МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра вычислительной техники

ОТЧЕТ

по лабораторной работе № 3
по дисциплине «Объектно-ориентированное программирование»
Тема: Обработка событий

Студент гр. 3312 _____ Сабиров Р. Д.

Преподаватель _____ Павловский М. Г.

Санкт-Петербург
2024

Содержание

Цель работы

Знакомство со способами подключения слушателей событий к графическим компонентам пользовательского интерфейса

Описание действий, которые должны реализовывать слушатели

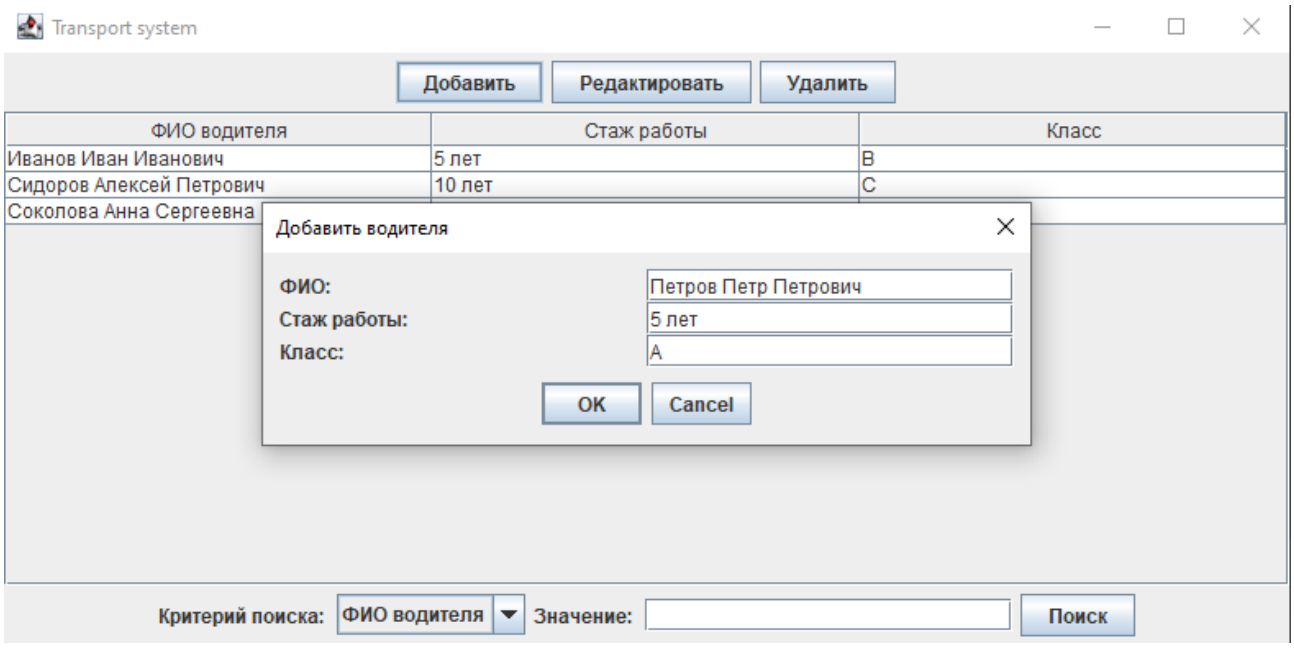
В программе реализовано приложение для управления данными о водителях, где таблица отображает информацию, а кнопки позволяют добавлять, редактировать, удалять и искать водителей. Слушатели событий выполняют определённые действия при нажатии на соответствующие кнопки:

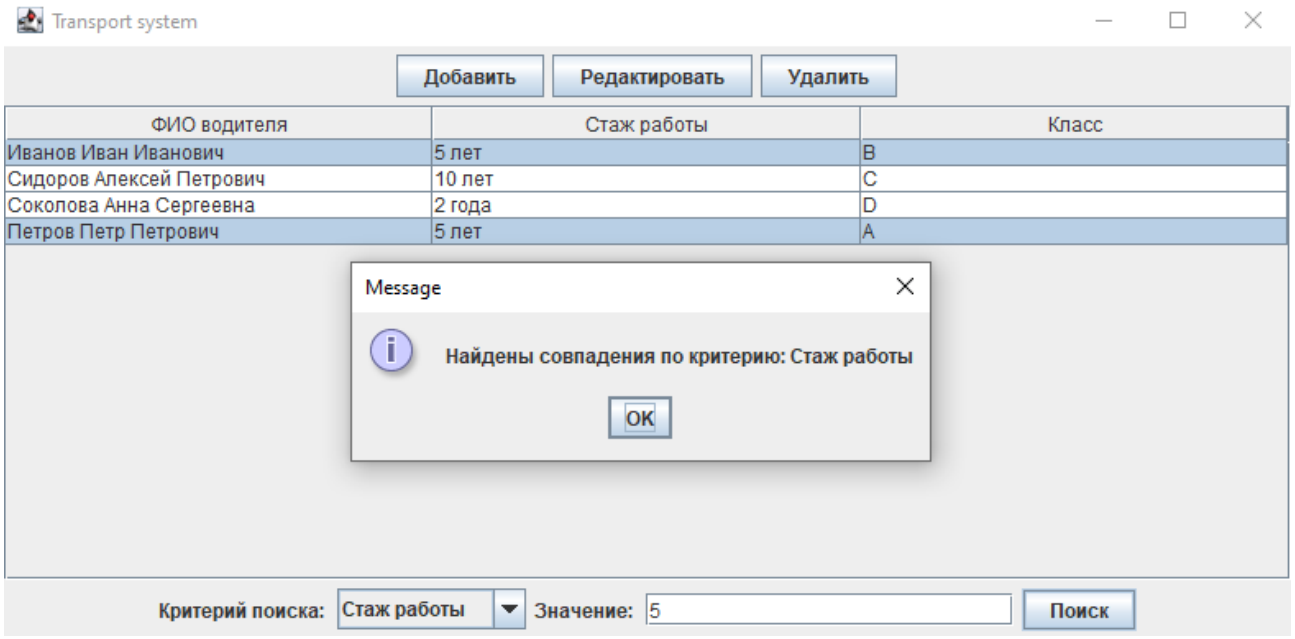
1. Слушатель для кнопки «Добавить»
    - Нажатие на кнопку открывает диалоговое окно с тремя полями для ввода данных: "ФИО водителя", "Стаж работы" и "Класс".
    - Если пользователь заполнил все поля и подтвердил ввод, новый водитель добавляется в таблицу.
    - Если одно или несколько полей не заполнены, показывается сообщение об ошибке.
    - После успешного добавления выводится сообщение "Водитель добавлен!".
2. Слушатель для кнопки «Редактировать»
    - Нажатие на кнопку редактирования открывает диалоговое окно для редактирования выбранной строки в таблице.
    - Если строка выбрана, открывается диалоговое окно с уже заполненными полями для редактирования, если нет, то выводится сообщение с просьбой о выборе строки для редактирования.
    - Пользователь может изменить значения "ФИО", "Стаж работы" и "Класс".
    - Если пользователь подтвердил изменения, обновляются данные в выбранной строке таблицы, и выводится сообщение "Данные водителя обновлены!".
3. Слушатель для кнопки «Удалить»

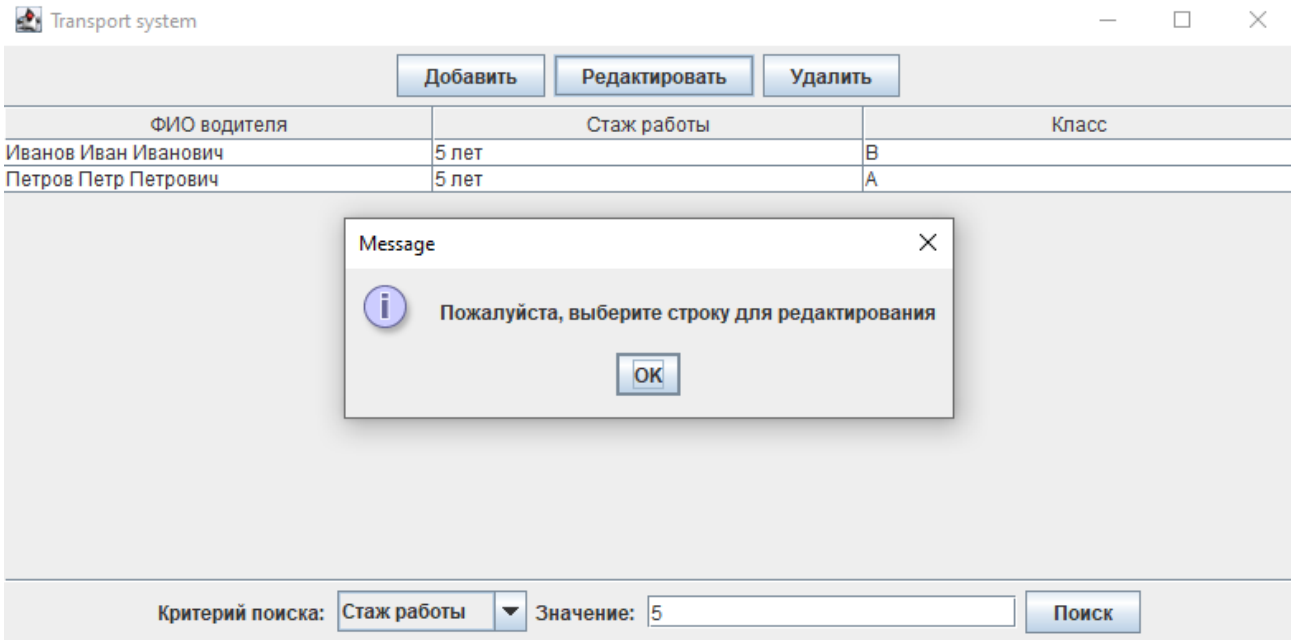
- Нажатие на кнопку удаления удаляет выбранную строку (водителя) из таблицы.
- Проверяется, выбрана ли строка в таблице. Если строка не выбрана, выводится сообщение с просьбой выбрать строку для удаления.
- Если строка выбрана, она удаляется из таблицы, и выводится сообщение "Водитель удален".

4. Слушатель для кнопки «Поиск»
   - Нажатие на кнопку поиска выполняет фильтрацию таблицы по выбранному критерию ("ФИО водителя", "Стаж работы" или "Класс").
   - Пользователь выбирает критерий поиска из выпадающего списка и вводит значение для поиска в текстовое поле.
   - Программа ищет строки, в которых значение выбранного столбца совпадает или содержит введённое значение.
   - Если найдены совпадения, соответствующие строки в таблице выделяются, и показывается сообщение о том, что совпадения найдены.
   - Если совпадений нет, выводится сообщение "Совпадений не найдено".

Скриншоты, иллюстрирующие работу слушателей




Transport system
Добавить
Редактировать
Удалить
ФИО водителя
Стаж работы
Класс
Иванов Иван Иванович
5 лет
B
Сидоров Алексей Петрович
10 лет
C
Соколова Анна Сергеевна
2 года
D
Петров Петр Петрович
5 лет
A
Message
Найдены совпадения по критерию: Стаж работы
OK
Критерий поиска:
Стаж работы
Значение:
5
Поиск


Transport system
Добавить
Редактировать
Удалить
ФИО водителя
Стаж работы
Класс
Иванов Иван Иванович
5 лет
B
Петров Петр Петрович
5 лет
A
Message
Пожалуйста, выберите строку для редактирования
OK
Критерий поиска:
Стаж работы
Значение:
5
Поиск

Текст документации, сгенерированный Javadoc

PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class lab03

java.lang.Object
lab03

public class lab03
extends Object

Version:

1.0

Author:

Сабиров Роман 3312

Constructor Summary

Constructors

Constructor	Description
lab03()	

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	Bus()	
static void	main(String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

lab03

public lab03()

Method Details

Bus

public void Bus()

main

public static void main(String[] args)

Parameters:

args - аргументы командной строки (не используются)

Исходные тексты слушателей

```
/**
 * Метод для добавления слушателей к кнопкам
 */
private void addListeners() {
    // Слушатель для кнопки "Добавить"
    addButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            addDriverDialog();
        }
    });

    // Слушатель для кнопки "Редактировать"
    editButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            int selectedRow = driverTable.getSelectedRow();
            if (selectedRow != -1) {
                editDriverDialog(selectedRow);
            } else {
                JOptionPane.showMessageDialog(frame, "Пожалуйста, выберите
строку для редактирования");
            }
        }
    });

    // Слушатель для кнопки "Удалить"
    deleteButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            int selectedRow = driverTable.getSelectedRow();
            if (selectedRow != -1) {
                tableModel.removeRow(selectedRow);
                JOptionPane.showMessageDialog(frame, "Водитель удален");
            } else {
                JOptionPane.showMessageDialog(frame, "Пожалуйста, выберите
строку для удаления");
            }
        }
    });

    searchButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String criterion = (String) searchCriteria.getSelectedItem();
            String value = searchField.getText().trim().toLowerCase();

            driverTable.clearSelection();

            boolean found = false;
            for (int i = 0; i < tableModel.getRowCount(); i++) {
                String cellValue = tableModel.getValueAt(i,
searchCriteria.getSelectedIndex()).toString().toLowerCase();

                if (cellValue.contains(value)) {
                    driverTable.addRowSelectionInterval(i, i);
                    found = true;
                }
            }
```

```
                if (found) {
                    JOptionPane.showMessageDialog(frame, "Найдены
совпадения по критерию: " + criterion);
                } else {
                    JOptionPane.showMessageDialog(frame, "Совпадений не
найдено");
                }
            }
        });
    }

    /**
     * Метод для отображения диалогового окна добавления водителя
     */
    private void addDriverDialog() {
        // Создание панелей для ввода данных
        JPanel panel = new JPanel(new GridLayout(3, 2));

        JTextField nameField = new JTextField(20);
        JTextField experienceField = new JTextField(20);
        JTextField categoryField = new JTextField(20);

        panel.add(new JLabel("ФИО: "));
        panel.add(nameField);
        panel.add(new JLabel("Стаж работы: "));
        panel.add(experienceField);
        panel.add(new JLabel("Класс: "));
        panel.add(categoryField);

        int result = JOptionPane.showConfirmDialog(frame, panel, "Добавить
водителя", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

        if (result == JOptionPane.OK_OPTION) {
            String name = nameField.getText();
            String experience = experienceField.getText();
            String category = categoryField.getText();

            if (!name.isEmpty() && !experience.isEmpty() &&
!category.isEmpty()) {
                tableModel.addRow(new Object[]{name, experience,
category});
                JOptionPane.showMessageDialog(frame, "Водитель
добавлен!");
            } else {
                JOptionPane.showMessageDialog(frame, "Все поля должны быть
заполнены!", "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    /**
     * Метод для отображения диалогового окна редактирования водителя
     * @param row номер редактируемой строки
     */
    private void editDriverDialog(int row) {
        // Создание панелей для ввода данных с уже заполненными значениями
        JPanel panel = new JPanel(new GridLayout(3, 2));
```

```
    JTextField nameField = new JTextField((String)
tableModel.getValueAt(row, 0), 20);
    JTextField experienceField = new JTextField((String)
tableModel.getValueAt(row, 1), 20);
    JTextField categoryField = new JTextField((String)
tableModel.getValueAt(row, 2), 20);

    panel.add(new JLabel("ФИО: "));
    panel.add(nameField);
    panel.add(new JLabel("Стаж работы: "));
    panel.add(experienceField);
    panel.add(new JLabel("Класс: "));
    panel.add(categoryField);

    int result = JOptionPane.showConfirmDialog(frame, panel,
"Редактировать водителя", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.PLAIN_MESSAGE);

    if (result == JOptionPane.OK_OPTION) {
        String name = nameField.getText();
        String experience = experienceField.getText();
        String category = categoryField.getText();

        if (!name.isEmpty() && !experience.isEmpty() &&
!category.isEmpty()) {
            // Обновляем данные в выбранной строке
            tableModel.setValueAt(name, row, 0);
            tableModel.setValueAt(experience, row, 1);
            tableModel.setValueAt(category, row, 2);
            JOptionPane.showMessageDialog(frame, "Данные водителя
обновлены!");
        } else {
            JOptionPane.showMessageDialog(frame, "Все поля должны быть
заполнены!", "Ошибка", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Вывод

В ходе выполнения работы были разработаны слушатели экранной формы для отображения и управления списком водителей с использованием языка программирования Java.

Ссылка на репозиторий: https://github.com/sabiroma/OOP_labs/tree/main/lab03

Ссылка на видео: https://disk.yandex.ru/i/_vkxK62rXqG3kg