Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №4

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В JAVA ПРИЛОЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ SWING

по дисциплине «Платформа Java»

Выполнил:

Студент группы ИС/б 17-2-о

Черняев Н.Г.

Проверил:

Кузнецов С.А.

г. Севастополь 2019

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.

2.Постановка задачи

Вариант №10

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.

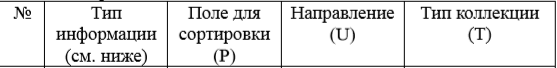
Переопределить методы:

* 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(<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)

1. Для реализации окна приложения реализовать дочерний класс JFrame.
2. Для представления таблицы с данными использовать компонент класса JTable, разместив его в контейнере JScrollPane (для возможности добавления полос прокрутки).
3. Поля ввода для добавления и редактирования данных реализовать текстовыми компонентами JTextField. Каждое поле снабдить подписью при помощи компонентов JLabel.
4. 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.
5. Для выбора файла при открытии и сохранении использовать компонент JFileChooser.
6. Удаление и сортировка элементов должно проходить по ключевому полю P (см. табл. 4.1). Направление сортировки U(см. табл. 4.1).







3.Текст программы

Класс Book:

**package** lab4java;

**import** java.util.Scanner;

**public** **class** Book {

String author;

**int** yearOfPublishing;

**int** amountOfPages;

String publishingHouse;

**public** Book(String author, **int** yearOfPublishing, **int** amountOfPages, String publishingHouse) {

**this**.author = author;

**this**.yearOfPublishing = yearOfPublishing;

**this**.amountOfPages = amountOfPages;

**this**.publishingHouse = publishingHouse;

}

**public** **int** getAmountOfPages() {

**return** amountOfPages;

}

**public** String getAuthor() {

**return** author;

}

**public** String getPublishingHouse() {

**return** publishingHouse;

}

**public** **int** getYearOfPublishing() {

**return** yearOfPublishing;

}

**public** **void** setAmountOfPages(**int** amountOfPages) {

**this**.amountOfPages = amountOfPages;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

**public** **void** setPublishingHouse(String publishingHouse) {

**this**.publishingHouse = publishingHouse;

}

**public** **void** setYearOfPublishing(**int** yearOfPublishing) {

**this**.yearOfPublishing = yearOfPublishing;

}

**public** **static** Book read(Scanner scn) {

**return** **new** Book(scn.next(),scn.nextInt(),scn.nextInt(),scn.next());

}

}

Класс BookComparator:

**package** lab4java;

**import** java.util.Comparator;

**public** **class** BookComparator **implements** Comparator<Book>{

@Override

**public** **int** compare(Book o1, Book o2) {

**return** o1.yearOfPublishing-o2.yearOfPublishing;

}

}

Класс BookColumnModel:

package lab4java;

import javax.swing.table.DefaultTableColumnModel;

import javax.swing.table.TableColumn;

import java.lang.reflect.Field;

public class BookColumnModel extends DefaultTableColumnModel{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

@SuppressWarnings("rawtypes")

public BookColumnModel(Class tClass, String[] headers) {

int i = 0;

for (Field tField : tClass.getDeclaredFields()){

TableColumn tColumn = new TableColumn();

tColumn.setHeaderValue(headers[i]);

tColumn.setModelIndex(getColumnCount());

tColumn.setIdentifier(tField);

tField.setAccessible(true);

this.addColumn(tColumn);

i++;

}

}

}

Класс BookTableModel:

**package** lab4java;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.Collections;

**import** java.util.LinkedList;

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

**public** **class** BookTableModel **extends** AbstractTableModel {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **final** String[] headers = { "Author", "Year of publishing", "Amount of pages", "Publishing house" };

@SuppressWarnings("unused")

**private** BookColumnModel columnModel;

**private** LinkedList<Book> books = **new** LinkedList<Book>();

**public** BookTableModel(BookColumnModel columnModel) {

**this**.columnModel = columnModel;

}

**public** String[] getHeaders() {

**return** headers;

}

@Override

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

**return** headers.length;

}

@Override

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

**switch** (columnIndex) {

**case** 0:

**return** String.**class**;

**case** 1:

**return** Integer.**class**;

**case** 2:

**return** Integer.**class**;

**case** 3:

**return** String.**class**;

}

**return** **null**;

}

@Override

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

**return** headers[columnIndex];

}

@Override

**public** **int** getRowCount() {

**return** books.size();

}

@Override

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

**return** **super**.isCellEditable(rowIndex, columnIndex);

}

@Override

**public** Object getValueAt(**int** i, **int** j) {

**if** (i >= books.size()) {

**return** **null**;

}

Book book = books.get(i);

**switch** (j) {

**case** 0:

**return** book.getAuthor();

**case** 1:

**return** book.getYearOfPublishing();

**case** 2:

**return** book.getAmountOfPages();

**case** 3:

**return** book.getPublishingHouse();

}

**return** **null**;

}

@Override

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

Book book = books.get(rowIndex);

**switch** (columnIndex) {

**case** 0:

book.setAuthor(String.*valueOf*(aValue));

**break**;

**case** 1:

book.setYearOfPublishing(Integer.*valueOf*(String.*valueOf*(aValue)));

**break**;

**case** 2:

book.setAmountOfPages(Integer.*valueOf*(String.*valueOf*(aValue)));

**break**;

**case** 3:

book.setPublishingHouse(String.*valueOf*(aValue));

**break**;

}

}

**public** **void** addRow(Book book) {

books.add(book);

fireTableDataChanged();

fireTableStructureChanged();

}

**public** **void** deleteRow(String authorField) {

**for** (**int** i = 0; i < books.size(); i++) {

**if** (books.get(i).author.equals(authorField)) {

books.remove(books.get(i));

fireTableDataChanged();

fireTableStructureChanged();

}

}

}

@SuppressWarnings("resource")

**public** **void** read(File file) **throws** IOException {

BufferedReader br = **new** BufferedReader(**new** FileReader(file));

{

String line = br.readLine();

String[] temp = line.split(",");

books.add(**new** Book(temp[0], Integer.*parseInt*(temp[1]), Integer.*parseInt*(temp[2]), temp[3]));

**while** ((line = br.readLine()) != **null**) {

temp = line.split(",");

books.add(**new** Book(temp[0], Integer.*parseInt*(temp[1]), Integer.*parseInt*(temp[2]), temp[3]));

}

}

fireTableDataChanged();

fireTableStructureChanged();

}

**public** **void** sort() {

Collections.*sort*(books, **new** BookComparator());

fireTableDataChanged();

fireTableStructureChanged();

}

}

Класс BookTable:

package lab4java;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import javax.swing.JTable;

public class BookTable extends JTable{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private BookColumnModel columnModel;

private BookTableModel bookTableModel = new BookTableModel(null);

public BookTable() {

setAutoCreateColumnsFromModel(false);

columnModel = new BookColumnModel(Book.class, bookTableModel.getHeaders());

setColumnModel(columnModel);

bookTableModel = new BookTableModel(columnModel);

setModel(bookTableModel);

}

public void addRow(Book book) {

bookTableModel.addRow(book);

}

public void deleteRow(String authorField) {

bookTableModel.deleteRow(authorField);

}

public void write(File file) {

int ColC = this.getColumnCount();

int ItemC = this.getRowCount();

StringBuilder sb;

FileWriter fw = null;

try {

fw = new FileWriter(file);

}catch (IOException e1) {

e1.printStackTrace();

}

for (int i = 0; i < ItemC; i++) {

sb = new StringBuilder();

for (int j = 0; j < ColC; j++) {

sb.append(this.getValueAt(i,j));

if (j < ColC - 1) sb.append(',');

if (j == ColC - 1) sb.append("\r\n");

}

try {

fw.write(sb.toString());

}catch (IOException e) {

e.printStackTrace();

}

sb = null;

}

try {

fw.close();

} catch (Exception e) {

e.printStackTrace();

}

bookTableModel.fireTableDataChanged();

}

public void read(File file) throws IOException {

bookTableModel.read(file);

}

public void sort() {

bookTableModel.sort();

}

}

Класс MainFrame:

**package** lab4java;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** javax.swing.JScrollPane;

**import** javax.swing.JButton;

**import** javax.swing.JFileChooser;

**import** java.awt.event.ActionListener;

**import** java.io.File;

**import** java.io.IOException;

**import** java.awt.event.ActionEvent;

**import** javax.swing.JTextField;

**import** javax.swing.JLabel;

**public** **class** MainFrame **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JPanel contentPane;

**private** JTextField textFieldAddAuthor;

**private** JTextField textFieldAddYearOfPublishing;

**private** JTextField textFieldAddAmounfOfPages;

**private** JTextField textFieldAddPublishingHouse;

**private** JTextField textFieldDeleteAuthor;

/\*\*

\* Launch the application.

\*/

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

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

MainFrame frame = **new** MainFrame();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

**public** MainFrame() {

setTitle("Lab4");

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

setBounds(100, 100, 514, 420);

contentPane = **new** JPanel();

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

setContentPane(contentPane);

contentPane.setLayout(**null**);

JScrollPane scrollPane = **new** JScrollPane();

scrollPane.setBounds(0, 0, 498, 211);

contentPane.add(scrollPane);

BookTable Table = **new** BookTable();

scrollPane.setViewportView(Table);

JButton btnAddButton = **new** JButton("Add");

btnAddButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Table.addRow(**new** Book(textFieldAddAuthor.getText(), Integer.*valueOf*(textFieldAddYearOfPublishing.getText()), Integer.*valueOf*((textFieldAddAmounfOfPages.getText())), textFieldAddPublishingHouse.getText()));

textFieldAddAuthor.setText("");

textFieldAddYearOfPublishing.setText("");

textFieldAddAmounfOfPages.setText("");

textFieldAddPublishingHouse.setText("");

}

});

btnAddButton.setBounds(399, 256, 89, 23);

contentPane.add(btnAddButton);

JButton btnDeleteButton = **new** JButton("Delete");

btnDeleteButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Table.deleteRow(textFieldDeleteAuthor.getText());

textFieldDeleteAuthor.setText("");

}

});

btnDeleteButton.setBounds(383, 305, 105, 23);

contentPane.add(btnDeleteButton);

JButton btnSortButton = **new** JButton("Sort");

btnSortButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Table.sort();

}

});

btnSortButton.setBounds(399, 222, 89, 23);

contentPane.add(btnSortButton);

JButton btnAddFileButton = **new** JButton("Add from file");

btnAddFileButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setCurrentDirectory(**new** File(System.*getProperty*("user.dir")));

**int** res = fileChooser.showDialog(MainFrame.**this**, "Load from file");

**if**(res == JFileChooser.***APPROVE\_OPTION***) {

**try** {

Table.read(fileChooser.getSelectedFile());

} **catch** (IOException e1) {

e1.printStackTrace();

}

}

}

});

btnAddFileButton.setBounds(10, 345, 186, 23);

contentPane.add(btnAddFileButton);

JButton btnWriteFileButton = **new** JButton("Write in file");

btnWriteFileButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setCurrentDirectory(**new** File(System.*getProperty*("user.dir")));

**int** res = fileChooser.showDialog(MainFrame.**this**, "Save to file");

**if**(res == JFileChooser.***APPROVE\_OPTION***) {

Table.write(fileChooser.getSelectedFile());

}

}

});

btnWriteFileButton.setBounds(268, 345, 186, 23);

contentPane.add(btnWriteFileButton);

textFieldAddAuthor = **new** JTextField();

textFieldAddAuthor.setBounds(0, 257, 86, 20);

contentPane.add(textFieldAddAuthor);

textFieldAddAuthor.setColumns(10);

JLabel labelAddAuthor = **new** JLabel("\u0410\u0432\u0442\u043E\u0440");

labelAddAuthor.setLabelFor(textFieldAddAuthor);

labelAddAuthor.setBounds(21, 239, 46, 14);

contentPane.add(labelAddAuthor);

textFieldAddYearOfPublishing = **new** JTextField();

textFieldAddYearOfPublishing.setColumns(10);

textFieldAddYearOfPublishing.setBounds(96, 257, 86, 20);

contentPane.add(textFieldAddYearOfPublishing);

JLabel labelAddYearOfPublishing = **new** JLabel("\u0413\u043E\u0434 \u0438\u0437\u0434\u0430\u043D\u0438\u044F");

labelAddYearOfPublishing.setLabelFor(textFieldAddYearOfPublishing);

labelAddYearOfPublishing.setBounds(96, 239, 86, 14);

contentPane.add(labelAddYearOfPublishing);

textFieldAddAmounfOfPages = **new** JTextField();

textFieldAddAmounfOfPages.setColumns(10);

textFieldAddAmounfOfPages.setBounds(192, 257, 101, 20);

contentPane.add(textFieldAddAmounfOfPages);

JLabel labelAddAmounfOfPages = **new** JLabel("\u041A\u043E\u043B-\u0432\u043E \u0441\u0442\u0440\u0430\u043D\u0438\u0446");

labelAddAmounfOfPages.setLabelFor(textFieldAddAmounfOfPages);

labelAddAmounfOfPages.setBounds(192, 239, 101, 14);

contentPane.add(labelAddAmounfOfPages);

textFieldAddPublishingHouse = **new** JTextField();

textFieldAddPublishingHouse.setColumns(10);

textFieldAddPublishingHouse.setBounds(303, 257, 86, 20);

contentPane.add(textFieldAddPublishingHouse);

JLabel labelAddPublishingHouse = **new** JLabel("\u0418\u0437\u0434\u0430\u0442\u0435\u043B\u044C\u0441\u0442\u0432\u043E");

labelAddPublishingHouse.setLabelFor(textFieldAddPublishingHouse);

labelAddPublishingHouse.setBounds(303, 239, 86, 14);

contentPane.add(labelAddPublishingHouse);

textFieldDeleteAuthor = **new** JTextField();

textFieldDeleteAuthor.setColumns(10);

textFieldDeleteAuthor.setBounds(202, 308, 138, 20);

contentPane.add(textFieldDeleteAuthor);

JLabel labelDeleteAuthor = **new** JLabel("\u0410\u0432\u0442\u043E\u0440");

labelDeleteAuthor.setLabelFor(textFieldDeleteAuthor);

labelDeleteAuthor.setBounds(223, 290, 117, 14);

contentPane.add(labelDeleteAuthor);

}

}

4.Результаты

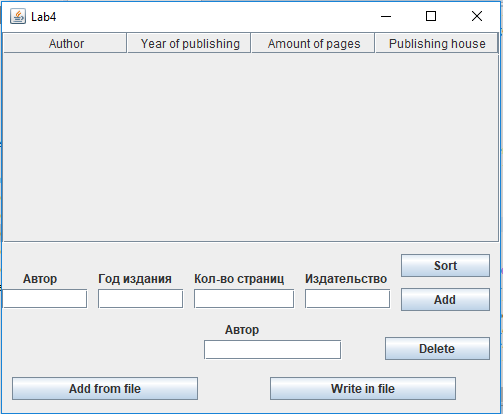


Рисунок 1 - Главное окно программы

Добавим данные из файла 1.txt

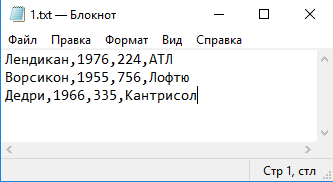


Рисунок 2 - Cодержимое файла 1.txt

Результат:

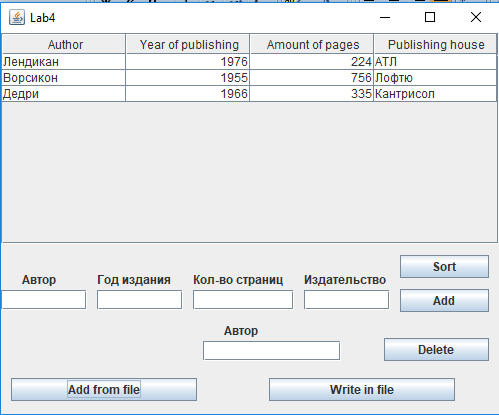


Рисунок 3 - Окно после добавления содержимого файла 1.txt

Добавим еще одну строку заполнив поля:

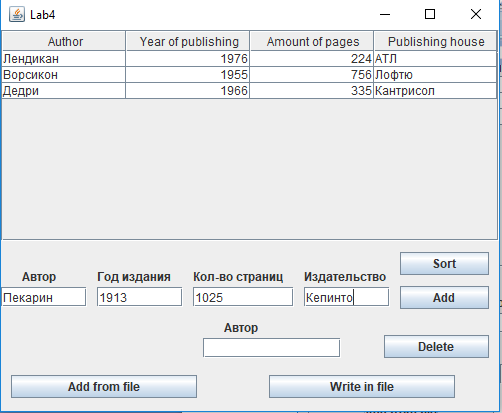


Рисунок 4 - Окно добавления строки

Отсортируем таблицу:

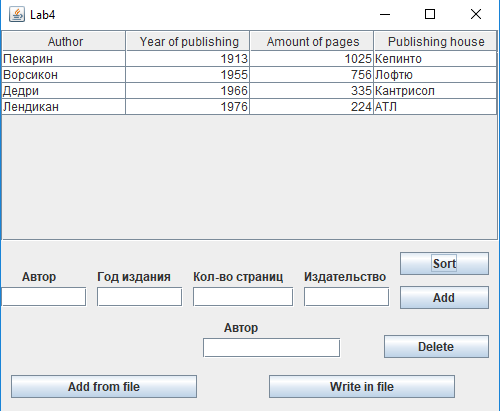


Рисунок 5 - Окно после сортировки таблицы

Удалим строку из таблицы:

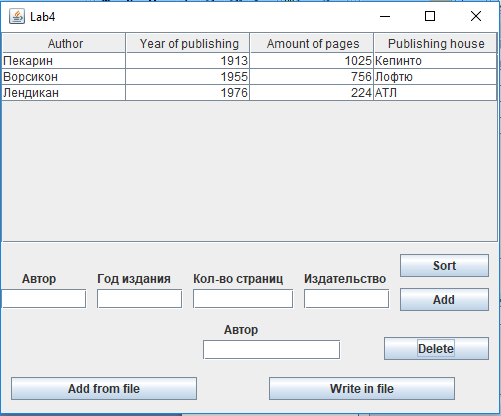


Рисунок 6 - Окно после удаления строки из таблицы

Запишем данную таблицу в файл 2.txt:

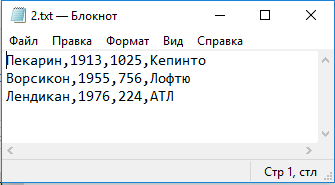


Рисунок 7 - Содержимое файла 2.txt

5.Вывод

Вданной лабораторной работе были изучены особенности инструментария библиотеки SWING для создания графического интерфейса приложений на языке Java и приобретены практические навыки создания Java-программ с графическим интерфейсом, позволяющие пользователю осуществлять взаимодействие с программой: задавать исходные данные, просматривать результаты работы программы в удобном виде.