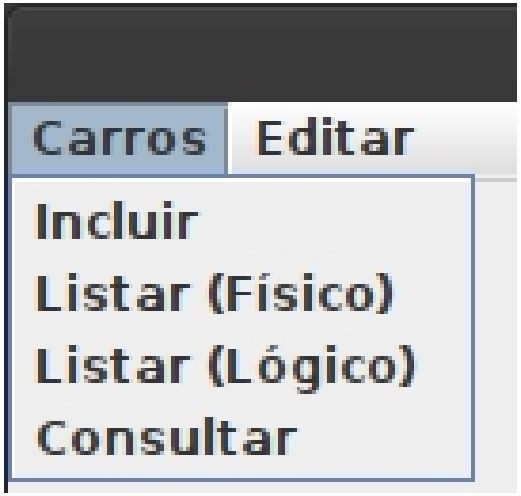
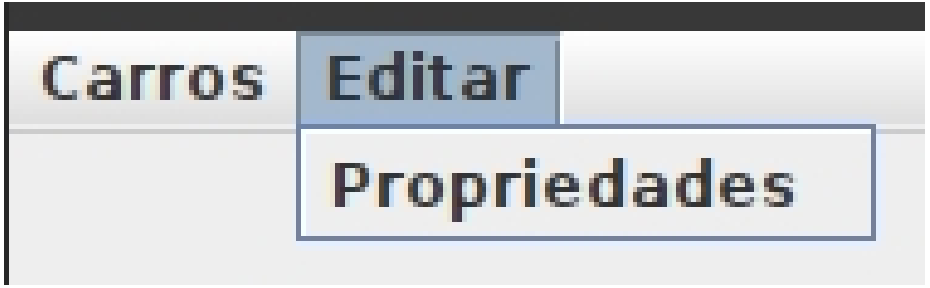
CAR-CRUD

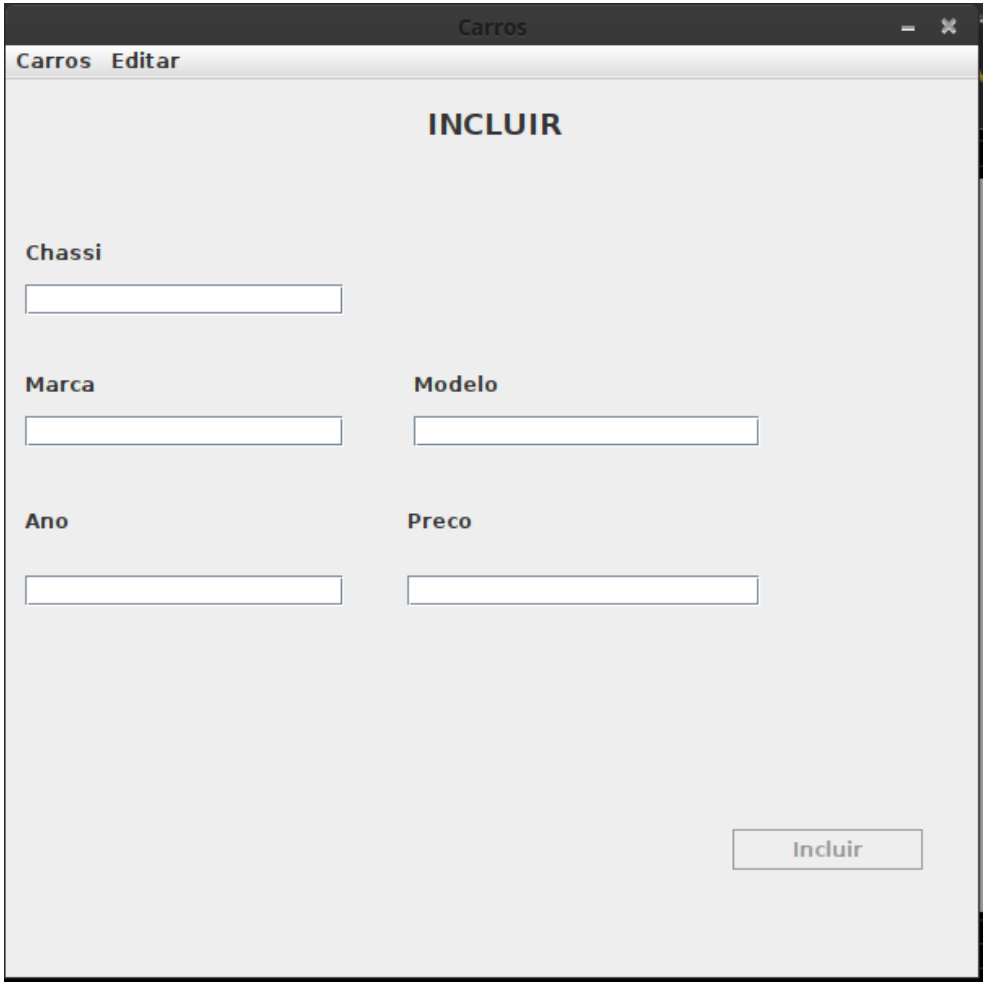
Sistema de revenda de carros utilizando java Swing e persistência em arquivo

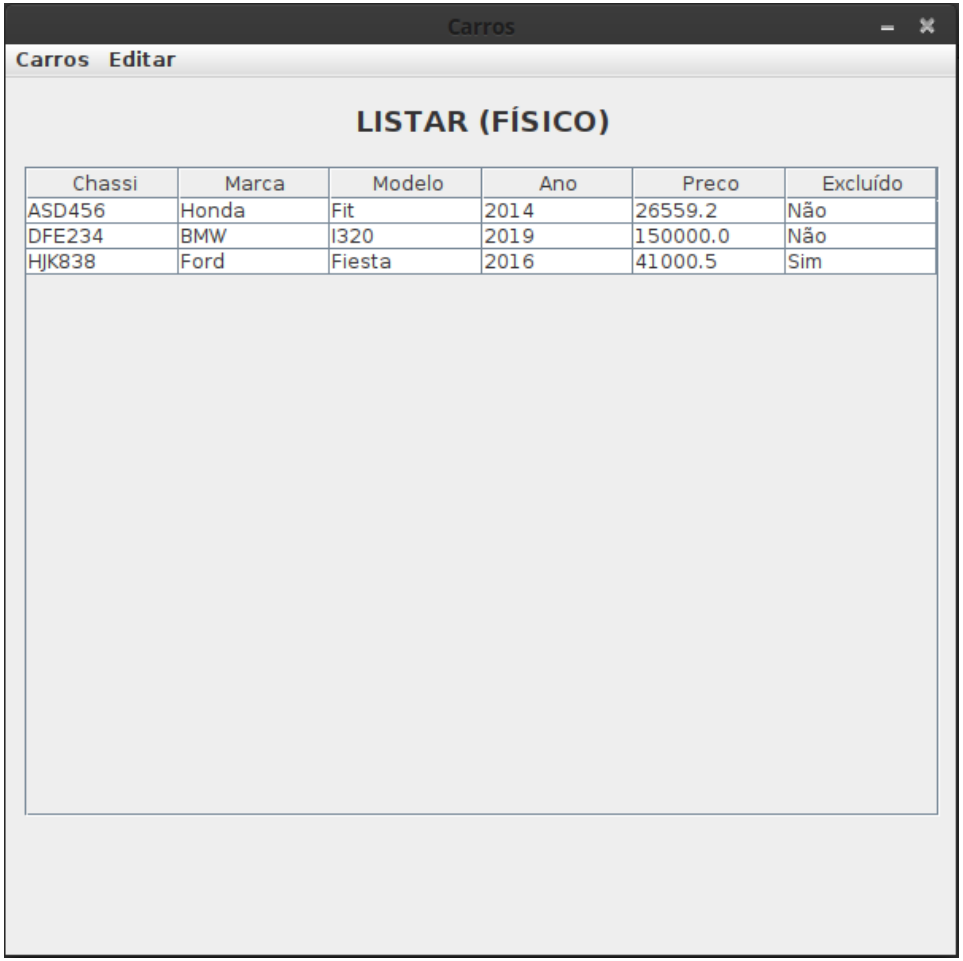
Software de revenda de carros

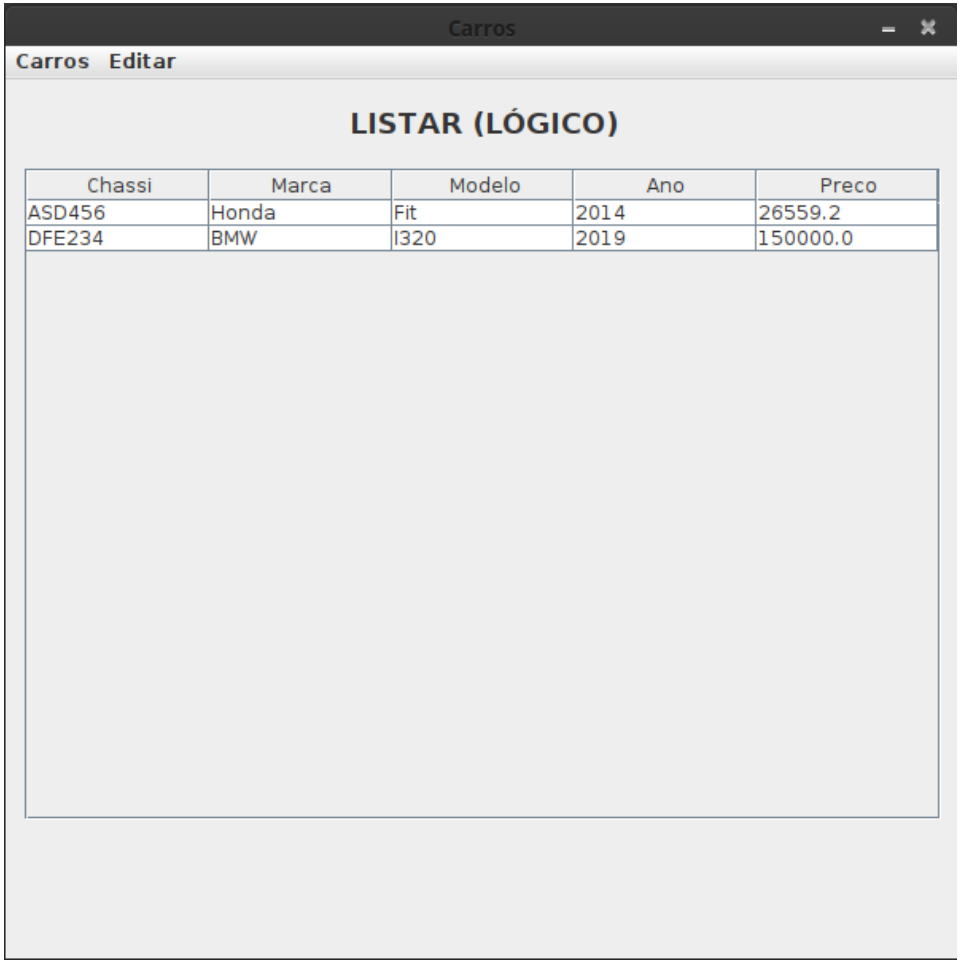
- Chassi (String, Primary Key)
- Marca
- Modelo (Obrigatório)
- Ano
- Preço

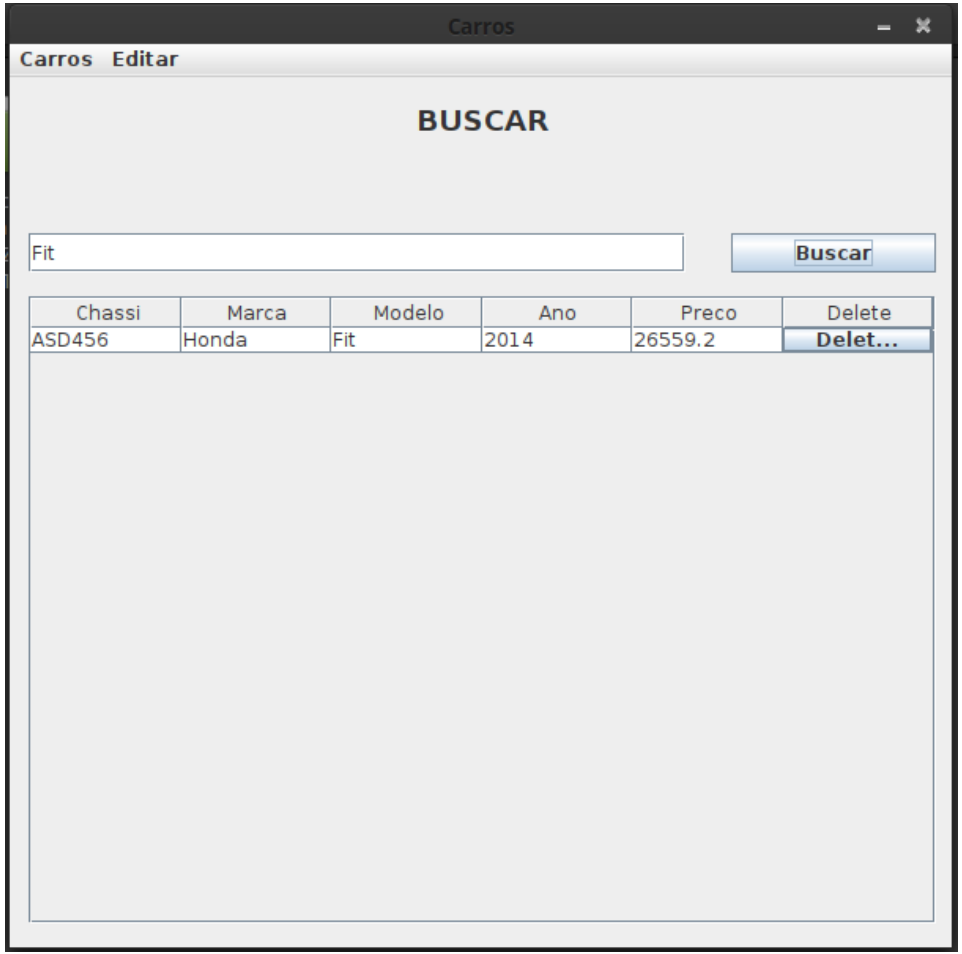

Carros
Editar
Incluir
Listar (Físico)
Listar (Lógico)
Consultar

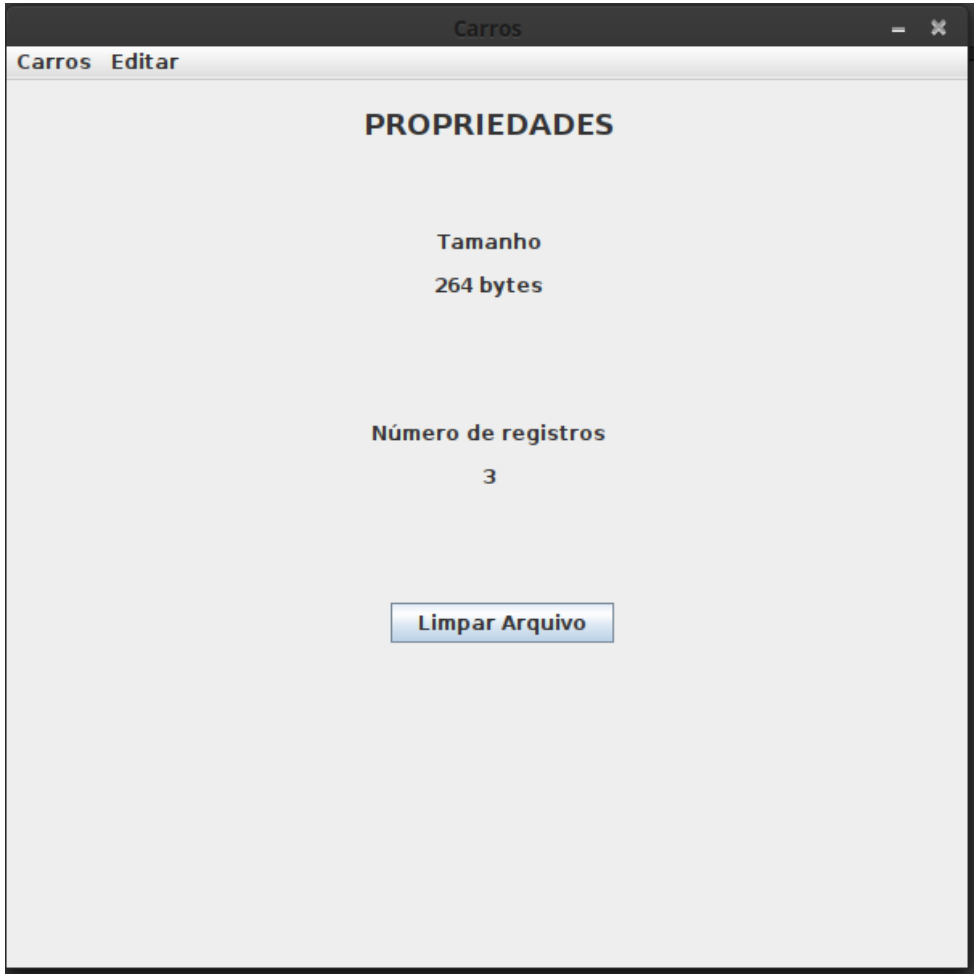

Carros
Editar
Propriedades


Carros
Carros Editar
INCLUIR
Chassi
Marca
Modelo
Ano
Preco
Incluir


Carros
Carros Editar
LISTAR (FÍSICO)
Chassi Marca Modelo Ano Preco Excluído
ASD456 Honda Fit 2014 26559.2 Não
DFE234 BMW I320 2019 150000.0 Não
HJK838 Ford Fiesta 2016 41000.5 Sim


Carros
Carros Editar
LISTAR (LÓGICO)
Chassi Marca Modelo Ano Preco
ASD456 Honda Fit 2014 26559.2
DFE234 BMW I320 2019 150000.0


Carros
Carros Editar
BUSCAR
Fit
Buscar
Chassi
Marca
Modelo
Ano
Preco
Delete
ASD456
Honda
Fit
2014
26559.2
Delet...


Carros
Carros Editar
PROPRIEDADES
Tamanho
264 bytes
Número de registros
3
Limpar Arquivo

Estrutura

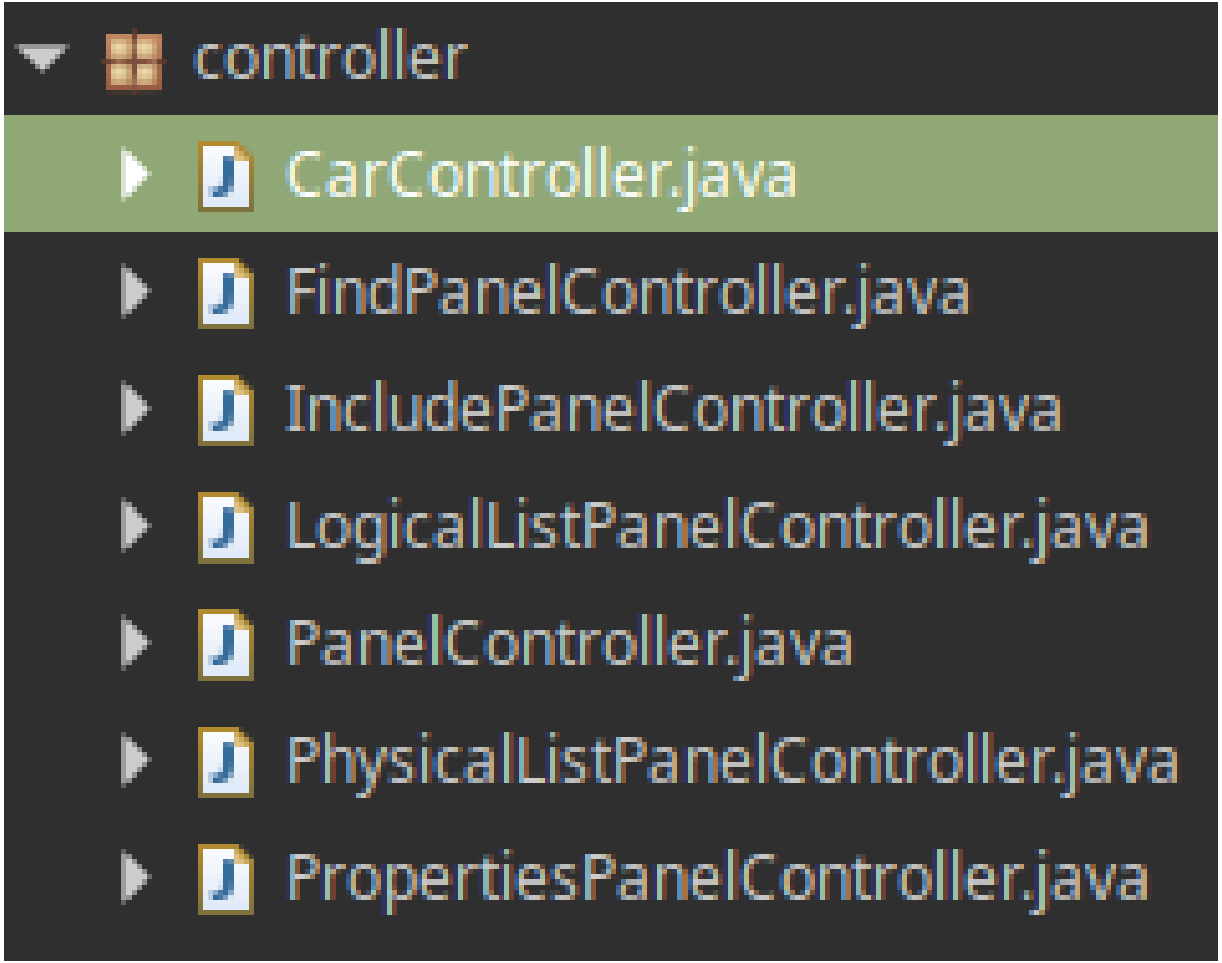

src
button
controller
entity
exception
main
persistence
service
view

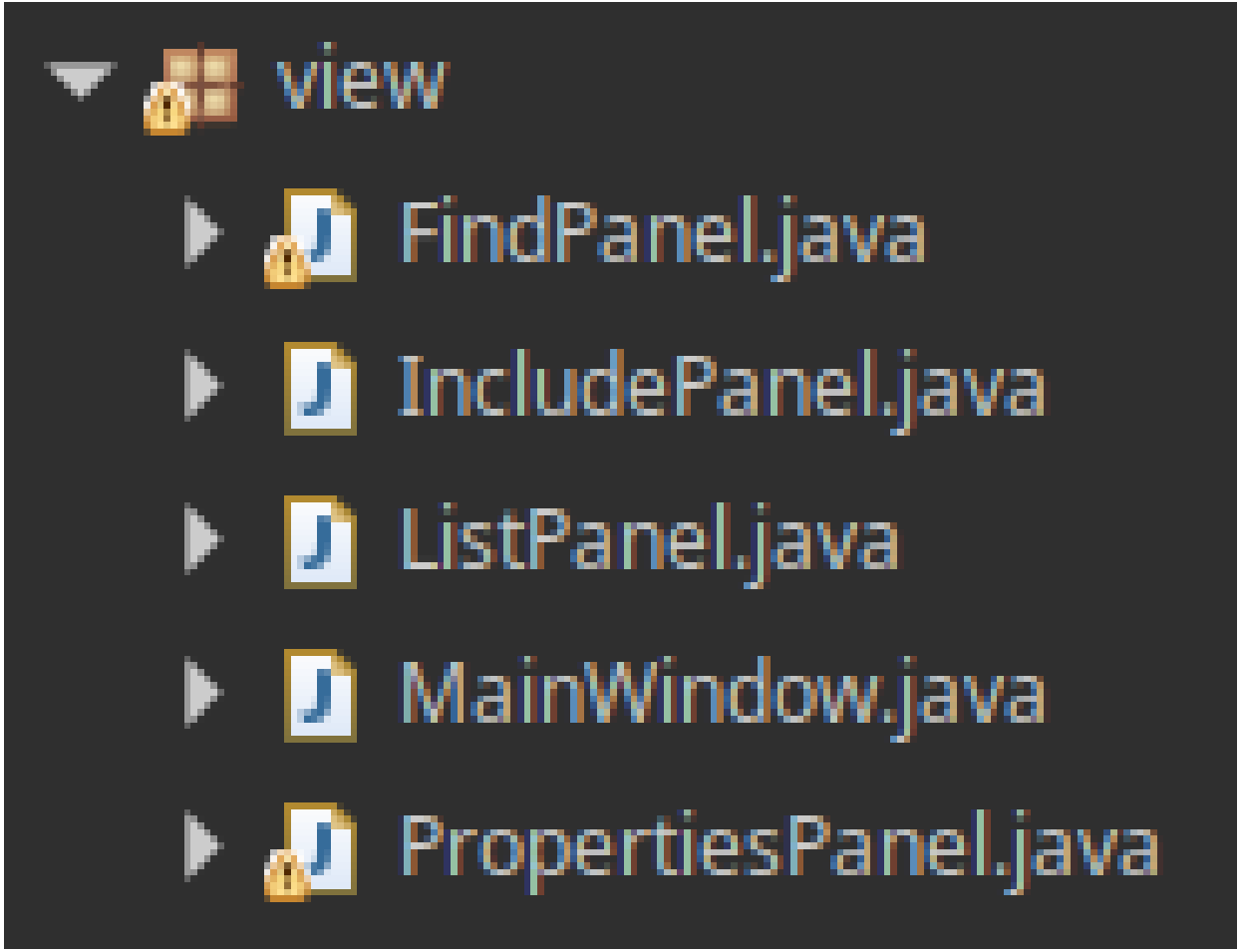
CarMain.java

```java
package main;

import controller.CarController;

public class CarMain {

    public static void main(String[] args) {
        CarController carController = new CarController();
        carController.execute();
    }

}
```

```
public void execute() {
    mainWindow = new MainWindow();
    includePanelController = new IncludePanelController();
    findPanelController = new FindPanelController();
    parentPanel = new JPanel();
    mainWindow.getContentPane().add(parentPanel);
    parentPanel.setLayout(cardLayout);
    parentPanel.add(includePanelController.includePanel, INCLUDE_PANEL);
    parentPanel.add(findPanelController.findPanel, FIND_PANEL);
    mainWindow.includeMenuItem.addActionListener(e -> showIncludePanel());
    mainWindow.findMenuItem.addActionListener(e -> showFindPanel());
    mainWindow.phyisicalListMenuItem.addActionListener(e -> showPhysicalListPanel());
    mainWindow.logicalListMenuItem.addActionListener(e -> showLogicalListPanel());
    mainWindow.propertiesMenuItem.addActionListener(e -> showPropertiesPanel());
}
```

```
private void showIncludePanel() {
    cardLayout.show(parentPanel, INCLUDE_PANEL);
}

private void showFindPanel() {
    cardLayout.show(parentPanel, FIND_PANEL);
}

private void showPhysicalListPanel() {
    physicalListPanelController = new PhysicalListPanelController();
    parentPanel.add(physicalListPanelController.physicalListPanel, PHYSICAL_LIST_PANEL);
    cardLayout.show(parentPanel, PHYSICAL_LIST_PANEL);
}

private void showLogicalListPanel() {
    logicalListPanelController = new LogicalListPanelController();
    parentPanel.add(logicalListPanelController.listPanel, LOGICAL_LIST_PANEL);
    cardLayout.show(parentPanel, LOGICAL_LIST_PANEL);
}

private void showPropertiesPanel() {
    propertiesPanelController = new PropertiesPanelController();
    parentPanel.add(propertiesPanelController.propertiesPanel, PROPERTIES_PANEL);
    cardLayout.show(parentPanel, PROPERTIES_PANEL);
}
```


controller
CarController.java
FindPanelController.java
IncludePanelController.java
LogicalListPanelController.java
PanelController.java
PhysicalListPanelController.java
PropertiesPanelController.java


view
FindPanel.java
IncludePanel.java
ListPanel.java
MainWindow.java
PropertiesPanel.java

```
public class IncludePanelController extends PanelController{

    IncludePanel includePanel;

    public IncludePanelController() {
        includePanel = new IncludePanel();
        includePanel.btnInclude.addActionListener(e2 -> insertCar());
    }

    private void insertCar() {
        String chassi = includePanel.txtChassi.getText();
```

```java
private void insertCar() {
    String chassi = includePanel.txtChassi.getText();
    String brand = includePanel.txtBrand.getText();
    String model = includePanel.txtModel.getText();
    Integer year = includePanel.txtYear.getText().equals("") ? 0 : Integer.valueOf(includePanel.txtYear.getText());
    Double price = includePanel.txtPrice.getText().equals("") ? 0 : Double.valueOf(includePanel.txtPrice.getText());
    if (chassi.equals("") || model.equals("")) {
        JOptionPane.showMessageDialog(null, "Chassi e modelo são obrigatórios");
        return;
    }
    Car newCar = new Car(chassi, brand, model, year, price);
    try {
        carService.save(newCar);
        JOptionPane.showMessageDialog(null, "Carro inserido com sucesso");
        clearFields();
    } catch (CarAlreadyExistsException caee) {
        JOptionPane.showMessageDialog(null, caee.getMessage(), "Aviso", JOptionPane.WARNING_MESSAGE);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public class LogicalListPanelController extends PanelController{

    ListPanel listPanel;
    private static final int NUMBER_OF_COLUMNS = 5;
    private static final String[] COLUMNS = {"Chassi", "Marca", "Modelo", "Ano", "Preco"};

    public LogicalListPanelController () {
        List<Car> cars = carService.list(false);
        listPanel = new ListPanel(super.buildTableData(cars, NUMBER_OF_COLUMNS, false), COLUMNS, "LISTAR (LÓGICO)");
    }

}
```

```java
public class PhysicalListPanelController extends PanelController{

    ListPanel physicalListPanel;
    private static final int NUMBER_OF_COLUMNS = 6;
    private static String[] COLUMNS = {"Chassi", "Marca", "Modelo", "Ano", "Preco", "Excluído"};

    public PhysicalListPanelController () {
        List<Car> cars = carService.list(true);
        physicalListPanel = new ListPanel(super.buildTableData(cars, NUMBER_OF_COLUMNS, true), COLUMNS, "LISTAR (FÍSICO)")
    }

}
```

```java
public class PanelController {

    CarService carService;

    public PanelController() {
        carService = new CarService();
    }

    protected String[][] buildTableData(List<Car> cars, int numberOfColumns, boolean showDeleted) {
        String[][] data = new String[cars.size()][numberOfColumns];
        int index = 0;
        for (Car car : cars) {
            data[index][0] = car.getChassi();
            data[index][1] = car.getBrand();
            data[index][2] = car.getModel();
            data[index][3] = car.getYearAsString();
            data[index][4] = car.getPriceAsString();
            if (showDeleted) data[index][5] = car.getDeletedAsString();
            index++;
        }
        return data;
    }

}
```

```java
public class FindPanelController extends PanelController{

    private static final int CHASSI_COLUMN_POSITION = 0;
    private static final int BRAND_COLUMN_POSITION = 1;
    private static final int MODEL_COLUMN_POSITION = 2;
    private static final int YEAR_COLUMN_POSITION = 3;
    private static final int PRICE_COLUMN_POSITION = 4;
    FindPanel findPanel;

    public FindPanelController() {
        findPanel = new FindPanel();
        findPanel.findButton.addActionListener(e2 -> findRegister(findPanel.txtFind.getText()));
        findPanel.tableModel.addTableModelListener(e2 -> tableChanged(e2));
    }

    public void findRegister(String text) {
        List<Car> cars = carService.findByChassiOrModel(text);
        String[] tableColumns = {"Chassi", "Marca", "Modelo", "Ano", "Preco", "Delete"};
        findPanel.tableModel.setDataVector(buildTableData(cars), tableColumns);
        ButtonRenderer buttonRenderer = new ButtonRenderer();
        findPanel.table.getColumn("Delete").setCellRenderer(buttonRenderer);
        findPanel.table.getColumn("Delete").setCellEditor(new ButtonEditor(new JCheckBox(), this));
    }
```

```java
private String[][] buildTableData(List<Car> cars) {
    String[][] data = new String[cars.size()][6];
    int index = 0;
    for (Car car : cars) {
        data[index][0] = car.getChassi();
        data[index][1] = car.getBrand();
        data[index][2] = car.getModel();
        data[index][3] = car.getYearAsString();
        data[index][4] = car.getPriceAsString();
        data[index][5] = "Delete " + car.getChassi();
        index++;
    }
    return data;
}
```

```
void tableChanged(TableModelEvent e) {
    int row = e.getFirstRow();
    int column = e.getColumn();
    if (row == -1 || column == -1)
        return;
    TableModel model = (TableModel) e.getSource();
    String chassi = (String) model.getValueAt(row, 0);
    Car car = new Car(chassi);
    Object data = model.getValueAt(row, column);
    switch (column) {
    case CHASSI_COLUMN_POSITION:
        return;
    case BRAND_COLUMN_POSITION:
        car.setBrand((String) data);
        break;
    case MODEL_COLUMN_POSITION:
        car.setModel((String) data);
        break;
    case YEAR_COLUMN_POSITION:
        car.setYear(Integer.valueOf((String)data));
        break;
    case PRICE_COLUMN_POSITION:
        car.setPrice(Double.valueOf((String) data));
        break;
    default:
        return;
    }
    carService.update(car);
}
```

```java
public class PropertiesPanelController extends PanelController {

    PropertiesPanel propertiesPanel;

    public PropertiesPanelController() {
        long[] properties = carService.getProperties();
        long fileSize = properties[0];
        long numberOfRegisters = properties[1];
        propertiesPanel = new PropertiesPanel(fileSize, numberOfRegisters);
        propertiesPanel.btnClearfile.addActionListener(e2 -> deleteFile());
    }

    private void deleteFile() {
        carService.clearFile();
        propertiesPanel.lblSize.setText("0 bytes");
        propertiesPanel.lblNumberOfRegisters.setText("0");
    }

}
```

```java
public class CarService {

    FileManager fileManager;

    public CarService() {
        fileManager = new FileManager();
    }

    public void save(Car car) throws Exception {
        fileManager.saveCar(car);
    }

    public void update(Car car) {
        fileManager.updateCar(car);
    }

    public void delete(Car car) {
        fileManager.deleteCar(car);
    }

    public List<Car> findByChassiOrModel(String text) {
        return fileManager.findCarsByChassiOrModel(text);
    }

    public List<Car> list(boolean showDeleted) {
        return fileManager.list(showDeleted);
    }
```

```java
    public void openFile() {
        try {
            File fileName = new File(FILE_NAME);
            this.raf = new RandomAccessFile(fileName, "rw");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            this.raf = null;
        }
    }

    public void closeFile() {
        try {
            this.raf.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
RandomAccessFile raf;
static final String FILE_NAME = "file.dat";
static final int CHASSI_FIELD_SIZE = 24;
static final int BRAND_FIELD_SIZE = 24;
static final int MODEL_FIELD_SIZE = 24;
static final int YEAR_FIELD_SIZE = 4;
static final int PRICE_FIELD_SIZE = 8;
static final int DELETED_FIELD_SIZE = 4;
static final int REGISTER_SIZE = CHASSI_FIELD_SIZE + BRAND_FIELD_SIZE +
                                 MODEL_FIELD_SIZE + YEAR_FIELD_SIZE +
                                 PRICE_FIELD_SIZE + DELETED_FIELD_SIZE;
static final int CHASSI_STRING_SIZE = CHASSI_FIELD_SIZE / 2;
static final int BRAND_STRING_SIZE = BRAND_FIELD_SIZE / 2;
static final int MODEL_STRING_SIZE = MODEL_FIELD_SIZE / 2;
```

```java
public void saveCar(Car car) throws Exception {
    openFile();
    if (doesChassiExist(car.getChassi())) {
        closeFile();
        throw new CarAlreadyExistsException();
    }
    try {
        this.raf.seek(this.raf.length());
        this.writeString(car.getChassi(), CHASSI_STRING_SIZE);
        this.writeString(car.getBrand(), BRAND_STRING_SIZE);
        this.writeString(car.getModel(), MODEL_STRING_SIZE);
        this.raf.writeInt(car.getYear() == null ? 0 : car.getYear());
        this.raf.writeDouble(car.getPrice() == null ? 0 : car.getPrice());
        this.raf.writeInt(Car.NOT_DELETED);
    } catch (IOException e) {
        e.printStackTrace();
        throw new Exception("Erro ao salvar o carro");
    } finally {
        closeFile();
    }
}
```

```java
/**
 * Retorn true ou false caso o chassi passado como parâmetro exista ou não.
 * @param chassi
 * @return boolean
 */
private boolean doesChassiExist(String chassi) {
    try {
        raf.seek(0);
        int registerIndex = 1;
        while (raf.getFilePointer() < raf.length()) {
            String fileChassi = this.readString(CHASSI_STRING_SIZE);
            if (chassi.equals(fileChassi.trim())) {
                return true;
            }
            raf.seek(registerIndex * REGISTER_SIZE);
            registerIndex++;
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}
```

```java
public void updateCar(Car car) {
    openFile();
    try {
        int registerIndex = 1;
        while (raf.getFilePointer() < raf.length()) {
            String chassi = this.readString(CHASSI_STRING_SIZE);
            if (car.getChassi().trim().equals(chassi.trim())) {
                if (car.getBrand() != null) {
                    this.writeString(car.getBrand(), BRAND_STRING_SIZE);
                } else if (car.getModel() != null) {
                    raf.seek(raf.getFilePointer() + BRAND_FIELD_SIZE);
                    this.writeString(car.getModel(), MODEL_STRING_SIZE);
                } else if (car.getYear() != null) {
                    raf.seek(raf.getFilePointer() + BRAND_FIELD_SIZE + MODEL_FIELD_SIZE);
                    raf.writeInt(car.getYear());
                } else if (car.getPrice() != null) {
                    raf.seek(raf.getFilePointer() + BRAND_FIELD_SIZE + MODEL_FIELD_SIZE + YEAR_FIELD_SIZE);
                    raf.writeDouble(car.getPrice());
                } else if (car.getDeleted() != null) {
                    raf.seek(raf.getFilePointer() + BRAND_FIELD_SIZE + MODEL_FIELD_SIZE + YEAR_FIELD_SIZE
                            + PRICE_FIELD_SIZE);
                    raf.writeInt(Car.DELETED);
                }
                break;
            }
            raf.seek(registerIndex * REGISTER_SIZE);
            registerIndex++;
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        closeFile();
    }
}
```

```java
/**
 * Retornar lista de carros em que o chassi ou modelo dão match com a string
 * @param text
 * @return
 */
public List<Car> findCarsByChassiOrModel(String text) {
    if (text.isEmpty()) return new ArrayList<Car>();
    openFile();
    List<Car> cars = new ArrayList<Car>();
    try {
        raf.seek(0);
        while (raf.getFilePointer() < raf.length()) {
            String chassi = this.readString(CHASSI_STRING_SIZE);
            String brand = this.readString(BRAND_STRING_SIZE);
            String model = this.readString(MODEL_STRING_SIZE);
            int year = raf.readInt();
            double price = raf.readDouble();
            int deleted = raf.readInt();
            if (deleted == Car.NOT_DELETED && (chassi.contains(text) || model.contains(text))) {
                cars.add(new Car(chassi, brand, model, year, price, deleted));
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    closeFile();
    return cars;
}
```

```java
/**
 * Retorna todos os carros salvos em arquivo
 * @param showDeleted Se inclui os deletados ou não
 * @return
 */
public List<Car> list(boolean showDeleted) {
    openFile();
    List<Car> cars = new ArrayList<Car>();
    try {
        raf.seek(0);
        while (raf.getFilePointer() < raf.length()) {
            String chassi = this.readString(CHASSI_STRING_SIZE);
            String brand = this.readString(BRAND_STRING_SIZE);
            String model = this.readString(MODEL_STRING_SIZE);
            int year = raf.readInt();
            double price = raf.readDouble();
            int deleted = raf.readInt();
            if (deleted == Car.NOT_DELETED || (deleted == Car.DELETED && showDeleted)) {
                cars.add(new Car(chassi, brand, model, year, price, deleted));
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        closeFile();
    }
    return cars;
}
```

```java
/**
 * Retorna as propriedades do arquivo em um array de tamanho 2 sendo:
 * [0] = tamanho do arquivo;
 * [1] = número de registros;
 * @return
 */
public long[] getProperties() {
    openFile();
    long[] properties = new long[2];
    try {
        properties[0] = raf.length();
        properties[1] = raf.length() / REGISTER_SIZE;
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        closeFile();
    }
    return properties;
}

public void deleteFile() {
    new File(FILE_NAME).delete();
}
```

Obrigado!