МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Сирота Марина Романовна

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №4

по дисциплине: «Платформа Java»

по теме: «Исследование возможностей разработки приложений с использованием платформы JAVAFX 2»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. пр. Кузнецов С.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2018

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.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Необходимо создать Java приложение с графическим интерфейсом пользователя, реализующее добавление, редактирование, сортировку и удаление данных заданного по варианту типа информации T(см. табл. 4.1). 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.

При написании программы следует учесть следующие требования и рекомендации:

1. Создать публичный класс, представляющий заданный по варианту задания (см. табл. 4.1) тип информации (т.е. строку таблицы).

2. Создать модель данных таблицы. Для этого создать класс, расширяющий абстрактный класс AbstractTableModel. Создать в нем объект коллекции типа T, соответствующий варианту задания. Переопределить методы:

* public Class
* getColumnClass(int columnIndex)
* public int getColumnCount()
* public String getColumnName(int columnIndex)
* public Object getValueAt(int rowIndex, int columnIndex)
* public boolean isCellEditable(int rowIndex, int columnIndex)
* public void setValueAt(Object value, int rowIndex, int columnIndex)

Определить методы:

* public void addRow() – добавления элемента (строки)
* public void deleteRow(String Поле\_1) – удаления элемента по значению поля 1
* public void updateRow(int row ) – изменения элемента заданной строки)

3. Для реализации окна приложения реализовать дочерний класс JFrame.

4. Для представления таблицы с данными использовать компонент класса JTable, разместив его в контейнере JScrollPane (для возможности добавления полос прокрутки).

5. Поля ввода для добавления и редактирования данных реализовать текстовыми компонентами JTextField. Каждое поле снабдить подписью при помощи компонентов JLabel.

6. Для выполнения действий открытия файла, добавления, изменения, удаления записи, сортировки и сохранения файла реализовать соответствующие кнопки, с использованием компонентов JButton и добавлением ActionListener. Реализовать загрузку записи в поля для редактирования при щелчке по строке таблицы.

7. Для выбора файла при открытии и сохранении использовать компонент JFileChooser.

8. Удаление и сортировка элементов должно проходить по ключевому полю P (см. табл. 4.1). Направление сортировки U(см. табл. 4.1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | B | 2 | Возрастание | LinkedList |

B: Ноутбук (Идентификатор модели, Производитель процессора, Тактовая частота процессора, Объем ОЗУ);

1. ВЫПОЛНЕНИЕ РАБОТЫ
   1. Листинг

PC.java

**public** **class** PC **implements** Comparable<PC> {

**protected** **int** id;

**protected** String makerCPU;

**protected** **int** frequency;

**protected** **int** RAM;

**public** PC(**int** id\_, String makerCPU\_, **int** frequency\_, **int** RAM\_) {

id=id\_;

makerCPU=makerCPU\_;

frequency=frequency\_;

RAM=RAM\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getMakerCPU() {

**return** makerCPU;

}

**public** **int** getFrequency() {

**return** frequency;

}

**public** **int** getRAM() {

**return** RAM;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setMakerCPU(String makerCPU) {

**this**.makerCPU = makerCPU;

}

**public** **void** setFrequency(**int** frequency) {

**this**.frequency = frequency;

}

**public** **void** setRAM(**int** RAM) {

**this**.RAM = RAM;

}

@Override

**public** **int** compareTo(PC pc) {

**return** **this**.makerCPU.compareTo(pc.makerCPU);

}

}

PCTableModel.java

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** javax.swing.table.AbstractTableModel;

**public** **class** PCTableModel **extends** AbstractTableModel {

//идентификатор для сериализации

**private** **static** **final** **long** ***serialVersionUID*** = 7927259757559420606L;

**private** LinkedList<PC> PCT\_obj; // хранилище данных

**public** PCTableModel(LinkedList<PC> PCT\_obj) {// конструктор модели

**this**.PCT\_obj = PCT\_obj;

}

//обязательный метод - возвращает тип значения столбца

**public** Class<?> getColumnClass(**int** columnIndex) {

**return** getValueAt(0, columnIndex).getClass();

}

**public** **int** getColumnCount() {

**return** 4;

}

//обязательный метод - возвращает имя столбца

**public** String getColumnName(**int** columnIndex) {

**switch** (columnIndex) {

**case** 0:

**return** "ID";

**case** 1:

**return** "Произв процессора";

**case** 2:

**return** "Тактовая частота";

**case** 3:

**return** "Обьем ОЗУ";

}

**return** "";

}

**public** **int** getRowCount() {// обязательный метод - возвращает количество строк

**return** PCT\_obj.size();

}

//обязательный метод - возвращает значение поля таблицы

**public** Object getValueAt(**int** rowIndex, **int** columnIndex) {

PC p = PCT\_obj.get(rowIndex);

**switch** (columnIndex) {

**case** 0:

**return** p.getID();

**case** 1:

**return** p.getMakerCPU();

**case** 2:

**return** p.getFrequency();

**case** 3:

**return** p.getRAM();

}

**return** "";

}

// определяет возможно ли редактирование таблицы

**public** **boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {

**return** **false**;

}

//выполняет изменение данных в ячейке

**public** **void** setValueAt(Object value, **int** rowIndex, **int** columnIndex) {

}

**public** **void** addRow(PC nr) { // добавление строки в таблицу

PCT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

**int** index = PCT\_obj.indexOf(nr);

myJframe.*table*.setRowSelectionInterval(index, index);

}

**public** **void** deleteRow(String makerCPU) { // удаление строки из таблицы

Iterator<PC> isph = PCT\_obj.iterator();

**boolean** flag = **false**;

**while** (isph.hasNext()) {

**if** (isph.next().getMakerCPU().equals(makerCPU)) {

isph.remove();

flag = **true**;

**break**;

}

}

**if** (flag) {

fireTableDataChanged();// вызываем для обновления таблицы

} **else**

// JOPCT\_objionPane.showMessageDialog(myJframe.this,"Запись "+Model+" ненайдена!")

;

}

**public** **void** updateRow(**int** index, PC nr) {// обновление строки в таблице

PCT\_obj.set(index, nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем измененную строку - иначе после обновления таблицы теряет выделение

myJframe.*table*.setRowSelectionInterval(index, index);

}

**public** **void** Sort() { // сортировка

Collections.*sort*(PCT\_obj);

fireTableDataChanged(); // вызываем для обновления таблицы

}

}

myJframe.java

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import java.awt.Color;

import javax.swing.JTextField;

import javax.swing.JLabel;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.util.LinkedList;

import java.util.Scanner;

import java.awt.event.ActionEvent;

import javax.swing.JOptionPane;

import javax.swing.JMenuBar;

import javax.swing.JMenu;

import javax.swing.JMenuItem;

public class myJframe extends JFrame {

private JPanel contentPane;

static JTable table;

private JTextField textID;

private JTextField textMakerCPU;

private JTextField textFrequency;

private JTextField textRAM;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

myJframe frame = new myJframe();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public myJframe() {

setTitle("ПК");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 500, 512);

setBackground(new Color(232, 202, 202));

JMenuBar menuBar = new JMenuBar();

menuBar.setToolTipText("");

setJMenuBar(menuBar);

//menuBar.setBackground(new Color(232, 202, 202));

JMenu menu = new JMenu("Файл");

menuBar.add(menu);

JMenuItem mntmNewMenuItem = new JMenuItem("Открыть файл");

menu.add(mntmNewMenuItem);

JMenuItem mntmNewMenuItem\_1 = new JMenuItem("Сохранить в файл");

menu.add(mntmNewMenuItem\_1);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

contentPane.setBackground(new Color(232, 202, 202));

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 11, 400, 250);

contentPane.add(scrollPane);

scrollPane.setBackground(new Color(232, 202, 202));

final LinkedList<PC> tbl = new LinkedList<PC>();

final PCTableModel PCT\_obj = new PCTableModel(tbl);

table = new JTable(PCT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel label = new JLabel("ID");

label.setBounds(10, 282, 130, 23);

contentPane.add(label);

JLabel label\_1 = new JLabel("Произв процессора");

label\_1.setBounds(10, 313, 130, 23);

contentPane.add(label\_1);

JLabel label\_2 = new JLabel("Тактовая частота");

label\_2.setBounds(10, 344, 130, 23);

contentPane.add(label\_2);

JLabel label\_3 = new JLabel("Объем ОЗУ");

label\_3.setBounds(10, 375, 130, 23);

contentPane.add(label\_3);

textID = new JTextField();

textID.setBounds(160, 282, 140, 23);

contentPane.add(textID);

textID.setColumns(10);

textMakerCPU = new JTextField();

textMakerCPU.setColumns(10);

textMakerCPU.setBounds(160, 313, 140, 23);

contentPane.add(textMakerCPU);

textFrequency = new JTextField();

textFrequency.setColumns(10);

textFrequency.setBounds(160, 344, 140, 23);

contentPane.add(textFrequency);

textRAM = new JTextField();

textRAM.setColumns(10);

textRAM.setBounds(160, 375, 140, 23);

contentPane.add(textRAM);

textRAM.getDocument().addDocumentListener(new DocumentListener() {

@Override

public void changedUpdate(DocumentEvent arg0) {

}

@Override

public void insertUpdate(DocumentEvent arg0) {

// TODO Auto-generated method stub

}

@Override

public void removeUpdate(DocumentEvent arg0) {

// TODO Auto-generated method stub

}

});

JButton addbtn = new JButton("Добавить");

addbtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

PCT\_obj.addRow(new PC(Integer.parseInt(textID.getText()), textMakerCPU.getText(),

Integer.parseInt(textFrequency.getText()), Integer.parseInt(textRAM.getText())));

}

});

addbtn.setBounds(310, 282, 95, 23);

contentPane.add(addbtn);

JButton delbtn = new JButton("Удалить");

delbtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

PCT\_obj.deleteRow(textMakerCPU.getText());

}

});

delbtn.setBounds(310, 344, 95, 23);

contentPane.add(delbtn);

JButton sortbtn = new JButton("Сортировать");

sortbtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

PCT\_obj.Sort();

}

});

sortbtn.setBounds(310, 375, 95, 23);

contentPane.add(sortbtn);

JButton btnUpdt = new JButton("Изменить");

//btnUpdt.setForeground(new Color(186, 108, 156).brighter());

//btnUpdt.setBackground(new Color(232, 202, 202));

btnUpdt.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0)

PCT\_obj.updateRow(table.getSelectedRow(),

new PC(Integer.parseInt(textID.getText()), textMakerCPU.getText(),

Integer.parseInt(textFrequency.getText()), Integer.parseInt(textRAM.getText())));

else

JOptionPane.showMessageDialog(myJframe.this, "Не выбрана ни одна строка таблицы");

}

});

//btnUpdt.setForeground(new Color(186, 108, 156));

btnUpdt.setBounds(310, 313, 95, 23);

contentPane.add(btnUpdt);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textID.setText(table.getValueAt(row, 0).toString());

textMakerCPU.setText(table.getValueAt(row, 1).toString());

textFrequency.setText(table.getValueAt(row, 2).toString());

textRAM.setText(table.getValueAt(row, 3).toString());

// сделать кнопку обновления недоступной(пока нет изменений)

btnUpdt.setEnabled(false);

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

textRAM.getDocument().addDocumentListener(new DocumentListener() {

public void changedUpdate(DocumentEvent arg0) {

btnUpdt.setEnabled(true);

System.out.println("Was changed...");

}

public void insertUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

public void removeUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

});

textID.getDocument().addDocumentListener(new DocumentListener() {

public void changedUpdate(DocumentEvent arg0) {

btnUpdt.setEnabled(true);

System.out.println("Was changed...");

}

public void insertUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

public void removeUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

});

textFrequency.getDocument().addDocumentListener(new DocumentListener() {

public void changedUpdate(DocumentEvent arg0) {

btnUpdt.setEnabled(true);

System.out.println("Was changed...");

}

public void insertUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

public void removeUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

});

textMakerCPU.getDocument().addDocumentListener(new DocumentListener() {

public void changedUpdate(DocumentEvent arg0) {

btnUpdt.setEnabled(true);

System.out.println("Was changed...");

}

public void insertUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

public void removeUpdate(DocumentEvent e) {

btnUpdt.setEnabled(true);

}

});

mntmNewMenuItem.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = new JFileChooser();

int returnVal = chooser.showOpenDialog(getParent());

if (returnVal == JFileChooser.APPROVE\_OPTION) {

System.out.println("Открытие файла - : " + chooser.getSelectedFile().getName());

Scanner in = null;

try {

in = new Scanner(new File(chooser.getSelectedFile().getAbsolutePath()));

in.useDelimiter(";");

} catch (FileNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

while (in.hasNext()) {

PC t = new PC(in.nextInt(),in.next(), in.nextInt(), in.nextInt());

PCT\_obj.addRow(t);

}

}

}

});

mntmNewMenuItem\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = new JFileChooser();

int returnVal = chooser.showSaveDialog(getParent());

if (returnVal == JFileChooser.APPROVE\_OPTION) {

System.out.println("Сохранение в файл - : " + chooser.getSelectedFile().getName());

PrintWriter out = null;

try {

out = new PrintWriter(chooser.getSelectedFile().getAbsolutePath());

} catch (FileNotFoundException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

PC[] myArray = {};

myArray = tbl.toArray(new PC[tbl.size()]);

for (int i = 0; i < myArray.length; i++) {

out.print(myArray[i].id + ";" + myArray[i].makerCPU + ";" + myArray[i].frequency + ";"

+ myArray[i].RAM + ";");

}

out.close();

}

}

});

}

}

* 1. Результаты выполнения

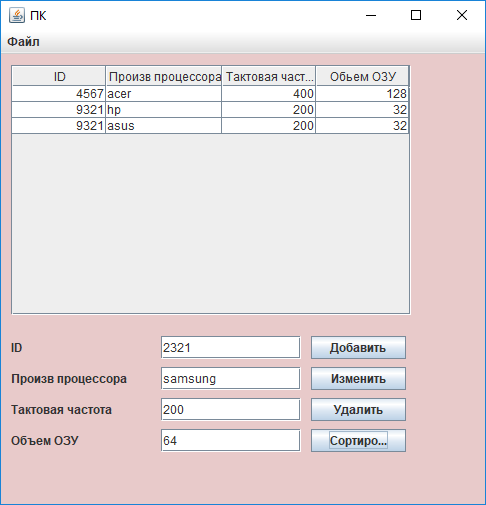


Рисунок 3.1 – Результаты работы программы

ВЫВОДЫ

В ходе четвертой лабораторной работы по дисциплине «Платформа Java» были изучены особенностями инструментария библиотеки SWING для создания графического интерфейса приложений на языке Java.

Были приобретены практические навыки создания Java-программ с графическим интерфейсом, позволяющим пользователю осуществлять взаимодействие с программой: задавать исходные данные, просматривать результаты работы программы в удобном виде.

Была написана программа в соответствии с вариантом задания.