Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №4

по дисциплине: «Платформа Java»

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Кузнецов С.А.

г. Севастополь

2016 г.

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. Вариант задания

Необходимо создать Java приложение с графическим интерфейсом пользователя, реализующее добавление, редактирование, сортировку и удаление данных заданного по варианту типа информации T (см. табл. 2.1). 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. 2.1) 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(<объект>) – добавления элемента (строки)

• public void deleteRow(String Поле\_1) – удаления элемента по значению поля 1

• public void updateRow(int row, <объект>) – 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. 2.1). Направление сортировки U (см. табл. 2.1).

Таблица 2.1 Варианты заданий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип информации (см. ниже) | Поле для сортировки (P) | Направление (U) | Тип коллекции (Т) |
| 2 | B | 2 | Возрастание | HashMap |

Тип информации:

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

1. Код программы

Класс Main:

public class Main extends JFrame {

private static final long serialVersionUID = 1L;

public static HashMap<Integer,Notebook> ListNb = new HashMap<Integer,Notebook>();

private JTable table;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Main frame = new Main();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

JButton button\_5 = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

//создаем стандартную модель

DefaultTableModel dtm = new DefaultTableModel();

/\*\*

\* Create the frame.

\*/

public Main() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 549, 394);

getContentPane().setLayout(null);

JScrollPane scrollPane = new JScrollPane();

scrollPane.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

}

});

scrollPane.setBounds(118, 66, 405, 278);

getContentPane().add(scrollPane);

//задаем названия столбцов

dtm.setColumnIdentifiers(new String[] {"ID", "Производитель", "Частота","RAM"});

//наполняем модель данными

table = new JTable(dtm);

scrollPane.setViewportView(table);

TextField textField = new TextField();

textField.setBounds(118, 38, 62, 22);

getContentPane().add(textField);

Label label = new Label("ID");

label.setFont(new Font("Dialog", Font.PLAIN, 14));

label.setBounds(115, 10, 62, 22);

getContentPane().add(label);

Label label\_1 = new Label("\u041F\u0440\u043E\u0438\u0437\u0432\u043E\u0434\u0438\u0442\u0435\u043B\u044C");

label\_1.setFont(new Font("Dialog", Font.PLAIN, 14));

label\_1.setBounds(205, 10, 110, 22);

getContentPane().add(label\_1);

Label label\_2 = new Label("\u0427\u0430\u0441\u0442\u043E\u0442\u0430 CPU");

label\_2.setFont(new Font("Dialog", Font.PLAIN, 14));

label\_2.setBounds(332, 10, 89, 22);

getContentPane().add(label\_2);

Label label\_3 = new Label("RAM");

label\_3.setFont(new Font("Dialog", Font.PLAIN, 14));

label\_3.setBounds(440, 10, 62, 22);

getContentPane().add(label\_3);

TextField textField\_1 = new TextField();

textField\_1.setBounds(204, 38, 111, 22);

getContentPane().add(textField\_1);

TextField textField\_2 = new TextField();

textField\_2.setBounds(332, 38, 89, 22);

getContentPane().add(textField\_2);

TextField textField\_3 = new TextField();

textField\_3.setBounds(440, 38, 83, 22);

getContentPane().add(textField\_3);

JButton btnNewButton = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

//добавление

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(textField.getText().length()!=0){

Integer i1 = new Integer(textField.getText());

Double d3 = new Double(textField\_2.getText());

Double d4 = new Double(textField\_3.getText());

Notebook nb = new Notebook(i1, textField\_1.getText(), d3, d4);

if(ListNb.containsKey(i1))return;

ListNb.put(i1, nb);

//заполнение таблицы

dtm.addRow(new String[]{textField.getText(),

textField\_1.getText(),

textField\_2.getText(),

textField\_3.getText()});

}

}

});

btnNewButton.setBounds(10, 10, 96, 23);

getContentPane().add(btnNewButton);

//очистка таблицы

Button button\_1 = new Button("\u0423\u0434\u0430\u043B\u0438\u0442\u044C \u0432\u0441\u0435");

button\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

while(table.getRowCount()>0){

dtm.removeRow(0);

}

ListNb.clear();

}

});

button\_1.setBounds(10, 122, 89, 22);

getContentPane().add(button\_1);

TextField textField\_4 = new TextField();

textField\_4.setBounds(10, 66, 89, 22);

getContentPane().add(textField\_4);

Label label\_4 = new Label("\u0418\u0437\u0433\u043E\u0442\u043E\u0432\u0438\u0442\u0435\u043B\u044C");

label\_4.setFont(new Font("Dialog", Font.PLAIN, 12));

label\_4.setBounds(10, 50, 96, 22);

getContentPane().add(label\_4);

//удаление строки

Button button = new Button("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(textField\_4.getText().length()==0) return;

for(int i = 0; i < table.getRowCount(); i++){

if(dtm.getValueAt(i, 1).equals(textField\_4.getText())){

Integer in = new Integer(""+dtm.getValueAt(i, 0));

ListNb.remove(in);

dtm.removeRow(i);

}

}

}

});

button.setBounds(10, 94, 89, 22);

getContentPane().add(button);

JButton button\_2 = new JButton("\u041E\u0442\u043A\u0440\u044B\u0442\u044C");

button\_2.addActionListener(new ActionListener() {

//загрузить

public void actionPerformed(ActionEvent e) {

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

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

File file = fileopen.getSelectedFile();

try {

FileRead(file.getAbsolutePath());

} catch (IOException e1) {

}

}

}

});

button\_2.setBounds(10, 291, 89, 23);

getContentPane().add(button\_2);

JButton button\_3 = new JButton("\u0421\u043E\u0445\u0440\u0430\u043D\u0438\u0442\u044C");

button\_3.addActionListener(new ActionListener() {

//сохранить

public void actionPerformed(ActionEvent e) {

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Сохранить в файл");

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

File file = fileopen.getSelectedFile();

try {

FileWrite(file.getAbsolutePath());

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

}

});

button\_3.setBounds(10, 321, 89, 23);

getContentPane().add(button\_3);

//СОРТИРОВКА

JButton button\_4 = new JButton("\u0421\u043E\u0440\u0442\u0438\u0440\u043E\u0432\u0430\u0442\u044C(2)");

button\_4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

while(table.getRowCount()>0){

dtm.removeRow(0);

}

LinkedList<Notebook> ln = new LinkedList<Notebook>();

for(Entry<Integer, Notebook> n: ListNb.entrySet()){

ln.add(n.getValue());

}

Collections.sort(ln, new Sort());

for(Notebook n: ln){

dtm.addRow(new String[]{""+n.ID,

n.CPU\_Maker,

""+n.CPU\_Freq,

""+n.RAM});

}

}

});

button\_4.setBounds(0, 170, 113, 23);

getContentPane().add(button\_4);

button\_5.addMouseListener(new MouseAdapter() {

@Override

//клик

public void mouseClicked(MouseEvent e) {

}

});

//изменить

button\_5.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

for(Entry<Integer, Notebook> n: ListNb.entrySet()){

for(int i=0; i<table.getRowCount(); i++){

//JOptionPane.showMessageDialog(table, ""+table.getValueAt(i,1));

if(table.getValueAt(i, 0).equals("" + n.getValue().ID)){

Double d1 = new Double("" + table.getValueAt(i,2));

Double d2 = new Double("" + table.getValueAt(i,3));

n.getValue().CPU\_Maker = "" + table.getValueAt(i,1);

n.getValue().CPU\_Freq = d1;

n.getValue().RAM = d2;

}

}

}

}

});

button\_5.setBounds(10, 229, 89, 23);

getContentPane().add(button\_5);

}

public void FileWrite(String Filename)throws IOException{

FileWriter f = new FileWriter(Filename);

for(Entry<Integer, Notebook> n : ListNb.entrySet()){

f.write(""+n.getValue().ID + "\r\n" +

n.getValue().CPU\_Maker + "\r\n" +

n.getValue().CPU\_Freq + "\r\n" +

n.getValue().RAM + "\r\n\r\n");

}

f.close();

}

public void FileRead(String Filename)throws IOException{

while(table.getRowCount()>0){

dtm.removeRow(0);

}

ListNb.clear();

Scanner fin = new Scanner(new File(Filename));

String author;

int ID;

double RAM, CPU;

//очитска таблицы

while(fin.hasNext())

{

if(ListNb.size()>0){

fin.nextLine();

}

ID = Integer.valueOf(fin.nextLine());

author = fin.nextLine();

CPU = Double.valueOf(fin.nextLine());

RAM = Double.valueOf(fin.nextLine());

ListNb.put(ID, new Notebook(ID,author,CPU,RAM));

dtm.addRow(new String[]{""+ID,

author,

"" + CPU,

"" + RAM});

}

JOptionPane.showMessageDialog(table, "Файл открыт!");

fin.close();

}

}

Класс Notebook:

public class Notebook {

//Идентификатор модели

public int ID;

//Производитель процессора

public String CPU\_Maker;

//Тактовая частота процессора

public double CPU\_Freq;

//Объем ОЗУ

public double RAM;

public Notebook(int \_ID, String \_CPU\_Maker, double \_CPU\_Freq, double \_RAM){

ID = \_ID;

CPU\_Maker = \_CPU\_Maker;

CPU\_Freq = \_CPU\_Freq;

RAM = \_RAM;

}

}

Класс NotebookContainer:

public abstract class NotebookContainer extends AbstractTableModel{

private static final long serialVersionUID = 1L;

@Override

public int getRowCount() {

return Main.ListNb.size();

}

@Override

public int getColumnCount() {

return 4;

}

public void keyTyped(KeyEvent e) {

Main m = new Main();

m.button\_5.setVisible(true);

}

public void keyPressed(KeyEvent e) {

// TODO Auto-generated method stub

}

public void keyReleased(KeyEvent e) {

// TODO Auto-generated method stub

}

}

Класс Sort:

public class Sort implements Comparator<Notebook>{

public int compare(Notebook obj1, Notebook obj2) {

return obj1.CPU\_Maker.compareTo(obj2.CPU\_Maker);

}

}

1. Выполнение программы

При запуске программы, будет открыто следующие окно, которое представляет следующие возможности:

* Добавление записи о ноутбуке
* Просмотр любого количества записей
* Загрузка данных с текстового файла
* Запись данных в текстовый файл
* Удаление по полю «Изготовитель»
* Очистка таблицы
* Изменение данных в самой таблице
* Сортировка списка по полю «Изготовитель»

При работе с данным оконным приложением будет выводится следующее окно (рис.4.1). В данной программе поле ID является уникальным и не должно повторяться.

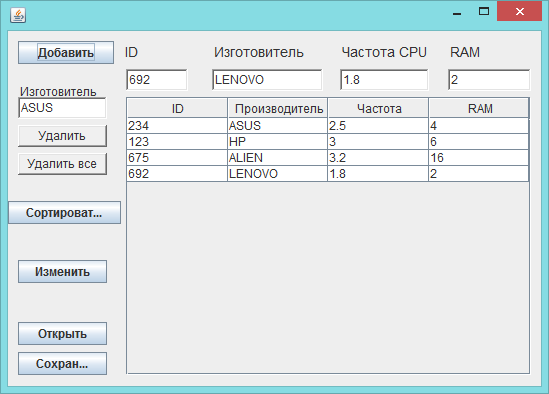


Рисунок 4.1 – Работы программы.

Добавление осуществляется при обязательном поле номер 1 и при условии, что нет элементов с таким значением. Удаление элемента производится при ввод в поле Изготовитель и нажатии кнопки удаление.

При нажатии кнопки Сортировать произведётся сортировка по второму полю записи по возрастанию.

Чтобы сохранить в файл, необходимо нажать кнопку сохранить и в возникшем окне выбрать файл или указать имя нового. (рис.4.2)

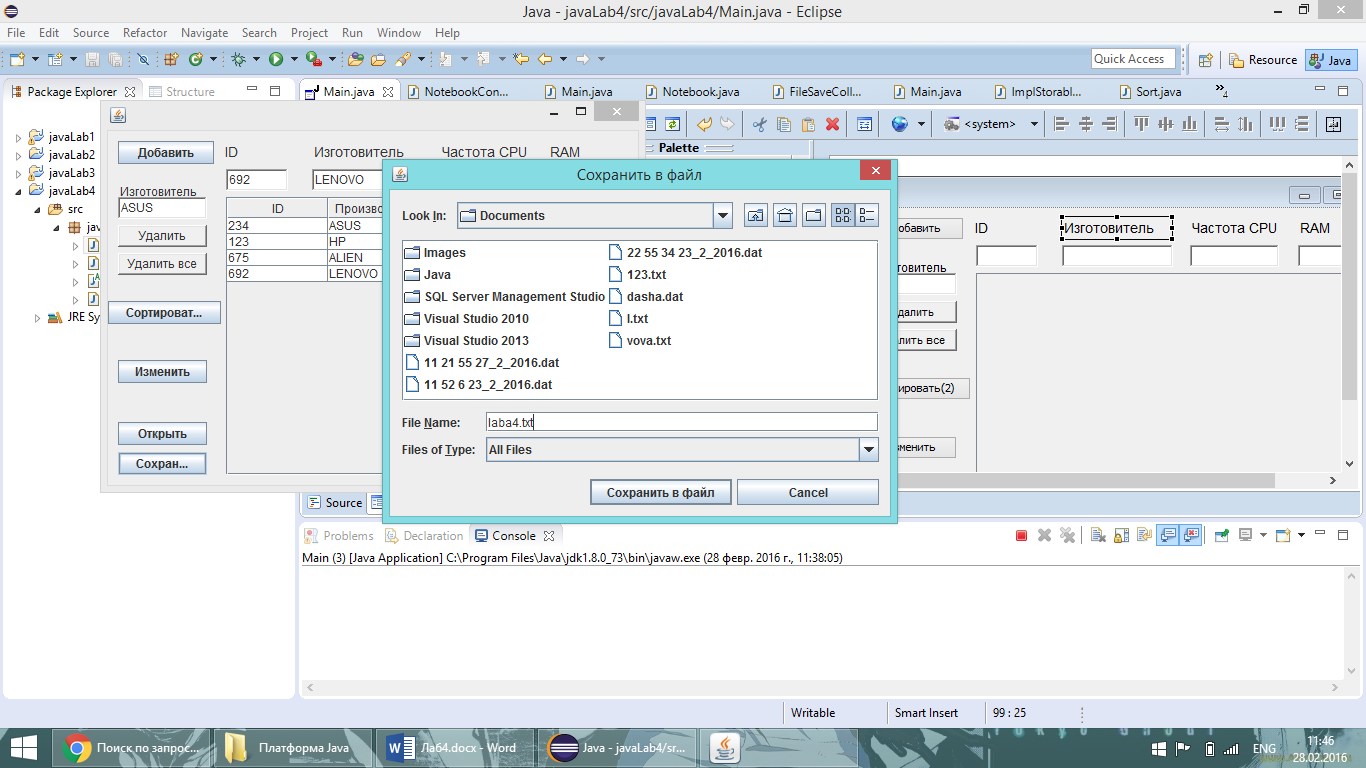


Рисунок 4.2 – Сохранение в файл.

Чтобы загрузить данные с файла необходимо в аналогичном окне выбрать необходимый файл.

Для того чтобы Изменить значение записи необходимо двойным кликом мыши на ячейку строку выбрать редактируемое поле и ввести своё значение, с учётом типа данного поля. Уникальное значение поля 1 менять невозможно. После необходимо нажать кнопку Изменить на панели слева, чтобы сохранить внесённую правку.

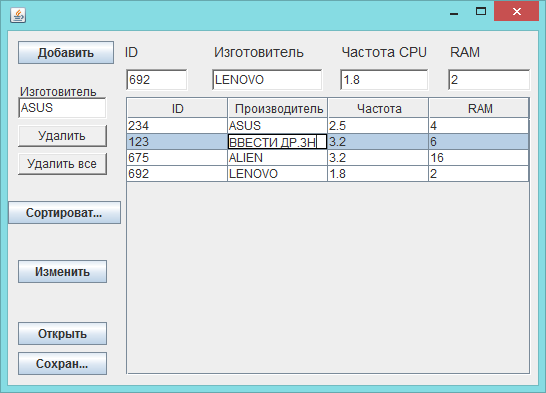


Рисунок 4.3 – Изменение записи.

ВЫВОДЫ

В данной лабораторной работе была изучена графическая библиотека SWING в среде Eclipse, с использованием графического менеджера. Были переопределены методы стандартных коллекций и библиотек. Были использованы контейнеры, стандартные строительные блоки и их взаимодействие. Использованы события для объектов и проведены тесты программы.