4 Лабораторная работа №4
Исследование возможностей разработки пользовательского интерфейса в Java приложениях с использованием библиотеки Swing

4.1. Цель работы

В ходе выполнения данной лабораторной работы необходимо ознакомиться с особенностями инструментария библиотеки Swing для создания графического интерфейса приложений на языке Java и приобрести практические навыки создания Java-программ с графическим интерфейсом, позволяющим пользователю осуществлять взаимодействие с программой: задавать исходные данные, просматривать результаты работы программы в удобном виде.

4.2. Постановка задачи

Был выдан вариант 13.

Необходимо создать Java приложение с графическим интерфейсом пользователя, реализующее добавление, редактирование, сортировку и удаление данных заданного по варианту типа информации Auto. Данные отображать в виде таблицы. Реализовать поля ввода для добавления и редактирования новых записей. Предусмотреть возможность загрузки информации из текстового файла и сохранения в текстовый файл.

4.3. Ход выполнения работы

4.3.1. Текст программы

Программа составлена на языке Java.

Листинг 1 — Содержимое основного класса App

```
package org.cory7666.lab4;

public class App
{
  public static void main (String[] args)
  {
    MainWindow.main(args);
  }
}
```

Листинг 2 — Содержимое класса MainWindow

```
package org.cory7666.lab4;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.GridLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.ListSelectionModel;
import javax.swing.border.EmptyBorder;

public class MainWindow extends JFrame
{
  private static final long serialVersionUID = -26268648356920251L;

  private final AutoTableModel tableModel = new AutoTableModel();

  private final JPanel rootPane = new JPanel();
  private final JPanel panelWithButtons = new JPanel();
  private final JPanel panelWithFields = new JPanel();
  private final JTable dataTable = new JTable(tableModel);
  private final JScrollPane scrollPane = new JScrollPane(dataTable);

  private final JButton addRecordButton = new JButton("Добавить запись"),
    updateRecordButton = new JButton("Изменить запись"), deleteRecordButton = new JButton("Удалить
выделенное"),
    chooseInputFileButton = new JButton("Загрузить данные"),
    chooseOutputFileButton = new JButton("Выгрузить данные");

  private final JLabel brandFieldLabel = new JLabel("Марка"), yearFieldLabel = new JLabel("Год
выпуска"),
    engineVolumeFieldLabel = new JLabel("Объём двигателя"),
    maxSpeedFieldLabel = new JLabel("Максимальная скорость");
  private final TextFieldGroup textFieldGroup = new TextFieldGroup();

  private final MainWindowController controller = new MainWindowController(dataTable, textFieldGroup);

  /**
   * Launch the application.
   */
  public static void main (String[] args)
  {
    EventQueue.invokeLater(new Runnable() {
      public void run ()
      {
        try
        {
          MainWindow frame = new MainWindow();
          frame.setVisible(true);
          frame.pack();
        }
        catch (Exception e)
        {
          e.printStackTrace();
```

```
        }
      }
    });
  }

  /**
   * Create the frame.
   */
  public MainWindow ()
  {
    /* Окно приложения */
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("Автомобили Ltd.");
    setBounds(100, 100, 450, 300);

    /* Корневая панель */
    rootPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    rootPane.setLayout(new BorderLayout(0, 0));
    setContentPane(rootPane);

    /* Прокручеваемая панель */
    rootPane.add(scrollPane, BorderLayout.CENTER);

    /* Панель с кнопками */
    rootPane.add(panelWithButtons, BorderLayout.EAST);
    panelWithButtons.setLayout(new BoxLayout(panelWithButtons, BoxLayout.Y_AXIS));
    panelWithButtons.add(addRecordButton);
    panelWithButtons.add(updateRecordButton);
    panelWithButtons.add(deleteRecordButton);
    panelWithButtons.add(chooseInputFileButton);
    panelWithButtons.add(chooseOutputFileButton);

    /* Панель с полями ввода */
    GridLayout gridLayout = new GridLayout(0, 2);
    panelWithFields.setLayout(gridLayout);
    rootPane.add(panelWithFields, BorderLayout.NORTH);

    panelWithFields.add(brandFieldLabel);
    panelWithFields.add(textFieldGroup.brandField);
    panelWithFields.add(yearFieldLabel);
    panelWithFields.add(textFieldGroup.yearField);
    panelWithFields.add(engineVolumeFieldLabel);
    panelWithFields.add(textFieldGroup.engineVolumeField);
    panelWithFields.add(maxSpeedFieldLabel);
    panelWithFields.add(textFieldGroup.maxSpeedField);

    /* Таблица */
    dataTable.getSelectionModel().setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

    setHandlers();
  }

  private void setHandlers ()
  {
    addRecordButton.addActionListener(controller::onAddRecord);
    updateRecordButton.addActionListener(controller::onUpdateRecord);
    deleteRecordButton.addActionListener(controller::onDeleteRecord);
    chooseInputFileButton.addActionListener(controller::onLoadDataFromFileAction);
    chooseOutputFileButton.addActionListener(controller::onSaveDataAction);
    dataTable.addMouseListener(new MouseAdapter() {
      @Override
      public void mouseClicked (MouseEvent event)
      {
        controller.onRowSelectedAction(event);
      }
    });
  }
}
```

Листинг 3 — Содержимое класса MainWindowController

```
package org.cory7666.lab4;

import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
```

```
import javax.swing.JTable;

public class MainWindowController
{
  private final JTable table;
  private final TextFieldGroup textFieldGroup;

  public MainWindowController (JTable table, TextFieldGroup textFieldGroup)
  {
    this.table = table;
    this.textFieldGroup = textFieldGroup;
  }

  public void onAddRecord (ActionEvent e)
  {
    try
    {
      Auto t = textFieldGroup.constructObjectFromTextFields();
      Logger.debug(getClass(), "Добавление в таблицу " + t + ".");
      ((AutoTableModel) table.getModel()).addRows(textFieldGroup.constructObjectFromTextFields());
    }
    catch (NumberFormatException ex)
    {
      Logger.error(getClass(), "Невозможно добавить объект: Одной из полей заполнено неправильно.");
    }
    catch (Exception ex)
    {
      Logger.error(getClass(), "Невозможно добавить объект: " + ex.getMessage() + ".");
    }
  }

  public void onUpdateRecord (ActionEvent e)
  {
    Logger.debug(getClass(), "Запрос на замену записи #" + table.getSelectedRow() + ".");
    try
    {
      try
      {
        if (table.getSelectedRow() >= 0)
        {
          var model = ((AutoTableModel) table.getModel());
          model.deleteRow(table.getSelectedRow());
          model.addRows(textFieldGroup.constructObjectFromTextFields());
        }
        else
        {
          throw new IllegalArgumentException("Нечего менять");
        }
      }
      catch (NumberFormatException ex)
      {
        throw new IllegalArgumentException("Одной из полей заполнено неправильно");
      }
    }
    catch (Exception ex)
    {
      Logger.error(getClass(), "Невозможно изменить запись: " + ex.getMessage() + ".");
    }
  }

  public void onDeleteRecord (ActionEvent e)
  {
    Logger.debug(getClass(), "Запрос на удаление записи #" + table.getSelectedRow() + ".");
    ((AutoTableModel) table.getModel()).deleteRow(table.getSelectedRow());
  }

  public void onLoadDataFromFileAction (ActionEvent e)
  {
    Logger.debug(getClass(), "Запрос на загрузку данных из файла.");
    try
    {
      ((AutoTableModel) table.getModel())
        .replaceDataWithIfPresent(
          new DataFile(new UserInteractionFileCase().getReadableFileLocation()).readAll());
    }
    catch (NullPointerException ex)
    {}
    catch (Exception ex)
```

```
    {
      Logger.error(getClass(), "Ошибка: " + ex.getMessage() + ".");
    }
  }

  public void onSaveDataAction (ActionEvent event)
  {
    Logger.debug(getClass(), "Запрос на сохранение данных в файл.");
    try
    {
      new DataFile(new UserInteractionFileCase().getSaveFileLocation())
        .writeAll(((AutoTableModel) table.getModel()).getStorableDataIter());
    }
    catch (NullPointerException ex)
    {}
    catch (Exception ex)
    {
      Logger.error(getClass(), "Невозможно сохранить данные: " + ex.getMessage() + ",");
    }
  }

  public void onRowSelectedAction (MouseEvent event)
  {
    textFieldGroup.fillTextFieldDataWith(((AutoTableModel)
table.getModel()).getRow(table.getSelectedRow()));
  }
}
```

Листинг 4 — Содержимое класса Logger

```
package org.cory7666.lab4;

import javax.swing.JOptionPane;

public class Logger
{
  public static int logLevel = 0;

  public static void debug (Class<?> who, String message)
  {
    if (logLevel <= 0)
    {
      postMessage("DEBUG", who, message);
    }
  }

  public static void error (Class<?> who, String message)
  {
    if (logLevel <= 2)
    {
      postMessage("ERROR", who, message);
      JOptionPane.showMessageDialog(null, message);
    }
  }

  private synchronized static void postMessage (String panicLevel, Class<?> who, String message)
  {
    System.out
      .println(
        String.format("[%s] [%s/%s] %s", panicLevel, Thread.currentThread().getName(), who.getName(),
message));
  }
}
```

Листинг 5 — Содержимое класса AutoTableModel

```
package org.cory7666.lab4;

import java.util.Arrays;
import java.util.Collection;
import java.util.HashSet;
import java.util.Iterator;
import java.util.NoSuchElementException;
import java.util.Set;
```

```
import javax.swing.table.AbstractTableModel;

public class AutoTableModel extends AbstractTableModel
{
  private static final long serialVersionUID = 6181788579049573897L;
  private final Set<Auto> data = new HashSet<>();
  private final String[] header = { "Марка", "Год выпуска", "Объём двигателя", "Максимальная
скорость" };

  @Override
  public int getRowCount ()
  { return data.size(); }

  @Override
  public int getColumnCount ()
  { return 4; }

  @Override
  public Object getValueAt (int rowIndex, int columnIndex)
  {
    try
    {
      Auto element = getRow(rowIndex);
      return switch (columnIndex)
      {
      case 0 -> element.brand;
      case 1 -> element.year;
      case 2 -> element.engineVolume;
      case 3 -> element.maxSpeed;
      default -> "";
      };
    }
    catch (Exception e)
    {
      return "";
    }
  }

  @Override
  public String getColumnName (int column)
  {
    try
    {
      return header[column];
    }
    catch (Exception ex)
    {
      return "";
    }
  }

  public Auto getRow (int index)
  {
    int counter = 0;
    for (var t : data)
    {
      if (counter++ == index)
      {
        return t;
      }
    }
    throw new NoSuchElementException("Строки с запрошенным индексом не существует");
  }

  public Iterator<Auto> getStorableDataIter ()
  { return data.iterator(); }

  public void addRows (Collection<Auto> rows)
  {
    rows.forEach(data::add);
    fireTableDataChanged();
  }

  public void addRows (Auto... rows)
  {
    addRows(Arrays.asList(rows));
  }
```

```
  public void deleteRow (int row)
  {
    int counter = 0;
    for (var iter = data.iterator(); iter.hasNext();)
    {
      iter.next();
      if (counter++ == row)
      {
        iter.remove();
        fireTableDataChanged();
        break;
      }
    }
  }

  public void deleteAllRows ()
  {
    data.clear();
    fireTableDataChanged();
  }

  public void replaceDataWithIfPresent (Collection<Auto> rows)
  {
    if (rows.size() > 0)
    {
      deleteAllRows();
      addRows(rows);
    }
  }

  public void replaceDataWithIfPresent (Auto... rows)
  {
    replaceDataWithIfPresent(Arrays.asList(rows));
  }
}
```

Листинг 6 — Содержимое класса TextFieldGroup

```
package org.cory7666.lab4;

import javax.swing.JTextField;

public class TextFieldGroup
{
  public final JTextField brandField = new JTextField();
  public final JTextField yearField = new JTextField();
  public final JTextField engineVolumeField = new JTextField();
  public final JTextField maxSpeedField = new JTextField();

  public Auto constructObjectFromTextFields ()
  {
    var x = new Auto(brandField.getText(), Integer.parseInt(yearField.getText()),
      Float.parseFloat(engineVolumeField.getText()), Integer.parseInt(maxSpeedField.getText()));
    return x;
  }

  public void fillTextFieldDataWith (Auto x)
  {
    brandField.setText(x.brand.toString());
    yearField.setText(x.year.toString());
    engineVolumeField.setText(x.engineVolume.toString());
    maxSpeedField.setText(x.maxSpeed.toString());
  }

  public void clearFields ()
  {
    brandField.setText("");
    yearField.setText("");
    engineVolumeField.setText("");
    maxSpeedField.setText("");
  }
}
```

Листинг 7 — Содержимое класса DataFile

```
package org.cory7666.lab4;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import com.opencsv.bean.CsvToBeanBuilder;
import com.opencsv.bean.StatefulBeanToCsvBuilder;
import com.opencsv.exceptions.CsvDataTypeMismatchException;
import com.opencsv.exceptions.CsvRequiredFieldEmptyException;

public class DataFile
{
  private final File file;

  public DataFile (File file)
  {
    this.file = file;
  }

  public Set<Auto> readAll () throws IOException
  {
    try (var buffReader = Files.newBufferedReader(file.toPath()))
    {
      return new HashSet<>(
        new
CsvToBeanBuilder<Auto>(buffReader).withSeparator(';').withType(Auto.class).build().parse());
    }
  }

  public void writeAll (Iterator<Auto> collection) throws IOException
  {
    try (var buffWriter = Files.newBufferedWriter(file.toPath()))
    {
      new StatefulBeanToCsvBuilder<Auto>(buffWriter).withSeparator(';').build().write(collection);;
    }
    catch (CsvDataTypeMismatchException | CsvRequiredFieldEmptyException ex)
    {
      Logger.error(getClass(), ex.getMessage());
    }
  }
}
```

Листинг 8 — Содержимое класса UserInteractionFileCase

```
package org.cory7666.lab4;
import java.io.File;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;
public class UserInteractionFileCase
{
  private final JFileChooser chooser;

  public UserInteractionFileCase ()
  {
    chooser = new JFileChooser();
    chooser.setFileFilter(new FileNameExtensionFilter("Файл CSV", "csv"));
  }
  public File getReadableFileLocation ()
  {
    chooser.setDialogTitle("Открыть файл");
    chooser.showOpenDialog(null);
    return chooser.getSelectedFile();
  }
  public File getSaveFileLocation ()
  {
    chooser.setDialogTitle("Сохранить");
    chooser.showSaveDialog(null);
    return chooser.getSelectedFile();
  }
}
```

4.3.2. Результаты тестирования

Программа была скомпилирована и запущена. Рисунок 1 демонстрирует главное окно программы. По нажатию кнопки «Загрузить данные» было создано окно выбора файла с данными (рисунок 2). После подтверждения выбора данные автоматически были загружены в таблицу (рисунок 3).

Было проведено редактирование записи (рисунок 4) и попытка добавления записи с одним незаполненным полем и выводом соответствующей ошибки (рисунок 5).

После работы данные были сохранены при нажатии на кнопку «Выгрузить данные» (рисунок 6).

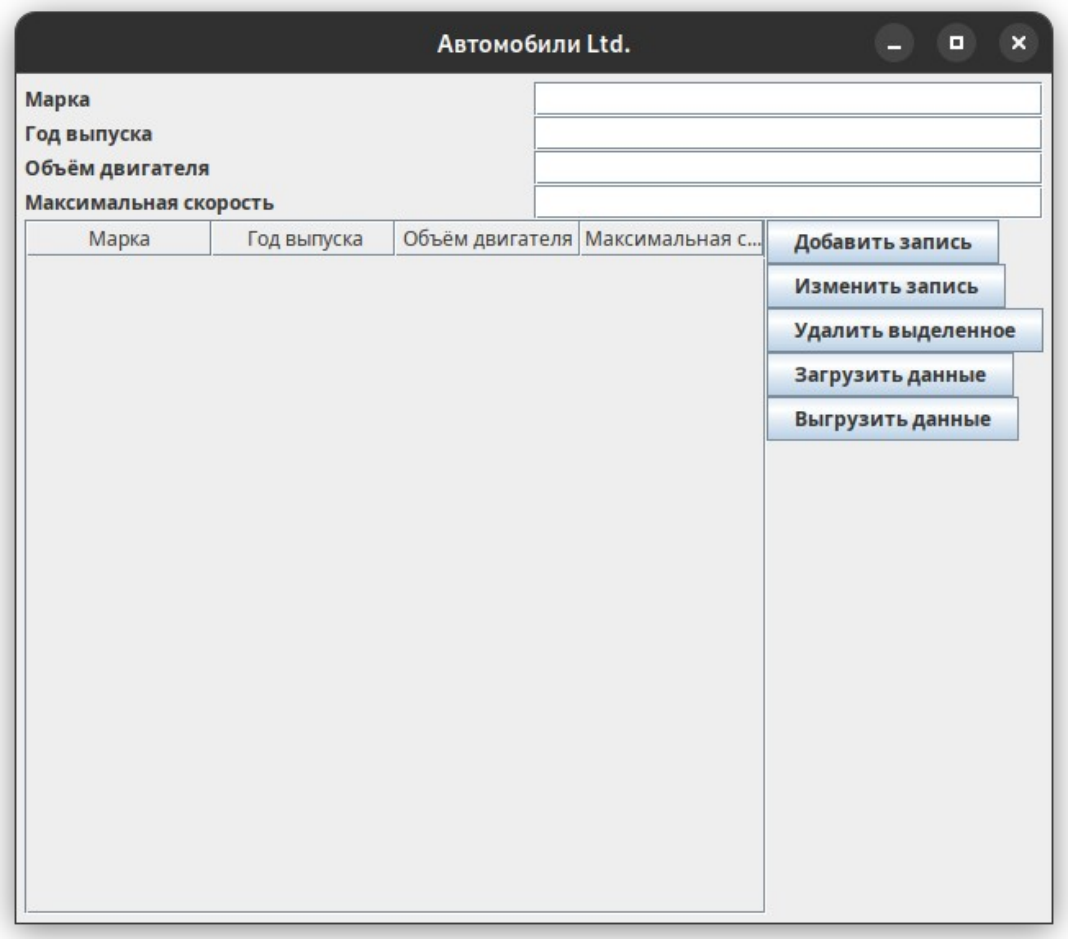

Рисунок 1 — Главное окно программы

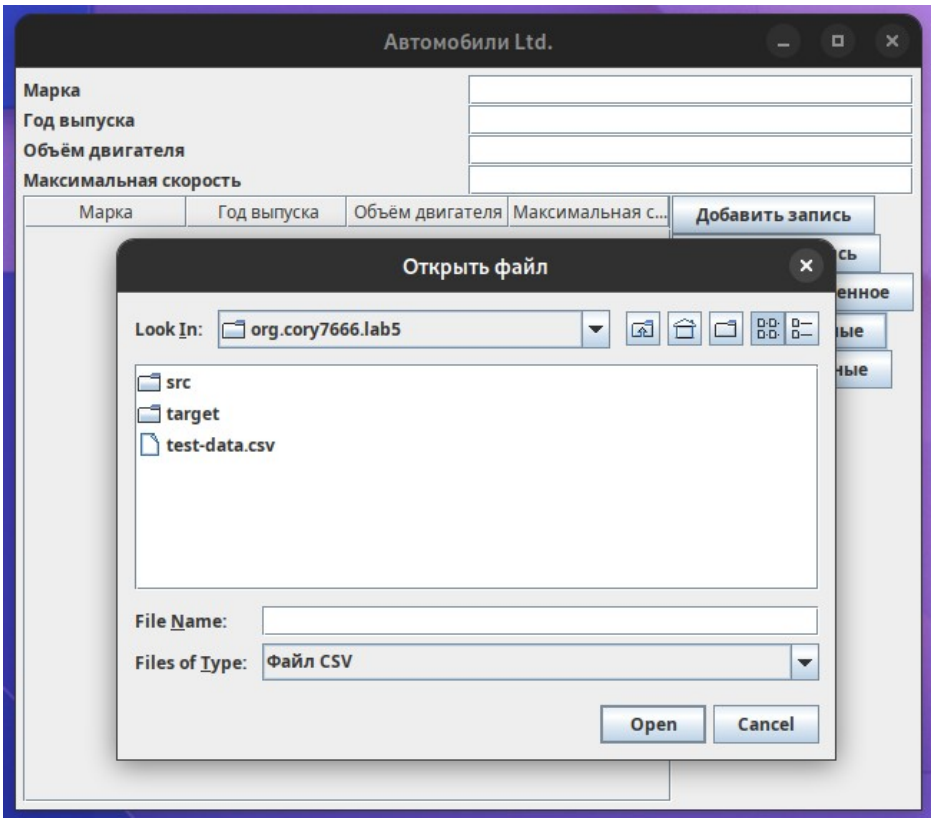

Рисунок 2 — Выбор файла с данными для загрузки

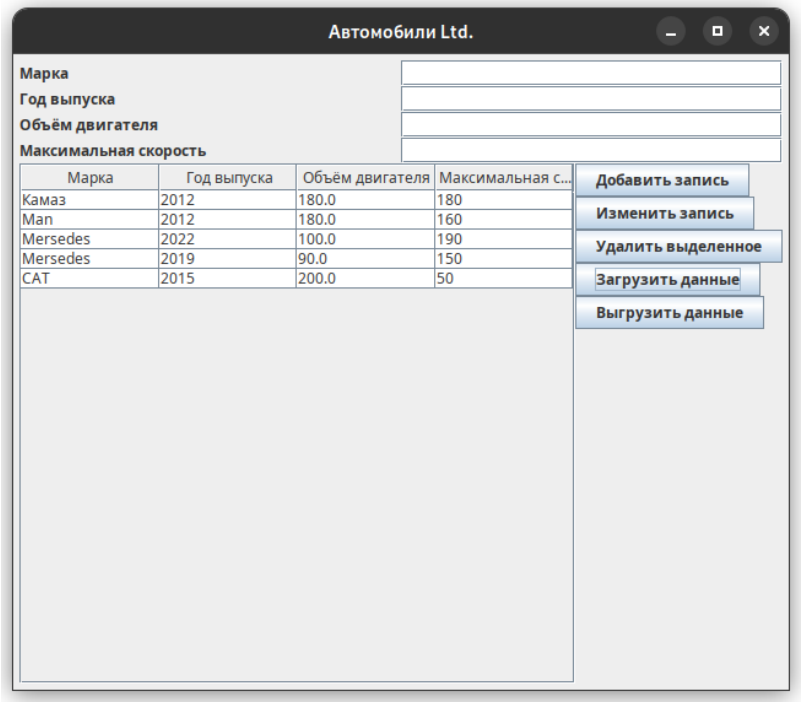

Рисунок 3 — Главное окно программы с загруженными данными

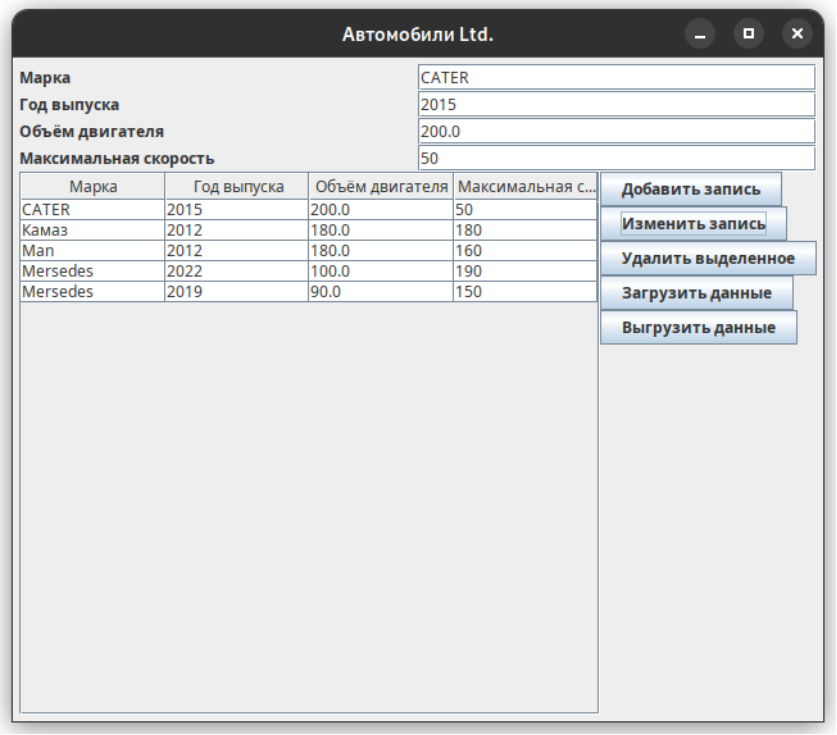

Рисунок 4 — Редактирование записи

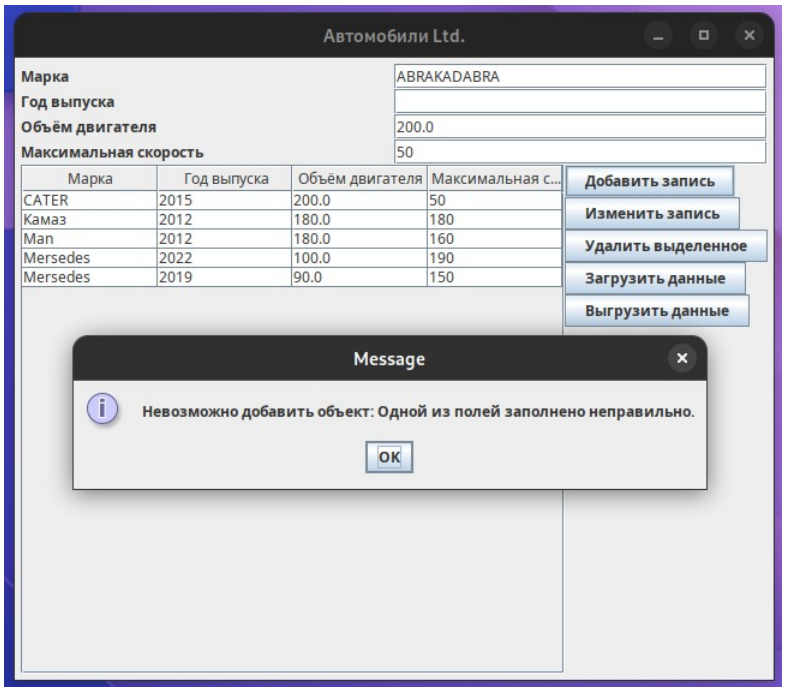

Рисунок 5 — Добавление элемента в таблицу

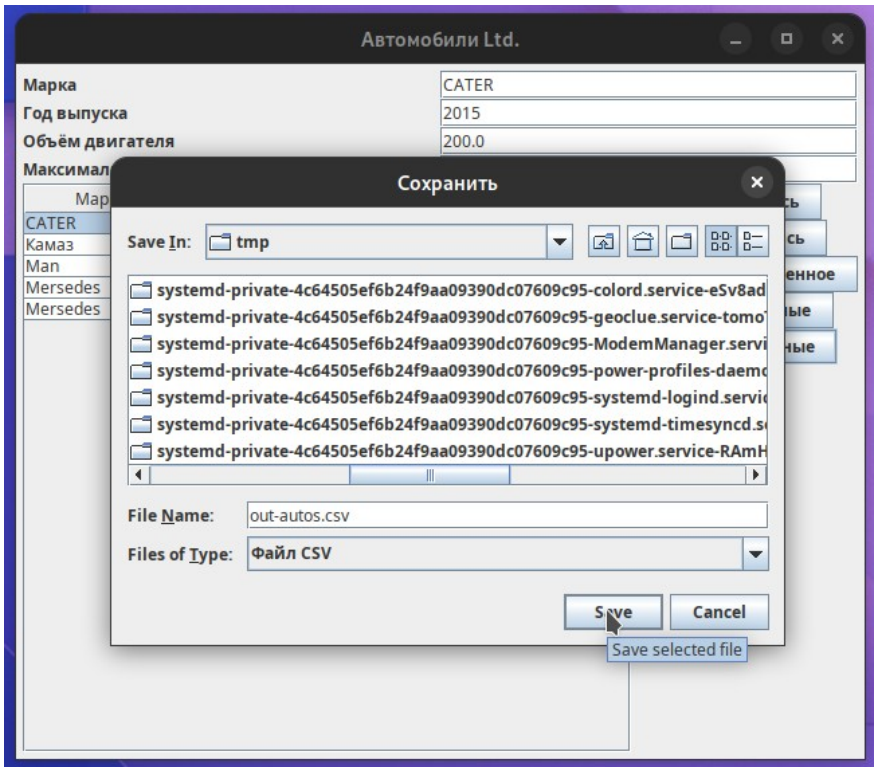

Рисунок 6 — Сохранение данных в файл

Вывод

При выполнении данной лабораторной работы были получены навыки создания графического интерфейса на основе платформы Swing. Были получены навыки создания таблиц, импорта/экспорта данных, добавления данных в таблицу.