**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

# по дисциплине «Введение в разработку программного обеспечения»

на тему: «Принципы и технологии создания дружественных

пользовательских интерфейсов в *Java*»

Выполнил: студент гр. ИТП-21

Рылушкин Б.А.

Принял: преподаватель-стажёр

Башаримов Ю.С.

Гомель 2024

Цель работы: улучшение пользовательского опыта за счёт разработки удобного интерфейса и реализации функционала, который упрощает взаимодействие с приложением.

**Ход работы**

**Задание.** 1. Создать новую ветку для графического приложения в ранее созданном *git* репозитории.

2. Используя библиотеку *Java Swing*, создать графический интерфейс для разработанных ранее лабораторных работ 1 и 3.

3. Элементы интерфейса должны позволять пользователю создавать, удалять и обновлять элементы коллекции, созданного в лабораторной работе 5 класса-репозитория.

4. Создать и одобрить пул-реквест на слияние ветки с интерфейсом с главной вет кой.

5. Составить отчет о проделанной работе.

Результат создания ветки *graphic\_application* с помощью команды *git checkout -b graphic\_application* представлен на рисунке 1.

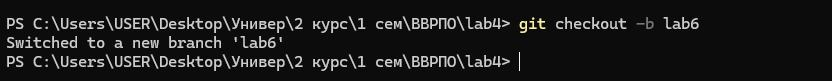


Рисунок 1 – Результат создания ветки *graphic\_application*

Результат использования библиотеки *Java Swing*, создавая графический интерфейс для разработанной ранее лабораторной работы 3 представлен на рисунке 2.

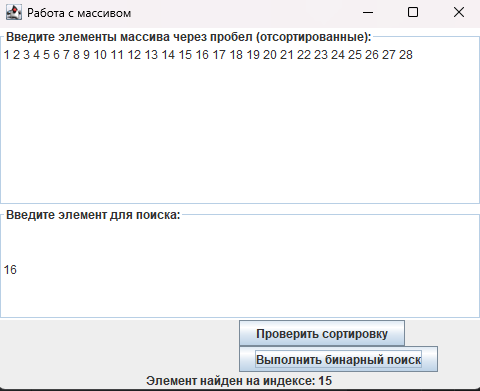


Рисунок 2 – Результат использования библиотеки *Java Swing*

для лабораторной работы 3

Результат использования библиотеки *Java Swing*, создавая графический интерфейс для разработанной ранее лабораторной работы 1 представлен на рисунке 3. Элементы интерфейса должны позволять пользователю создавать, удалять и обновлять элементы коллекции, созданного в лабораторной работе 5 класса-репозитория.

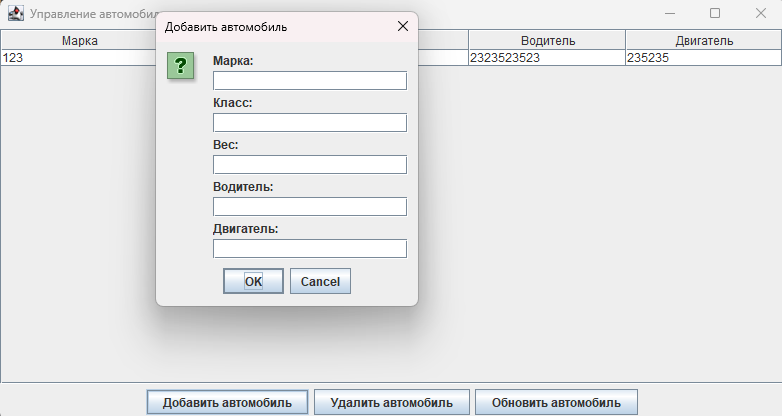
**

Рисунок 3 – Результат использования библиотеки *Java Swing*

для лабораторной работы 5

Результаты выполнения заданий использования библиотеки *Java Swing*, создать графический интерфейс для разработанных ранее лабораторных работ 1 и 3 представлены в приложении А и Б.

Результат запушивания ветки *graphic\_application* в удаленный репозиторий и её результат закоммичивания с помощью команд *git add ., git commit -m "Добавление графического приложения" и git push origin graphic\_application* представлен на рисунке 4.

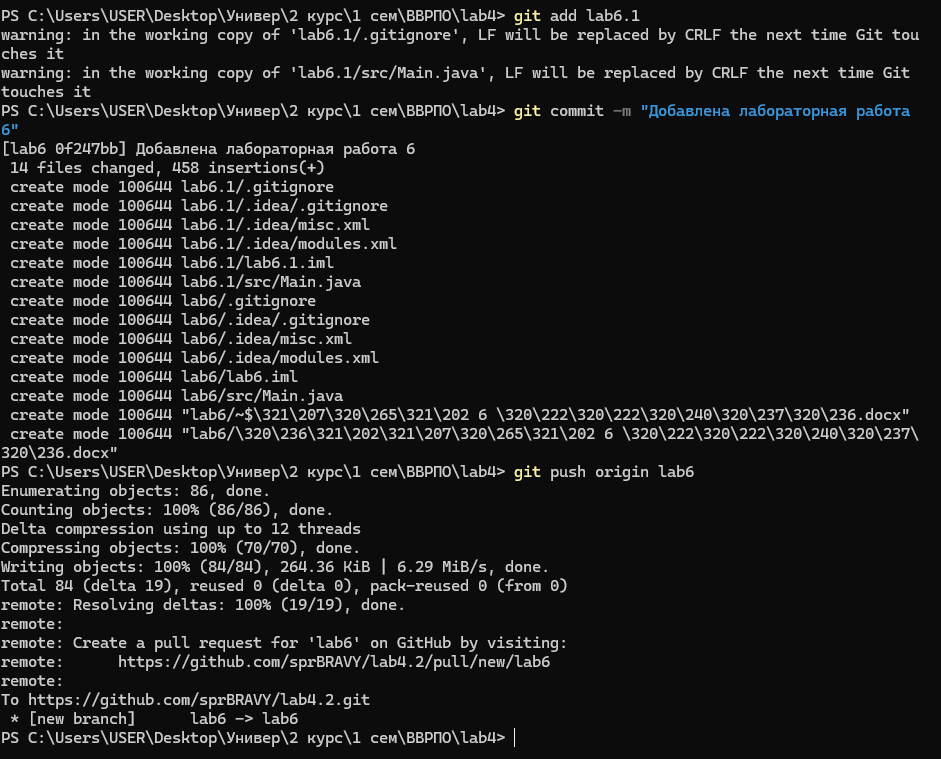


Рисунок 4 – Результат запушивания ветки *graphic\_application*

в удаленный репозиторий

Результат создания и одобрения пул-реквеста на слияние ветки *graphic\_application* с веткой *main* представлен на рисунке 5.

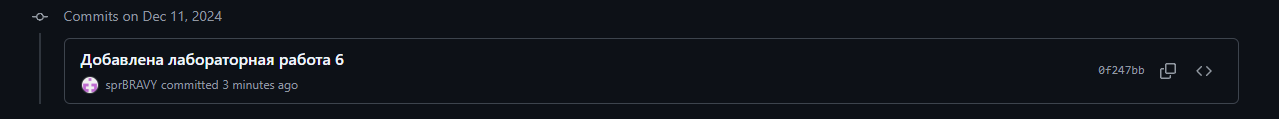


Рисунок 5 – Результат создания и одобрения пул-реквеста на слияние ветки *graphic\_application* с веткой *main*

Вывод: в ходе выполнения лабораторной работы мы улучшили пользовательский опыт за счёт разработки удобного интерфейса и реализации функционала, который упрощает взаимодействие с приложением. Современные инструменты и технологии, такие как системы контроля версий и инструменты для статического анализа, играют ключевую роль в поддержании высоких стандартов разработки. Внедрение этих практик способствует более эффективной командной работе и снижению технического долга.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

***Main.java*:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

class Car {

private String brand;

private String carClass;

private double weight;

private Driver driver;

private Engine engine;

public Car(String brand, String carClass, double weight, Driver driver, Engine engine) {

this.brand = brand;

this.carClass = carClass;

this.weight = weight;

this.driver = driver;

this.engine = engine;

}

public String getBrand() {

return brand;

}

public String getCarClass() {

return carClass;

}

public double getWeight() {

return weight;

}

public Driver getDriver() {

return driver;

}

public Engine getEngine() {

return engine;

}

}

class Driver {

private String fullName;

private int experience;

public Driver(String fullName, int experience) {

this.fullName = fullName;

this.experience = experience;

}

public String getFullName() {

return fullName;

}

public int getExperience() {

return experience;

}

}

class Engine {

private double power;

private String manufacturer;

public Engine(double power, String manufacturer) {

this.power = power;

this.manufacturer = manufacturer;

}

public double getPower() {

return power;

}

public String getManufacturer() {

return manufacturer;

}

}

public class Main{

private List<Car> cars = new ArrayList<>();

private DefaultTableModel tableModel;

public Main() {

JFrame frame = new JFrame("Управление автомобилями");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(800, 600);

// Таблица для отображения автомобилей

String[] columnNames = {"Марка", "Класс", "Вес", "Водитель", "Двигатель"};

tableModel = new DefaultTableModel(columnNames, 0);

JTable carTable = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(carTable);

// Панель для кнопок

JPanel buttonPanel = new JPanel();

JButton addButton = new JButton("Добавить автомобиль");

JButton deleteButton = new JButton("Удалить автомобиль");

JButton updateButton = new JButton("Обновить автомобиль");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

frame.setLayout(new BorderLayout());

frame.add(scrollPane, BorderLayout.CENTER);

frame.add(buttonPanel, BorderLayout.SOUTH);

// Действия кнопок

addButton.addActionListener(e -> addCar());

deleteButton.addActionListener(e -> deleteCar(carTable));

updateButton.addActionListener(e -> updateCar(carTable));

frame.setVisible(true);

}

private void addCar() {

JTextField brandField = new JTextField();

JTextField classField = new JTextField();

JTextField weightField = new JTextField();

JTextField driverField = new JTextField();

JTextField engineField = new JTextField();

Object[] fields = {

"Марка:", brandField,

"Класс:", classField,

"Вес:", weightField,

"Водитель:", driverField,

"Двигатель:", engineField

};

int option = JOptionPane.showConfirmDialog(null, fields, "Добавить автомобиль", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

try {

String brand = brandField.getText();

String carClass = classField.getText();

double weight = Double.parseDouble(weightField.getText());

String driverName = driverField.getText();

String engineDetails = engineField.getText();

Driver driver = new Driver(driverName, 0); // Упрощённые данные водителя

Engine engine = new Engine(0, engineDetails); // Упрощённые данные двигателя

Car car = new Car(brand, carClass, weight, driver, engine);

cars.add(car);

tableModel.addRow(new Object[]{brand, carClass, weight, driverName, engineDetails});

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Некорректный ввод!", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

private void deleteCar(JTable carTable) {

int selectedRow = carTable.getSelectedRow();

if (selectedRow >= 0) {

cars.remove(selectedRow);

tableModel.removeRow(selectedRow);

} else {

JOptionPane.showMessageDialog(null, "Выберите автомобиль для удаления.", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateCar(JTable carTable) {

int selectedRow = carTable.getSelectedRow();

if (selectedRow >= 0) {

Car car = cars.get(selectedRow);

JTextField brandField = new JTextField(car.getBrand());

JTextField classField = new JTextField(car.getCarClass());

JTextField weightField = new JTextField(String.valueOf(car.getWeight()));

JTextField driverField = new JTextField(car.getDriver().getFullName());

JTextField engineField = new JTextField(car.getEngine().getManufacturer());

Object[] fields = {

"Марка:", brandField,

"Класс:", classField,

"Вес:", weightField,

"Водитель:", driverField,

"Двигатель:", engineField

};

int option = JOptionPane.showConfirmDialog(null, fields, "Обновить автомобиль", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

try {

car = new Car(

brandField.getText(),

classField.getText(),

Double.parseDouble(weightField.getText()),

new Driver(driverField.getText(), 0),

new Engine(0, engineField.getText())

);

cars.set(selectedRow, car);

tableModel.setValueAt(car.getBrand(), selectedRow, 0);

tableModel.setValueAt(car.getCarClass(), selectedRow, 1);

tableModel.setValueAt(car.getWeight(), selectedRow, 2);

tableModel.setValueAt(car.getDriver().getFullName(), selectedRow, 3);

tableModel.setValueAt(car.getEngine().getManufacturer(), selectedRow, 4);

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Некорректный ввод!", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

} else {

JOptionPane.showMessageDialog(null, "Выберите автомобиль для обновления.", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(Main::new);

}

}  
  
***Main.Java:****import javax.swing.\*;*

*import java.awt.\*;*

*import java.util.Arrays;*

*public class Main {*

*private JTextArea arrayInputField;*

*private JTextField targetField;*

*private JLabel resultLabel;*

*public Main() {*

*JFrame frame = new JFrame("Работа с массивом");*

*frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

*frame.setSize(500, 400);*

*JPanel panel = new JPanel();*

*panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));*

*// Поле для ввода массива*

*arrayInputField = new JTextArea(5, 40);*

*arrayInputField.setBorder(BorderFactory.createTitledBorder("Введите элементы массива через пробел (отсортированные):"));*

*// Поле для ввода целевого значения*

*targetField = new JTextField();*

*targetField.setBorder(BorderFactory.createTitledBorder("Введите элемент для поиска:"));*

*// Кнопки*

*JButton checkSortButton = new JButton("Проверить сортировку");*

*JButton searchButton = new JButton("Выполнить бинарный поиск");*

*// Результат*

*resultLabel = new JLabel("Результат будет отображён здесь");*

*resultLabel.setAlignmentX(Component.CENTER\_ALIGNMENT);*

*// Добавляем элементы в панель*

*panel.add(arrayInputField);*

*panel.add(targetField);*

*panel.add(checkSortButton);*

*panel.add(searchButton);*

*panel.add(resultLabel);*

*frame.add(panel);*

*// Действия кнопок*

*checkSortButton.addActionListener(e -> checkSort());*

*searchButton.addActionListener(e -> performSearch());*

*frame.setVisible(true);*

*}*

*private void checkSort() {*

*try {*

*int[] array = parseArray(arrayInputField.getText());*

*if (ArrayUtils.isSorted(array)) {*

*resultLabel.setText("Массив отсортирован.");*

*} else {*

*resultLabel.setText("Массив не отсортирован.");*

*}*

*} catch (NumberFormatException ex) {*

*resultLabel.setText("Ошибка: некорректный ввод массива.");*

*}*

*}*

*private void performSearch() {*

*try {*

*int[] array = parseArray(arrayInputField.getText());*

*if (!ArrayUtils.isSorted(array)) {*

*resultLabel.setText("Ошибка: массив не отсортирован.");*

*return;*

*}*

*int target = Integer.parseInt(targetField.getText());*

*BinarySearch binarySearch = new BinarySearch();*

*int result = binarySearch.binarySearch(array, target);*

*if (result == -1) {*

*resultLabel.setText("Элемент не найден.");*

*} else {*

*resultLabel.setText("Элемент найден на индексе: " + result);*

*}*

*} catch (NumberFormatException ex) {*

*resultLabel.setText("Ошибка: некорректный ввод массива или элемента.");*

*}*

*}*

*private int[] parseArray(String text) throws NumberFormatException {*

*return Arrays.stream(text.trim().split("\\s+")).mapToInt(Integer::parseInt).toArray();*

*}*

*public static void main(String[] args) {*

*SwingUtilities.invokeLater(Main::new);*

*}*

*}*

*class ArrayUtils {*

*public static boolean isSorted(int[] array) {*

*for (int i = 1; i < array.length; i++) {*

*if (array[i] < array[i - 1]) {*

*return false;*

*}*

*}*

*return true;*

*}*

*}*

*class BinarySearch {*

*public int binarySearch(int[] array, int target) {*

*int left = 0;*

*int right = array.length - 1;*

*while (left <= right) {*

*int mid = left + (right - left) / 2;*

*if (array[mid] == target) {*

*return mid;*

*}*

*if (array[mid] < target) {*

*left = mid + 1;*

*} else {*

*right = mid - 1;*

*}*

*}*

*return -1;*

*}*

*}*