Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |
| Кафедра вычислительной техники |
| наименование кафедры |

|  |
| --- |
| **Отчет** |
| по лабораторной работе №5 по дисциплине «Разработка приложения с оконным графическим пользовательским интерфейсом.» |
| наименование темы  Вариант №4 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | АСУб-19-1 |  |  |  | Аласаев П.М |
|  |  | шифр |  | подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  |  | Маланова Т.В. |
|  |  |  |  | подпись |  | И.О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2020 г.

Содержание

[1 Постановка задачи 3](#_Toc58958625)

[2 Проектирование графического интерфейса пользователя 4](#_Toc58958626)

[3 Описание структуры классов 5](#_Toc58958627)

[4 Таблица тестов 6](#_Toc58958628)

[5 Результаты тестирования 7](#_Toc58958629)

[6 Исходный код 11](#_Toc58958630)

[Список использованных источников 24](#_Toc58958631)

1 Постановка задачи

Для выполнения задания необходимо разработать графический пользовательский интерфейс средствами java для работы с классами реализованными при выполнении либо 3-ей, либо 4-ой лабораторной работы.

Графический пользовательский интерфейс должен предоставлять возможности работы со всеми возможными функциями программы (ввод и вывод данных в соответствующей форме, изменение условий выполнения задания, работа с файлами и т.д.). Состав используемых компонентов определяется самостоятельно, но для них должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий. Для этого может понадобиться модифицировать программный код из предыдущих лабораторных работ.

2 Проектирование графического интерфейса пользователя

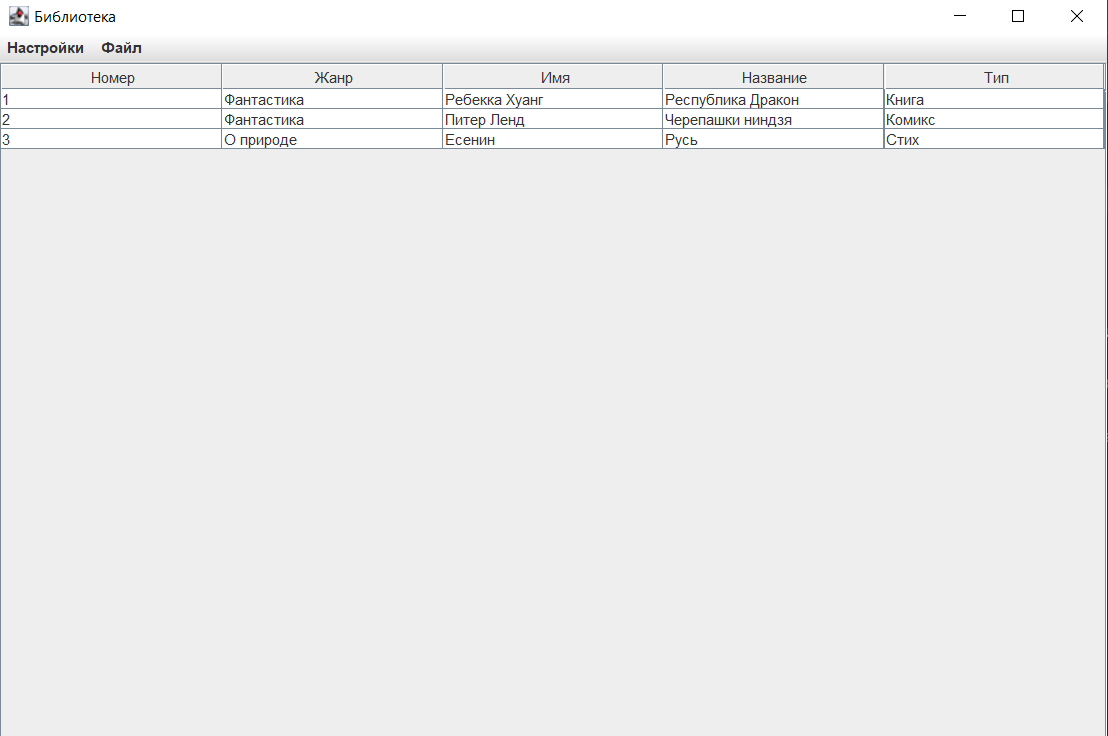


Рисунок 1 – Главное окно

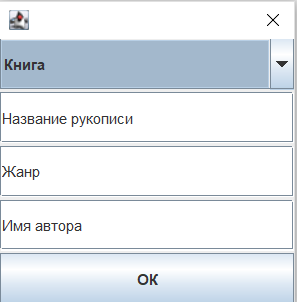


Рисунок 2 – Окно для добавления рукописи

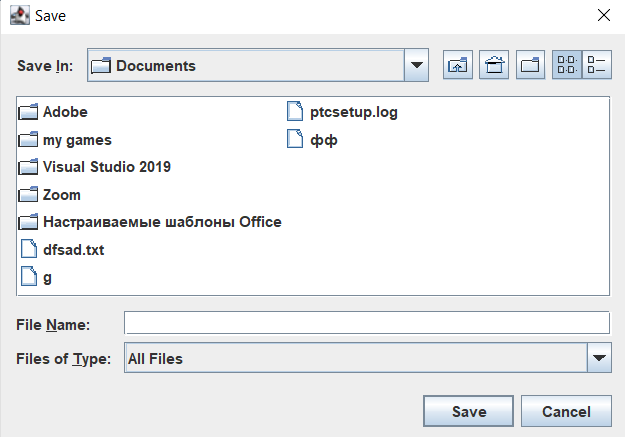


Рисунок 3 – Окно сохранения

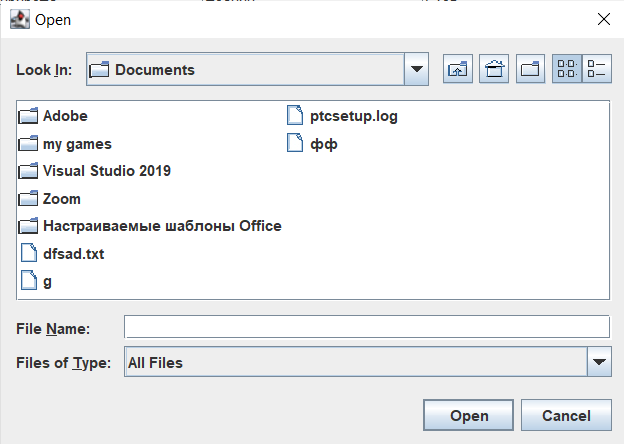


Рисунок 4 – Окно добавления

На Рис 1 мы видим таблицу. Для добавления новой строчки нужно нажать на кнопку настройки тогда выведется кнопки и при нажатии кнопки “Добавить рукопись” выходит окно Рис 2. Тут пользователю нужно ввести название, жанр, имя автора и выбрать рукопись (Книга, Комикс и Стих).

Далее на панеле есть вторая кнопка “Файл”, при ее нажатии выводятся кнопки “Сохранить файл” и “Открыть файл” при их нажатии выводится окно выбора файла.

3 Описание структуры классов

Manu.java – Запуск программы

MyWindow.java – Отвечает за вывод главного окна

MiniWindow.java – Отвечает за таблицу

ModalAdd.java – Отвечает за окно добавления

MainController.java – Отвечает за обработку удаления и добавления данных

4 Таблица тестов

Таблица 1 - Тесты

|  |  |  |
| --- | --- | --- |
| № | Действия | Выходные данные |
| 1 | Нажимаем на кнопку “Настройки” и нажимаем на кнопку “Добавить рукопись” | Открывается окно добавления. |
| 2 | Добавляем рукопись книги | Происходит добавление в таблицу |
| 3 | Не вводим в окно добавления данные | Выведется окно ошибки |
| 4 | Нажимаем на кнопку “Настройки” и нажимаем на кнопку “Удалить” | Выведется окно ошибки, что не выделили строку |
| 5 | Выделяем строку, которую хотим удалить | Происходит удаление строки |
| 6 | Нажимаем на кнопку “Файл” и нажимаем на кнопку “Сохранить файл” | Выводится окно сохранения в файл |
| 7 | Нажимаем на кнопку “Файл” и нажимаем на кнопку “Сохранить файл” | Выводится окно открытия файла |

5 Результаты тестирования

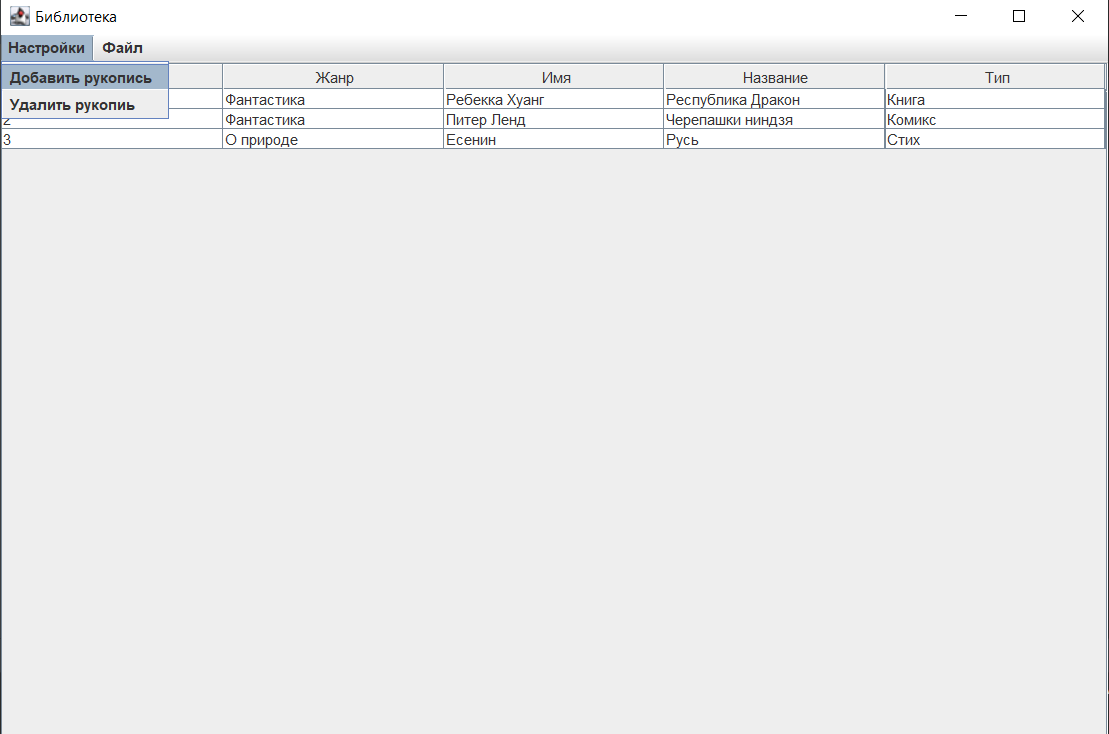


Рисунок 5 – Тест 1.1

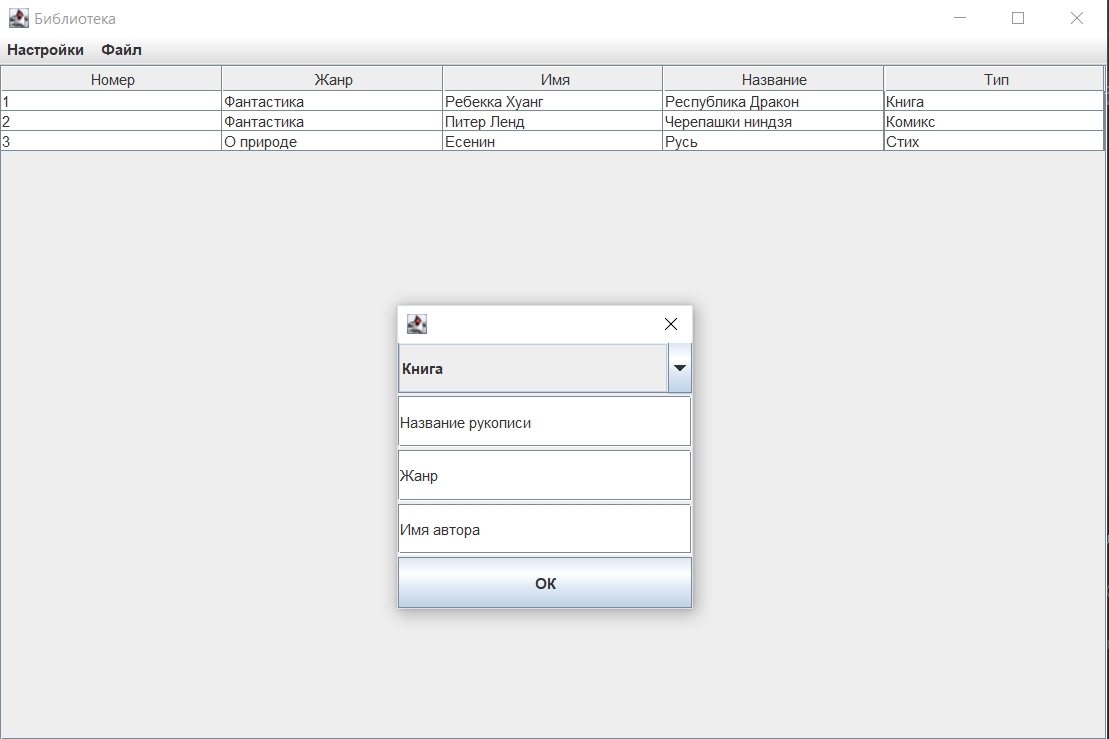


Рисунок 6 – Тест 1.2

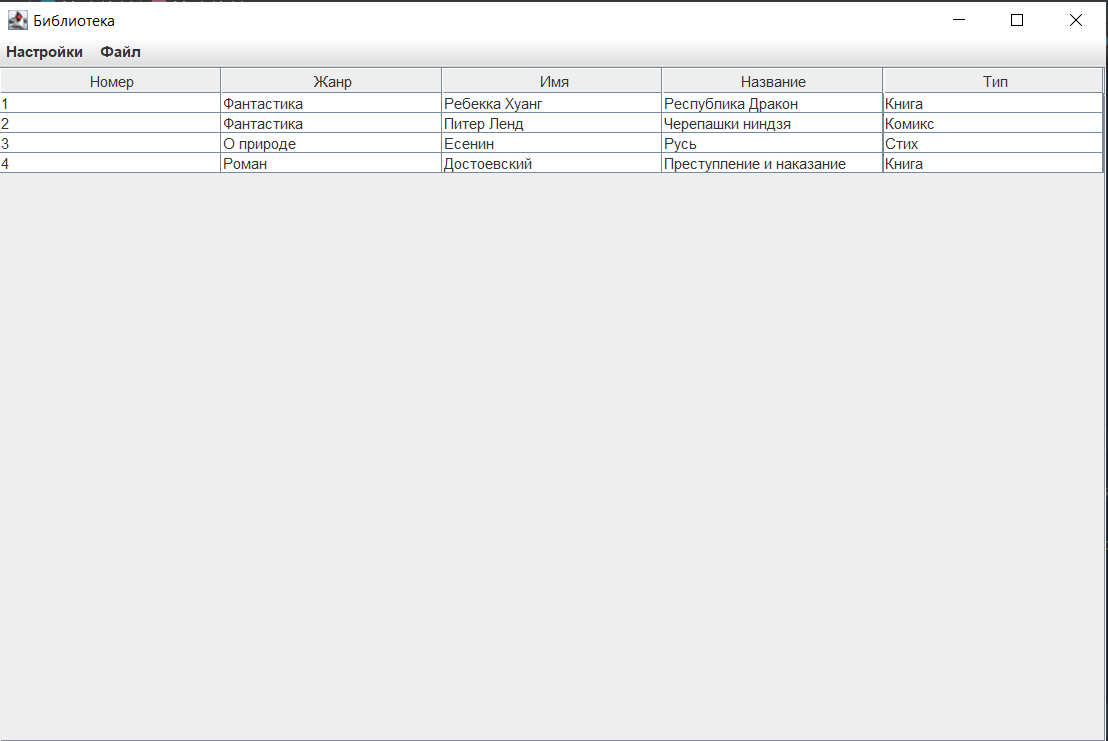


Рисунок 7 – Тест 2

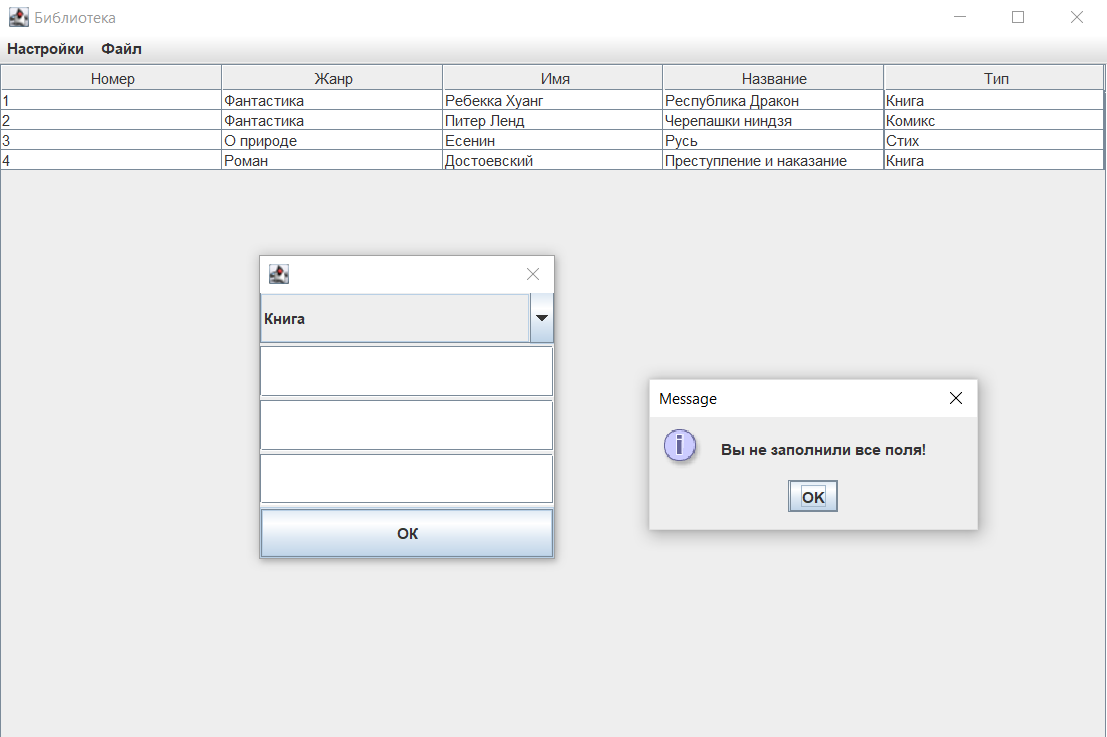


Рисунок 8 – Тест 3

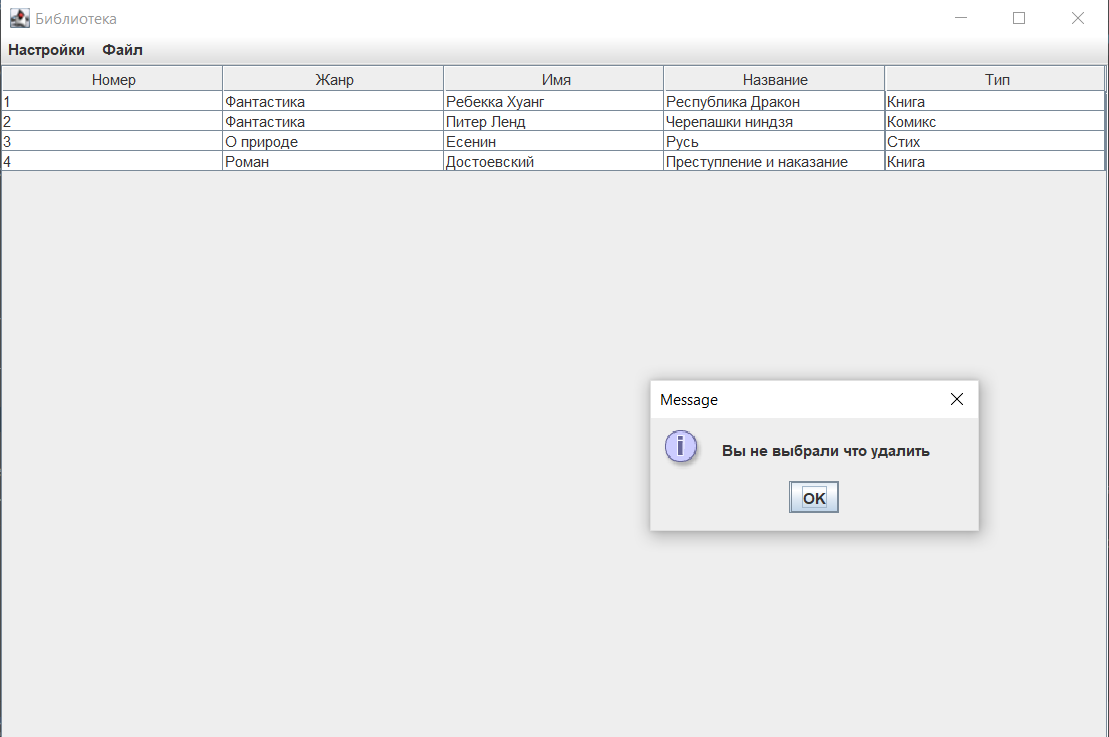


Рисунок 9 – Тест 4

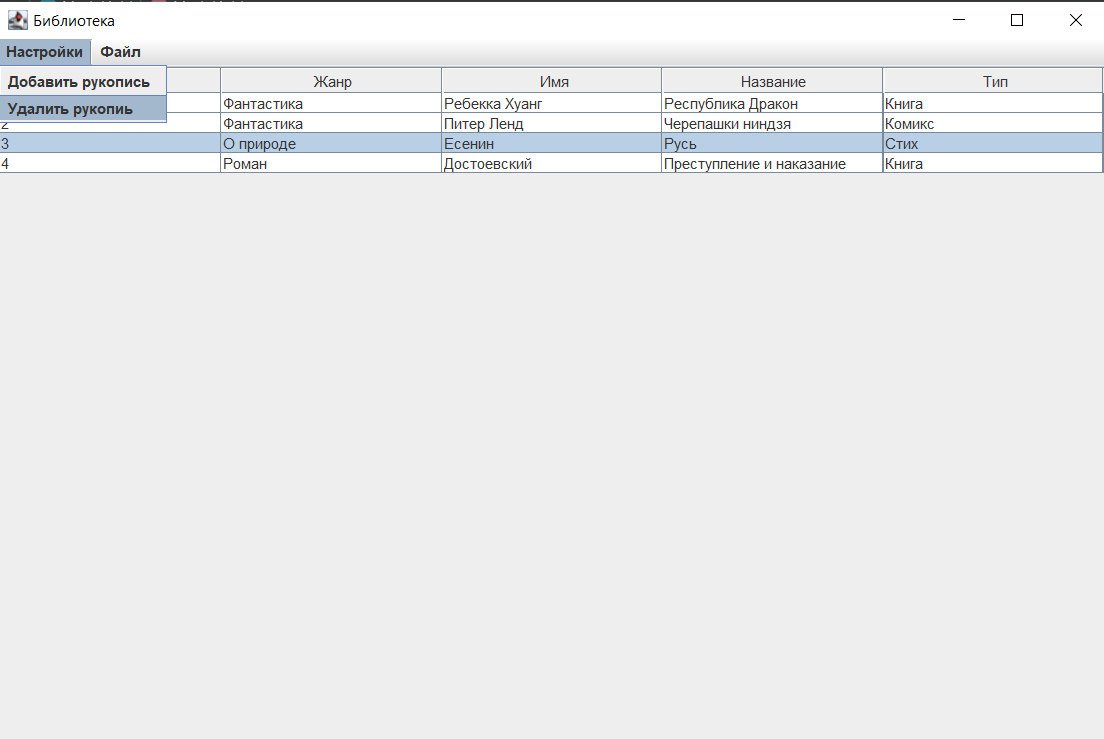


Рисунок 10 – Тест 5.1

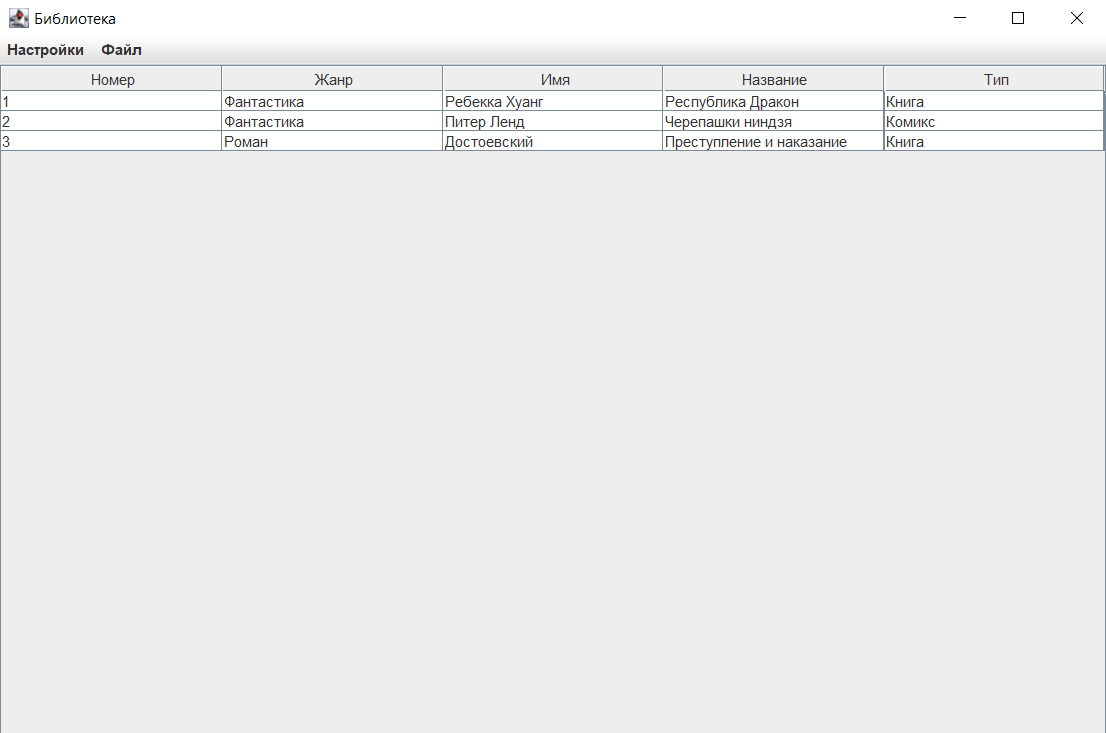


Рисунок 11 – Тест 5.2

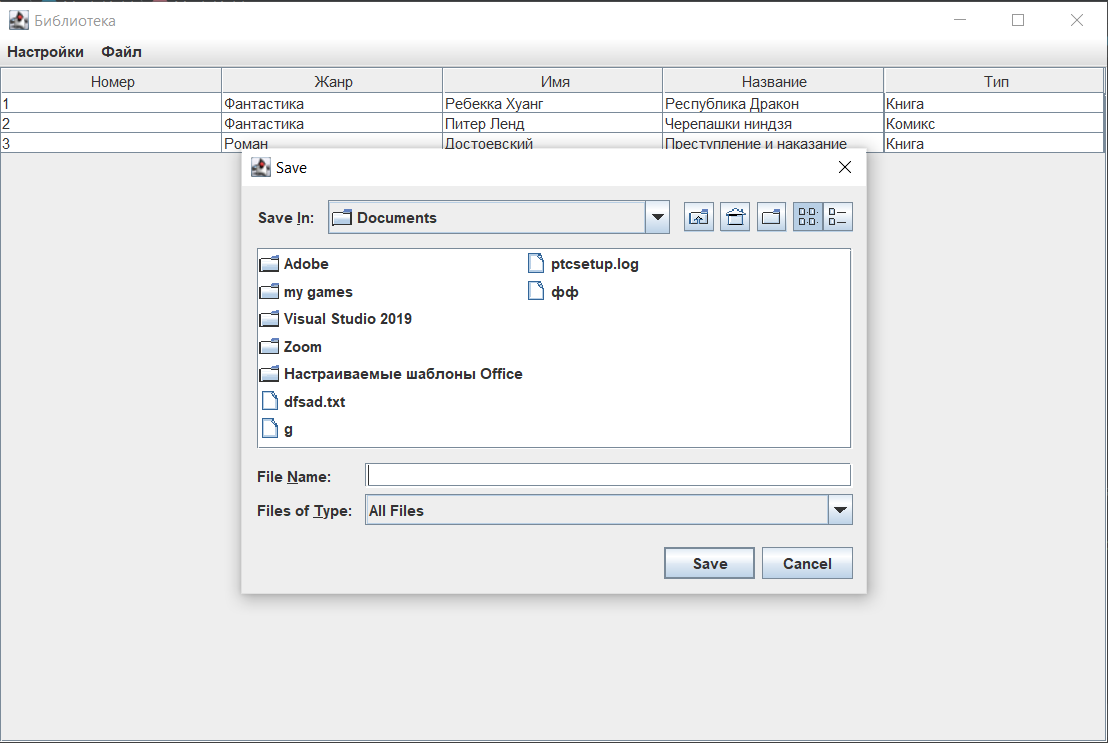


Рисунок 12 – Тест 6

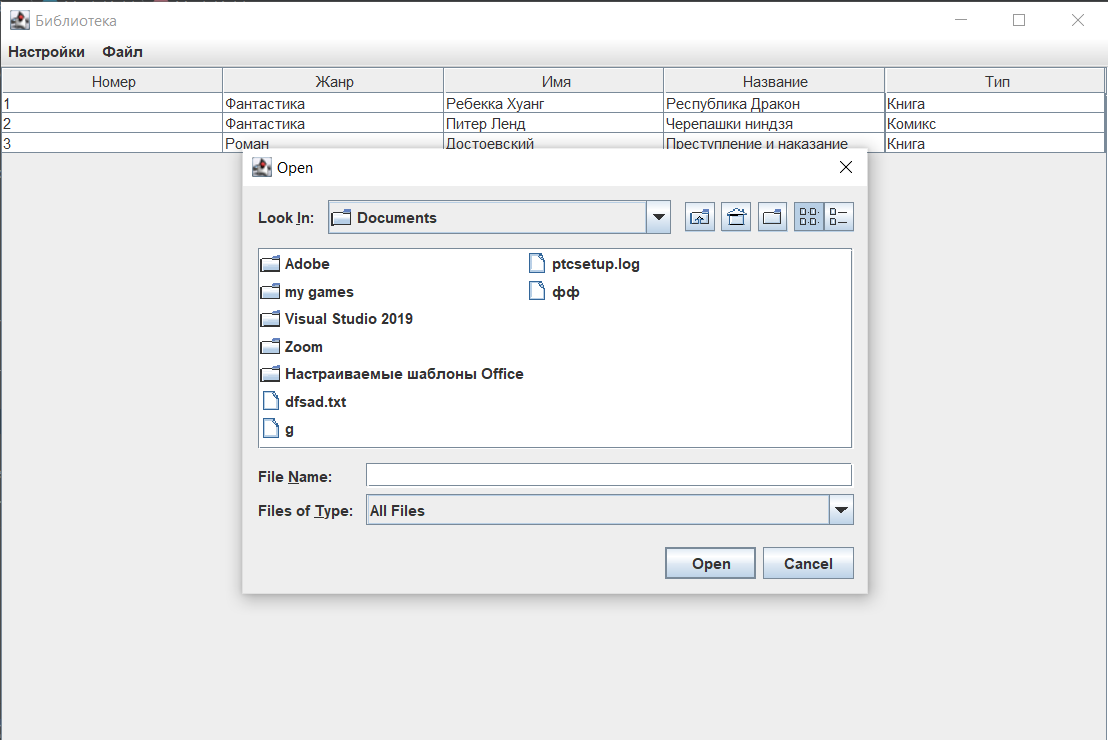


Рисунок 13 – Тест 7

6 Исходный код

1. Menu.java

import controllers.MainController;

import window.MyWindow;

public class Menu {

public static void main(String[] args){

new MainController().execute(new MyWindow());

}

}

1. MyWindow.java

package window;  
  
import box.Ruc;  
  
import javax.swing.\*;  
import javax.swing.table.TableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.io.File;  
  
public class MyWindow extends JFrame {  
  
 private JTable table;  
 private JMenu menu;  
 private JMenuItem a1, a2, a3, a4;  
 private MiniWindow model;  
 private ModalAdd modalAdd = new ModalAdd();  
 private JMenu menuFile;  
 private JFileChooser chooser;  
  
 public ModalAdd getModal() {  
 return modalAdd;  
 }  
 private JMenu getMenuFile() {  
 return menuFile;  
 }  
  
 public JTable getTable() {  
 return table;  
 }  
  
 public MiniWindow getModel() {  
 return model;  
 }  
  
 public JMenuItem getA2() {  
 return a2;  
 }  
  
 public JMenuItem getA1(){  
 return a1;  
 }  
  
 public JMenuItem getA3(){  
 return a3;  
 }  
  
 public JMenuItem getA4(){  
 return a4;  
 }  
  
 public void showModal() {  
 modalAdd.setVisible(true);  
 }  
  
 public void showErrorMessage(String msg) {  
 JOptionPane.*showMessageDialog*(null, msg);  
 }  
  
  
 public MyWindow() {  
  
 super("Библиотека");  
 table = new JTable();  
 chooser = new JFileChooser();  
 menu = new JMenu("Настройки");  
 menuFile = new JMenu("Файл");  
 JMenuBar m = new JMenuBar();  
 a1 = new JMenuItem("Добавить рукопись");  
 a2 = new JMenuItem("Удалить рукопиь");  
 a3 = new JMenuItem("Сохранение файл");  
 a4 = new JMenuItem("Окрыть файл");  
 model = new MiniWindow();  
  
 table.setModel(model);  
 JScrollPane pane = new JScrollPane(table);  
  
 setSize(900, 600);  
 setLayout(new GridLayout());  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
  
 menu.add(a1);  
 menu.add(a2);  
  
 menuFile.add(a3);  
 menuFile.add(a4);  
  
 m.add(menu);  
 m.add(menuFile);  
 add(pane);  
 setJMenuBar(m);  
  
 setVisible(true);  
 }  
  
 *//Открытие файла* public File openFile() {  
 int option = chooser.showOpenDialog(this);  
 if (option == JFileChooser.*APPROVE\_OPTION*) {  
 return chooser.getSelectedFile();  
 }  
 return null;  
 }  
  
 *//Сохранение файла* public File saveFile() {  
 int option = chooser.showSaveDialog(this);  
 if (option == JFileChooser.*APPROVE\_OPTION*) {  
 return chooser.getSelectedFile();  
 }  
 return null;  
 }  
}

1. MiniWindow.java

package window;  
  
import box.Library;  
import box.Ruc;  
  
import javax.swing.table.AbstractTableModel;  
  
public class MiniWindow extends AbstractTableModel {  
 private static Library *box*;  
  
 public MiniWindow() {  
 *box* = new Library();  
 }  
  
 public MiniWindow(Library library){  
 *box* = library;  
 }  
  
 public Library getBox(){  
 return *box*;  
 }  
  
 public void setBox(Library box){  
 MiniWindow.*box* = box;  
 fireTableDataChanged();  
 }  
  
 *//Удаление* public void delete(int num) throws IndexOutOfBoundsException {  
 *box*.delRucInd(num);  
 fireTableDataChanged();  
 }  
  
 *//Добавление* public void add(Ruc ruc) {  
 *box*.addRuc(ruc);  
 fireTableDataChanged();  
 }  
  
 @Override  
 public int getRowCount() {  
 return *box*.getRucList().size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return 5;  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 return switch (columnIndex) {  
 case 0 -> rowIndex + 1;  
 case 1 -> *box*.getRuc(rowIndex).getGenre();  
 case 2 -> *box*.getRuc(rowIndex).getName();  
 case 3 -> *box*.getRuc(rowIndex).getNameBook();  
 case 4 -> *box*.getRuc(rowIndex).getType();  
 default -> null;  
 };  
 }  
  
 @Override  
 public String getColumnName(int column){  
 return switch (column) {  
 case 0 -> "Номер";  
 case 1 -> "Жанр";  
 case 2 -> "Имя";  
 case 3-> "Название";  
 case 4 -> "Тип";  
 default -> "error";  
 };  
 }  
}

1. ModalAdd.java

package window;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class ModalAdd extends JDialog {  
 public JTextField nameRucInput;  
 public JTextField nameAuthorInput;  
 public JTextField genreInput;  
 public JComboBox typeCombo;  
 private JButton btn;  
  
 public JButton getBtn() {  
 return btn;  
 }  
  
 public ModalAdd() {  
  
 JPanel jPanel = new JPanel(new GridLayout(5, 1, 2, 2));  
  
 typeCombo= new JComboBox(new String[]{"Книга", "Комикс", "Стих"});  
 nameRucInput = new JTextField("Название рукописи");  
 nameAuthorInput = new JTextField("Имя автора");  
 genreInput = new JTextField("Жанр");  
 btn = new JButton("ОК");  
  
 jPanel.add(typeCombo);  
 jPanel.add(nameRucInput);  
 jPanel.add(genreInput);  
 jPanel.add(nameAuthorInput);  
 jPanel.add(btn);  
  
 add(jPanel);  
 pack();  
 setSize(250, 250);  
 setLocationRelativeTo(null);  
 }  
}

1. MainController.java

package controllers;  
  
import box.\*;  
import org.w3c.dom.ls.LSOutput;  
import window.ModalAdd;  
import window.MyWindow;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.io.File;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.StandardOpenOption;  
import java.util.ArrayList;  
  
public class MainController {  
 private MyWindow view;  
  
 *//метод для навешивания обработчиков модального окна добавления* public void executeModal(ModalAdd modalAdd) {  
 modalAdd.getBtn().addActionListener(new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String genre = modalAdd.genreInput.getText();  
 String nameAuthor = modalAdd.nameAuthorInput.getText();  
 String nameRuc = modalAdd.nameRucInput.getText();  
 String type = modalAdd.typeCombo.getSelectedItem().toString();  
  
 if (nameAuthor.equals("") || genre.equals("") || nameRuc.equals("")) {  
 view.showErrorMessage("Вы не заполнили все поля!");  
 return;  
 }  
  
 Ruc newRuc = switch(type) {  
 case "Комикс" -> new Comix(genre, nameAuthor, nameRuc);  
 case "Книга" -> new Book(genre, nameAuthor, nameRuc);  
 case "Стих" -> new Poems(genre, nameAuthor, nameRuc);  
 default -> null;  
 };  
  
 view.getModel().add(newRuc);  
 view.getModal().dispose();  
 }  
 });  
 }  
  
 public void execute(MyWindow view) {  
 this.view = view;  
 executeModal(view.getModal());  
  
 *//Добавление* view.getA1().addActionListener(new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 view.showModal();  
 }  
 });  
  
 *//Удаление* view.getA2().addActionListener(new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int ind = view.getTable().getSelectedRow();  
 try {  
 view.getModel().delete(ind);  
 } catch(IndexOutOfBoundsException err) {  
 view.showErrorMessage("Вы не выбрали что удалить");  
 }  
 }  
 });  
  
 *// Обработчик сохранения файла* view.getA3().addActionListener(new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 File file = view.saveFile();  
 if(file == null) return;  
 String path = file.getAbsolutePath();  
 String saveString = libraryToString(view.getModel().getBox());  
 try {  
 Files.*write*(Path.*of*(path), saveString.getBytes());  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 view.showErrorMessage("Файл не сохранен");  
 System.*out*.println("пРОИЗОШЛА ОШИБКА");  
 }  
 System.*out*.println("Файл успешно сохранен");  
 }  
 });  
  
 *// Обработчик открытия файла* view.getA4().addActionListener(new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 File file = view.openFile();  
 String stringLib = null;  
 try {  
 stringLib = Files.*readString*(Path.*of*(file.getAbsolutePath()));  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 view.showErrorMessage("Файл сломан ");  
 System.*out*.println("ОШИБКА В ОТКРЫТИИ ФАЙЛА");  
 } catch (NullPointerException err) {  
 return;  
 }  
  
 Library lib = null;  
 try {  
 lib = parseLibraryFromString(stringLib);  
 } catch (IndexOutOfBoundsException err) {  
 view.showErrorMessage("Файл сломан ");  
 System.*out*.println("файл некорректен");  
 return;  
 }  
 view.getModel().setBox(lib);  
 System.*out*.println("файл успешно открыт");  
 }  
 });  
 }  
  
 *//Превращение библиотеки в строку* private String libraryToString(Library lib) {  
 StringBuilder ans = new StringBuilder();  
 for (Ruc ruc : lib.getRucList()) {  
 ans.append(rucToString(ruc)).append("\r\n");  
 }  
 return ans.toString();  
 }  
  
 *// Преврощение рукописи в строковый формат* private String rucToString(Ruc ruc) {  
 String ans = "";  
 ans += ruc.getGenre() + ";";  
 ans += ruc.getName() + ";";  
 ans += ruc.getNameBook() + ";";  
 ans += ruc.getType();  
 return ans;  
 }  
  
 *// Получение библиотеки из строки* private Library parseLibraryFromString(String libString) {  
 ArrayList<Ruc> rucList = new ArrayList<>();  
 String[] stringsRuc = libString.split("\r\n");  
 for(String stringRuc : stringsRuc) {  
 rucList.add(parseRucFromString(stringRuc));  
 }  
 return new Library(rucList);  
 }  
  
 *//Получение рукописи из строки* private Ruc parseRucFromString(String rucString) {  
 String[] stringParams = rucString.split(";");  
 String genre = stringParams[0];  
 String name = stringParams[1];  
 String nameBook = stringParams[2];  
 String type = stringParams[3];  
 return switch (type) {  
 case "Книга" -> new Book(genre, name, nameBook);  
 case "Комикс" -> new Comix(genre, name, nameBook);  
 case "Стих" -> new Poems(genre, name, nameBook);  
 default -> throw new IllegalArgumentException("Неверный тип рукописи");  
 };  
 }  
}

6) Ruc.java

package box;  
  
import java.awt.\*;  
  
public abstract class Ruc {  
  
 private String genre;  
 private String name;  
 private String nameBook;  
 private String type;  
  
 public Ruc(String genre, String name, String nameBook, String type){  
 this.genre = genre;  
 this.name = name;  
 this.nameBook = nameBook;  
 this.type = type;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public String getGenre(){  
 return genre;  
 }  
  
 public void setGenre(String genre){  
 this.genre = genre;  
 }  
  
 public String getName(){  
 return name;  
 }  
  
 public void setName(String name){  
 this.name = name;  
 }  
  
 public String getNameBook(){  
 return nameBook;  
 }  
  
 public void setNameBook(String nameBook){  
 this.nameBook = nameBook;  
 }  
  
 public abstract void look();  
  
}

7) Book

package box;  
  
public class Book extends Ruc {  
  
 public Book(String genre, String name, String nameBook) {  
 super(genre, name, nameBook, "Книга");  
 }  
  
 @Override  
 public void look() {  
 System.*out*.println();  
 System.*out*.println("Имя автора: " + getName());  
 System.*out*.println("Название: " + getNameBook());  
 System.*out*.println("Жанр: " + getGenre());  
 System.*out*.println();  
 }  
}

8) Poems

package box;  
  
public class Poems extends Ruc {  
  
 public Poems(String genre, String name, String nameBook) {  
 super(genre, name, nameBook, "Стих");  
 }  
  
  
 @Override  
 public void look() {  
 System.*out*.println();  
 System.*out*.println("Имя автора: " + getName());  
 System.*out*.println("Название: " + getNameBook());  
 System.*out*.println("Жанр: " + getGenre());  
 System.*out*.println();  
 }  
}

9) Comix

package box;  
  
public class Comix extends Ruc {  
  
 public Comix(String genre, String name, String nameBook) {  
 super(genre, name, nameBook, "Комикс");  
 }  
  
 @Override  
 public void look() {  
 System.*out*.println();  
 System.*out*.println("Имя автора: " + getName());  
 System.*out*.println("Название: " + getNameBook());  
 System.*out*.println("Жанр: " + getGenre());  
 System.*out*.println();  
 }  
}

10)Library

package box;  
  
import box.Book;  
import box.Comix;  
import box.Poems;  
import box.Ruc;  
import java.util.ArrayList;  
  
public class Library {  
 ArrayList<Ruc> rucList = new ArrayList<>();  
  
 public Library() {  
 Book republicDragon = new Book("Фантастика", "Ребекка Хуанг", "Республика Дракон");  
 Comix turtles = new Comix("Фантастика","Питер Ленд", "Черепашки ниндзя");  
 Poems blackMan = new Poems("О природе", "Есенин", "Русь");  
 rucList.add(republicDragon);  
 rucList.add(turtles);  
 rucList.add(blackMan);  
 }  
  
 public Library(ArrayList<Ruc> ruc) {  
 rucList = ruc;  
 }  
 *//добавление рукописей в массив* public void addRuc(Ruc ruc){  
 rucList.add(ruc);  
 }  
  
 *//удаление рукописи из массива* public void delRuc(Ruc ruc){  
 rucList.remove(ruc);  
 }  
  
 *//удаление по индексу* public void delRucInd(int ind) {  
 rucList.remove(ind);  
 }  
  
 *//вывод рукописей* public ArrayList<Ruc> getRucList(){  
 return rucList;  
 }  
  
 public Ruc getRuc(int ind) {  
 return rucList.get(ind);  
 }  
 *//вывод имен* public String[] getRucName(){  
 String[] names = new String[rucList.size()];  
 for (int i = 0; i < rucList.size(); i++){  
 names[i] = rucList.get(i).getName();  
 }  
 return names;  
 }  
  
 *//поиск по жанрам* public ArrayList<Ruc> search(String genre){  
 ArrayList<Ruc> answer = new ArrayList<>();  
 for (Ruc a:rucList) {  
 if (a.getGenre().contains(genre)) {  
 answer.add(a);  
 }  
 }  
 return answer;  
 }  
  
 *//получение всех книг* public ArrayList<Book> getAllBooks(){  
 ArrayList<Book> answer = new ArrayList<>();  
 for (Ruc a:rucList) {  
 if (a instanceof Book) {  
 answer.add((Book) a);  
 }  
 }  
 return answer;  
 }  
  
 *//получение всех комиксов* public ArrayList<Comix> getAllComix(){  
 ArrayList<Comix> answer = new ArrayList<>();  
 for (Ruc a:rucList) {  
 if (a instanceof Comix) {  
 answer.add((Comix) a);  
 }  
 }  
 return answer;  
 }  
  
 *//получение всех стихов* public ArrayList<Poems> getAllPoems(){  
 ArrayList<Poems> answer = new ArrayList<>();  
 for (Ruc a:rucList) {  
 if (a instanceof Poems) {  
 answer.add((Poems) a);  
 }  
 }  
 return answer;  
 }  
  
 *//поиск по названию* public ArrayList<Ruc> searchNameBook(String nameBook) {  
 ArrayList<Ruc> answer = new ArrayList<>();  
 for (Ruc a : rucList) {  
 if (a.getNameBook().contains(nameBook)) {  
 answer.add(a);  
 }  
 }  
 return answer;  
 }  
  
 *//вывод информации* public void lookAll(){  
 for (Ruc a:rucList) {  
 a.look();  
 }  
 }  
}

Список использованных источников

1. Методические указания по выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во ИРНИТУ, 2017. – 24 c.
2. Файн Я. – Программирование на Java для детей, родителей, дедушек и бабушек, 2011