első feladat:

```
public class Car {
    private String plateNumber;
    private String carType;
    private int price;
    private int year;
    private int carBrandId;

    public Car(String plateNumber, String carType, int price, int year, int carBrandId) {
        this.plateNumber = plateNumber;
        this.carType = carType;
        this.price = price;
        this.year = year;
        this.carBrandId = carBrandId;
    }
```

ezek után egy getter – setter és egy string override kódot illessz be

```
    @Override
    public String toString() {
        return "Car{" + "plateNumber=" + plateNumber + ", carType=" + carType + ", price=" + price + ",
year=" + year + ", carBrandId=" + carBrandId + '}';
    }
}
```

második feladat:

```
public class CarBrand {
    private int id;
    private String brandName;
    private String description
    public CarBrand(int id, String brandName, String description) {
        this.id = id;
        this.brandName = brandName;
        this.description = description; } ( itt is getter- setter és string override)
```

hamadik feladat:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author kovacs.mark
 */
public class CarManager {

   private final String URL = "jdbc:sqlite:Secondhand.db";

   public List<Car> getAllCars() {
      try {
         Statement st = getStatement();
         List<Car> cars = new ArrayList<>();
         ResultSet resultSet = st.executeQuery(
               "SELECT * FROM Cars");
         while (resultSet.next()) {
            String plateNumber = resultSet.getString(
                  "PlateNumber");
            String carType = resultSet.getString(
                  "CarType");
            int price = resultSet.getInt(
                  "Price");
```

```
            int year = resultSet.getInt(
                    "Year");
            int carBrandId = resultSet.getInt(
                    "CarBrandId");
            Car car = new Car(plateNumber, carType,
                    price, year, carBrandId);
            cars.add(car);
        }
        return cars;
    } catch (SQLException e) {
        System.out.println("Hiba történt: " + e.getMessage());
        return null;
    }
}

public List<CarBrand> getAllCarBrands() {
    try {
        Statement st = getStatement();
        List<CarBrand> carBrands = new ArrayList<>();
        ResultSet resultSet = st.executeQuery(
                "SELECT * FROM CarBrands");
        while (resultSet.next()) {
            int id = resultSet.getInt(
                    "Id");
            String brandName = resultSet.getString(
                    "BrandName");
            String description = resultSet.getString(
                    "Description");
            CarBrand cb = new CarBrand(
                    id, brandName, description);
            carBrands.add(cb);
```

```
        }
        return carBrands;
    } catch (SQLException e) {
        System.out.println("Hiba történt: " + e.getMessage());
        return null;
    }
}

public List<CarQuery> getAll() {
    try {
        Statement st = getStatement();
        String query = "SELECT c.PlateNumber, b.BrandName, \n"
                + "c.CarType, c.Price, c.Year\n"
                + "FROM Cars as c\n"
                + "INNER JOIN CarBrands as b\n"
                + "on c.CarBrandId = b.Id";
        List<CarQuery> carQueries = new ArrayList<>();
        boolean isQuery = st.execute(query);
        ResultSet rs = st.getResultSet();

        while (rs.next()) {
            String plateNumber
                    = rs.getString("PlateNumber");
            String carType
                    = rs.getString("CarType");
            String brandName
                    = rs.getString("BrandName");
            int price = rs.getInt("Price");
            int year = rs.getInt("Year");
            carQueries.add(new CarQuery(plateNumber, carType,
                    price, year, brandName));
```

```
            }
            return carQueries;
        } catch (SQLException e) {
            return null;
        }
    }
```

negyedik feladat:

```
public class CarQuery {
    private String plateNumber;
    private String carType;
    private int price;
    private int year;
    private String brandName;

    public CarQuery(String plateNumber, String carType, int price, int year, String brandName) {
        this.plateNumber = plateNumber;
        this.carType = carType;
        this.price = price;
        this.year = year;
        this.brandName = brandName;
    } (getter – setter és string)
```

ötödik feladat:

```
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 *
 * @author kovacs.mark
 */
public class SecondHandDbCars {
```

```
public static void main(String[] args) {
    CarManager cm = new CarManager();
    Scanner scanner = new Scanner(System.in);
    String input = null;
    do {

        input = scanner.next();
        switch (input) {
            case "a":
                List<CarQuery> carQueries = cm.getAll();
                for (CarQuery c : carQueries) {
                    System.out.println(c.toString());
                }
                break;
            case "b":
                System.out.println("Legdrágább autó:");
                System.out.println(cm.maxPrice().toString());
                break;
            case "c":
                cm.insertCar(new Car("ASD-323", "Fabia",
                500000, 2005, 11));
                break;
            case "d":
                updateCarPrice(scanner, cm);
                break;
            case "e":
                deleteCar(scanner, cm);
                break;
        }
    } while (input == null || !input.equals("q"));
```