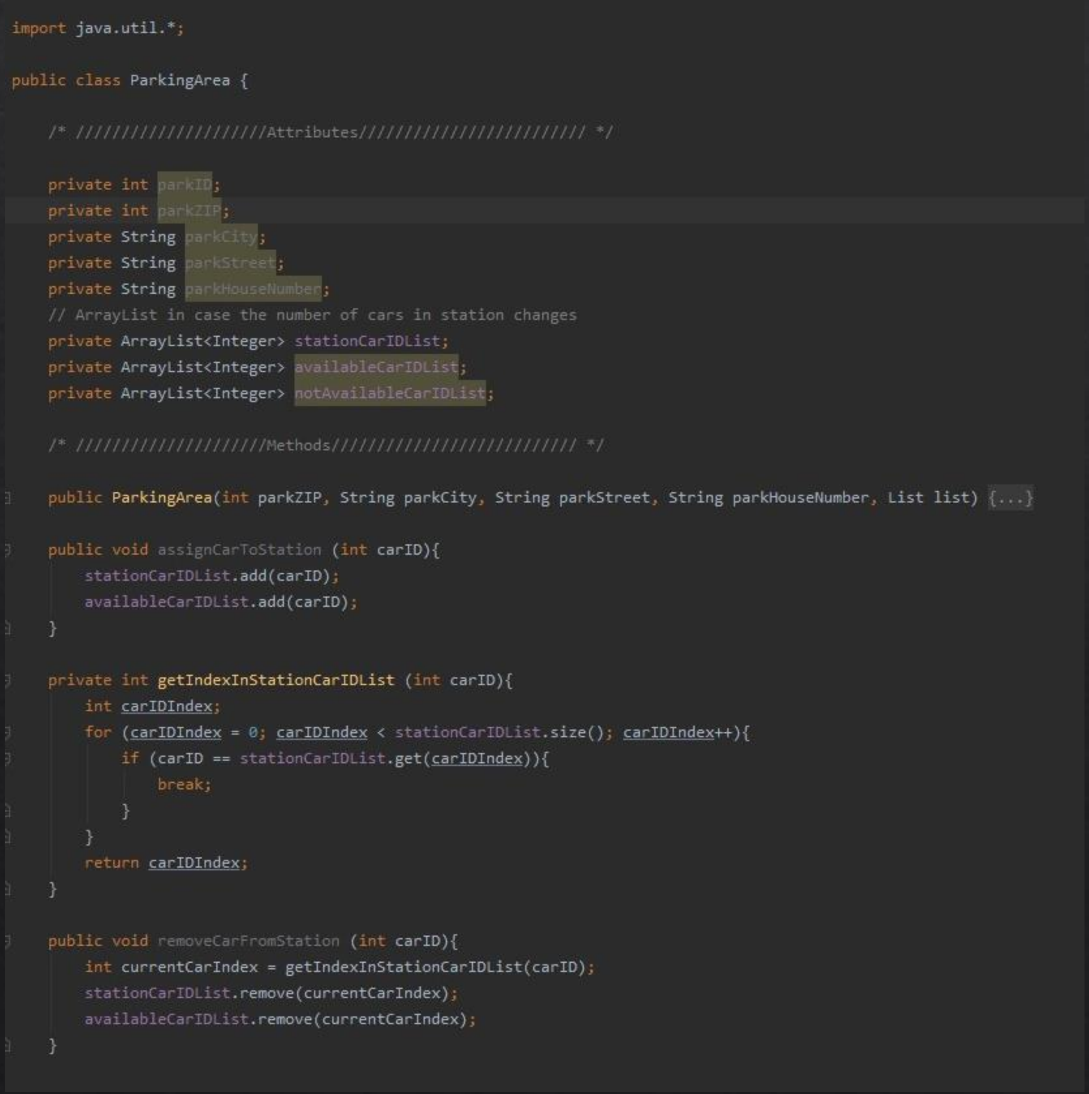
Komponenten

- Stationen: Verwaltung von dazugehörigen Autos (Verfügbarkeit)
- Autos: Verwaltung von Autos (Auto Informationen, Benzin, …)
- Users: Verwaltung von Kunden (Registrieren, Login, Type, …)
- Reservierung: Reservierung von Autos verwalten (Abrechnung, Datum)

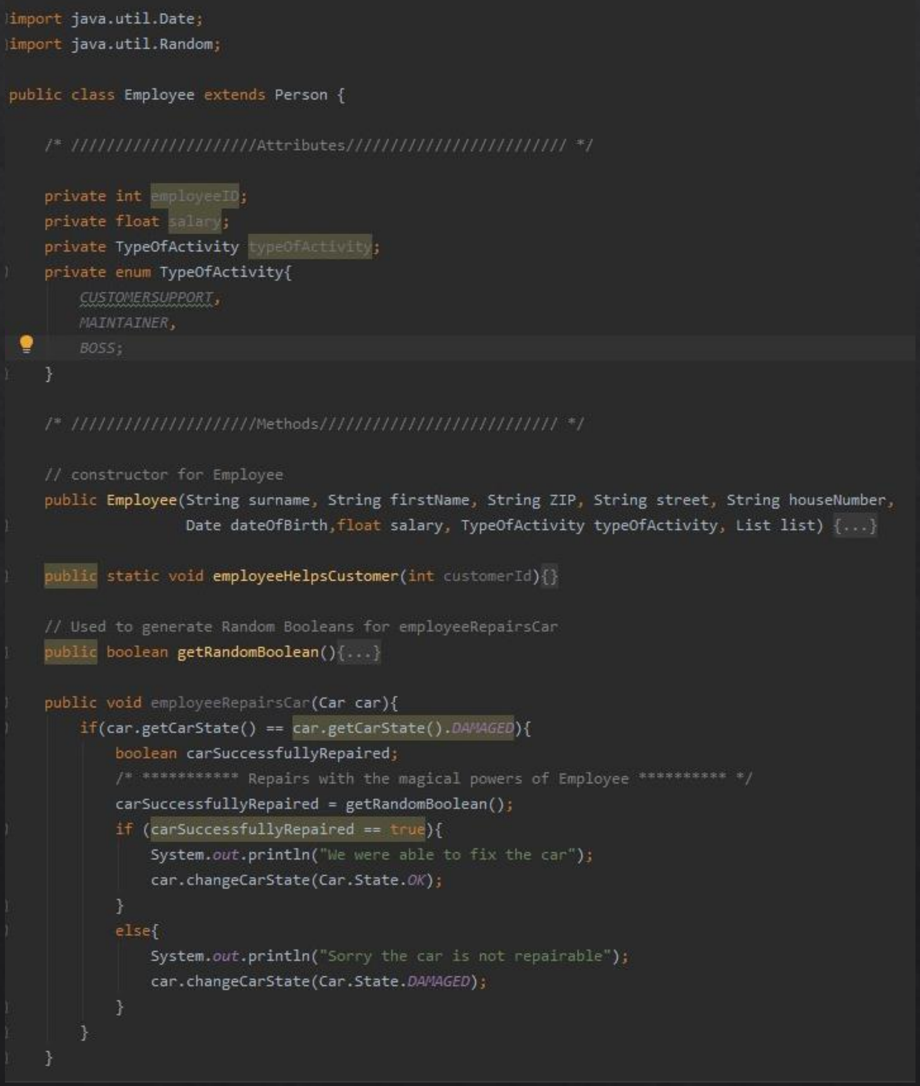
Was haben wir bis jetzt gemacht?


List
customers: ArrayList
employees: ArrayList
renals: ArrayList
parckingAreas: ArrayList
cars: ArrayList
+ getSizeOfCustomers( )
+ getSizeOfEmployees( )
+ getSizeOfCars( )
+ getCarIDFromCars( )
+ returnCarWithIndex( )
+ getSizeOfParkingAreas( )
+ getSizeOfRentals( )
Employee
employeeId: int
salary: float
typeOfActivity: enum
+ employeeHelpsCustomer( )
+ employeeRepairsCar( )
+ employeeRefuelsCar( )
+ setter ()
+ getter ()
ParkingArea
parkID: int
parkZIP: int
parkCity: string
parkStreet: string
parkHouseNumber: string
+ assignCarToStation( )
+ getIndexInStationCarIDList( )
+ removeCarFromStation( )
+ carIsBeingUsed( )
+ carIsNoLongerBeingUsed( )
+ numberOfCarAssignedToStation( )
Car
carID: int
carType: enum
brand: string
model: string
state: enum
transmition: enum
fuelType: enum
consumption: float
tankSize: float
fuelLevel: float
odometer: long
GPSLongitude: double
GPSLatitude: double
price: float
+ changeCarState( )
+ getCarState( )
+ setNewGPSLocation( )
+ getGPSLongitude( )
+ getGPSLatitude( )
+ getFuelLevel( )
+ calculateRemainingFuelInTank( )
+ setFuelLevel( )
+ getPrice( )
+ getCarId( )
+ getOdometer( )
+ setOdometer( )
+ getFuelType( )
+ getTankSize( )
+ setCarStateToDamage( )
Person
surname: string
firstName: string
ZIP: string
street: string
houseNumber: string
dateOfBirth: date
+ modifyAddress( )
+ setNewName( )
+ set New Adress( )
+ getter()
+ setter()
Customer
customerId: int
customerType: enum
cardNumber: string
cardType: enum
validThrough: Date
CCV: string
paymentID: int
+ modifyPaymentMethod( )
+ customerNeedsHelp( )
+ getCarFromRental( )
+ customerDamagesCar( )
+ getter()
+ setter()
Rental
rentalID: int
carID: int
costomerID: int
rentalPrice: float
odometerBefore: long
odometerAfter: long
date: date
departureTime: time
arrivalTime: time
fuelAfter: float
+ calculateElapsedHours( )
+ setOdometerAfter( )
+ getOdometerAfter( )
+ getFuelAfter( )
+ getCarID( )

Klassen in Java definiert und einige Methode dazu implementiert

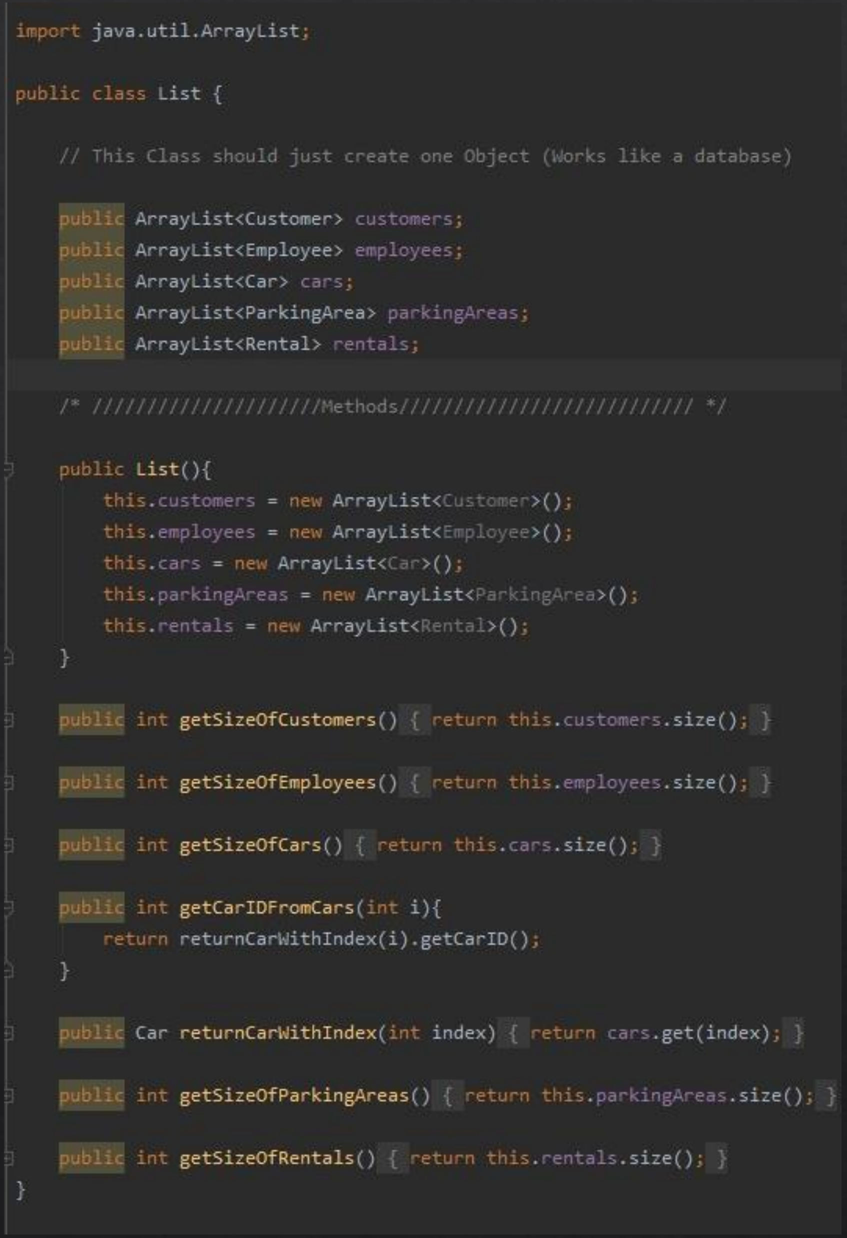


```java
import java.util.*;

public class ParkingArea {

    /* ////////////////////Attributes//////////////////////// */

    private int parkID;
    private int parkZIP;
    private String parkCity;
    private String parkStreet;
    private String parkHouseNumber;
    // ArrayList in case the number of cars in station changes
    private ArrayList<Integer> stationCarIDList;
    private ArrayList<Integer> availableCarIDList;
    private ArrayList<Integer> notAvailableCarIDList;

    /* ////////////////////Methods////////////////////////// */

    public ParkingArea(int parkZIP, String parkCity, String parkStreet, String parkHouseNumber, List list) {...}

    public void assignCarToStation (int carID){
        stationCarIDList.add(carID);
        availableCarIDList.add(carID);
    }

    private int getIndexInStationCarIDList (int carID){
        int carIDIndex;
        for (carIDIndex = 0; carIDIndex < stationCarIDList.size(); carIDIndex++){
            if (carID == stationCarIDList.get(carIDIndex)){
                break;
            }
        }
        return carIDIndex;
    }

    public void removeCarFromStation (int carID){
        int currentCarIndex = getIndexInStationCarIDList(carID);
        stationCarIDList.remove(currentCarIndex);
        availableCarIDList.remove(currentCarIndex);
    }
```



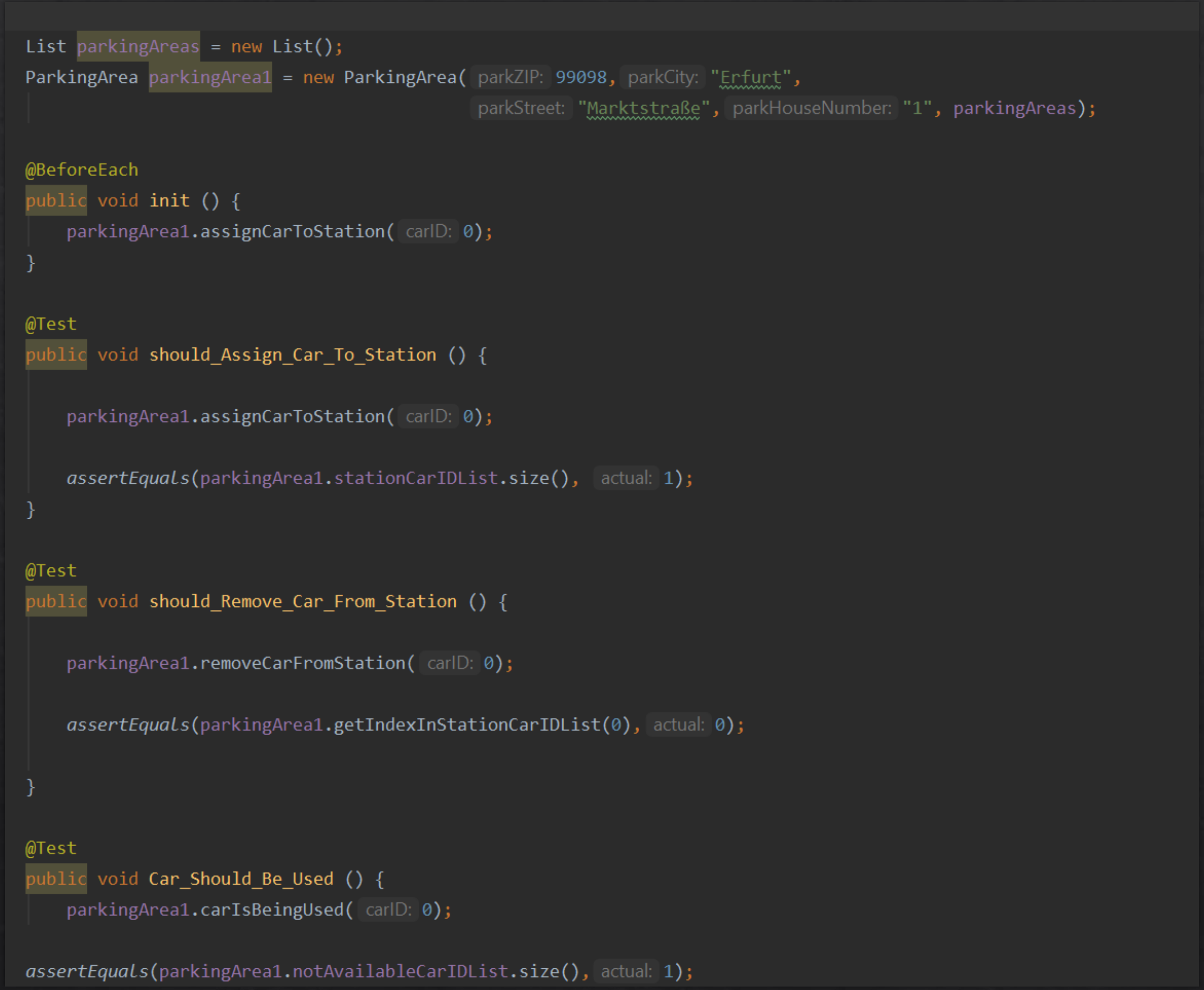
```java
import java.util.Date;
import java.util.Random;

public class Employee extends Person {

    /* ////////////////////Attributes//////////////////////// */

    private int employeeID;
    private float salary;
    private TypeOfActivity typeOfActivity;
    private enum TypeOfActivity{
        CUSTOMERSUPPORT,
        MAINTAINER,
        BOSS;
    }

    /* ////////////////////Methods////////////////////////// */

    // constructor for Employee
    public Employee(String surname, String firstName, String ZIP, String street, String houseNumber,
                    Date dateOfBirth,float salary, TypeOfActivity typeOfActivity, List list) {...}

    public static void employeeHelpsCustomer(int customerId){}

    // Used to generate Random Booleans for employeeRepairsCar
    public boolean getRandomBoolean(){...}

    public void employeeRepairsCar(Car car){
        if(car.getCarState() == car.getCarState().DAMAGED){
            boolean carSuccessfullyRepaired;
            /* *********** Repairs with the magical powers of Employee ********** */
            carSuccessfullyRepaired = getRandomBoolean();
            if (carSuccessfullyRepaired == true){
                System.out.println("We were able to fix the car");
                car.changeCarState(Car.State.OK);
            }
            else{
                System.out.println("Sorry the car is not repairable");
                car.changeCarState(Car.State.DAMAGED);
            }
        }
    }
```

Eine Verwaltungsklasse (List) erstellt, wo wir mithilfe von ArrayLists eine dynamische Verwaltung von unseren verschiedenen Objekten haben

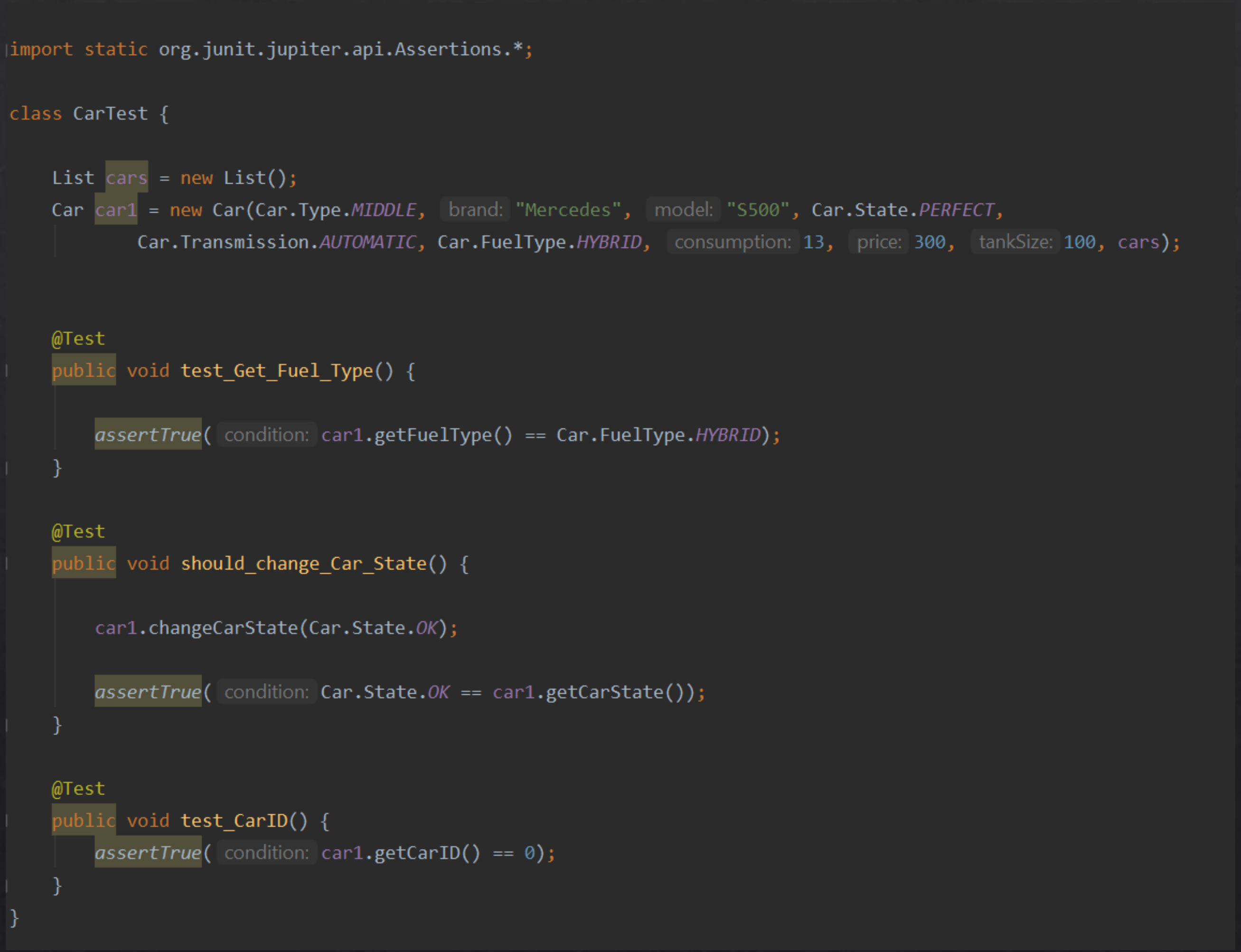


```java
import java.util.ArrayList;

public class List {

    // This Class should just create one Object (Works like a database)

    public ArrayList<Customer> customers;
    public ArrayList<Employee> employees;
    public ArrayList<Car> cars;
    public ArrayList<ParkingArea> parkingAreas;
    public ArrayList<Rental> rentals;

    /* ////////////////////Methods///////////////////////// */

    public List(){
        this.customers = new ArrayList<Customer>();
        this.employees = new ArrayList<Employee>();
        this.cars = new ArrayList<Car>();
        this.parkingAreas = new ArrayList<ParkingArea>();
        this.rentals = new ArrayList<Rental>();
    }

    public int getSizeOfCustomers() { return this.customers.size(); }

    public int getSizeOfEmployees() { return this.employees.size(); }

    public int getSizeOfCars() { return this.cars.size(); }

    public int getCarIDFromCars(int i){
        return returnCarWithIndex(i).getCarID();
    }

    public Car returnCarWithIndex(int index) { return cars.get(index); }

    public int getSizeOfParkingAreas() { return this.parkingAreas.size(); }

    public int getSizeOfRentals() { return this.rentals.size(); }
}
```

Einige Unit-Test gemacht



```
List parkingAreas = new List();
ParkingArea parkingArea1 = new ParkingArea( parkZIP: 99098, parkCity: "Erfurt",
                                            parkStreet: "Marktstraße", parkHouseNumber: "1", parkingAreas);

@BeforeEach
public void init () {
    parkingArea1.assignCarToStation( carID: 0);
}

@Test
public void should_Assign_Car_To_Station () {

    parkingArea1.assignCarToStation( carID: 0);

    assertEquals(parkingArea1.stationCarIDList.size(),  actual: 1);
}

@Test
public void should_Remove_Car_From_Station () {

    parkingArea1.removeCarFromStation( carID: 0);

    assertEquals(parkingArea1.getIndexInStationCarIDList(0), actual: 0);

}

@Test
public void Car_Should_Be_Used () {
    parkingArea1.carIsBeingUsed( carID: 0);

assertEquals(parkingArea1.notAvailableCarIDList.size(), actual: 1);
```



```
import static org.junit.jupiter.api.Assertions.*;

class CarTest {

    List cars = new List();
    Car car1 = new Car(Car.Type.MIDDLE,  brand: "Mercedes",  model: "S500", Car.State.PERFECT,
            Car.Transmission.AUTOMATIC, Car.FuelType.HYBRID,  consumption: 13,  price: 300,  tankSize: 100, cars);


    @Test
    public void test_Get_Fuel_Type() {

        assertTrue( condition: car1.getFuelType() == Car.FuelType.HYBRID);
    }

    @Test
    public void should_change_Car_State() {

        car1.changeCarState(Car.State.OK);

        assertTrue( condition: Car.State.OK == car1.getCarState());
    }

    @Test
    public void test_CarID() {
        assertTrue( condition: car1.getCarID() == 0);
    }
}
```