Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им.В.И.Ульянова (Ленина) »

Кафедра МОЭВМ

Лабораторная работа по ООП №8

по теме “Организация многопоточных приложений ”

Выполнил: Цацкис Артём гр. 3311

Проверил: Павловский М. Г.

Подпись преподавателя: _____

Санкт-Петербург

2024

Цель работы:

Знакомство с правилами и классами построения парал лельных приложений в языке Java.

Описание задания:

1. Создайте новый проект, который будет дублировать проект лабораторной работы № 7.

2. В новом проекте опишите 3 параллельных потока, один из которых будет загружать данные из XML-файла, второй – редактировать данные и формировать XML-файл для отчета, а третий – строить отчет в HTML- формате. Второй поток не должен формировать XML-файл для отчета, пока первый не загрузит данные в экранную форму, а третий поток не должен формировать отчет, пока второй поток редактирует данные и записывает их в XML-файл.

3. С помощью конструктора подготовьте шаблон для отчета.

4. Запустите приложение и убедитесь, что сформирован HTML-файл. Просмотрите его в браузере и проверьте правильность данных и формы.

5. Сгенерируйте документацию с помощью Javadoc и просмотрите ее в браузере.

XML-файл:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book>
        <title>Война и мир</title>
        <author>Лев Толстой</author>
        <font>Arial</font>
        <pinned>Нет</pinned>
    </book>
    <book>
        <title>1984</title>
        <author>Джордж Оруэлл</author>
        <font>Calibri</font>
        <pinned>Да</pinned>
    </book>
    <book>
        <title>Прощай оружие!</title>
        <author>Эрнест Хемингуэй</author>
        <font>Garamond</font>
        <pinned>Нет</pinned>
    </book>
    <book>
        <title>Убить пересмешника</title>
        <author>Харпер Ли</author>
        <font>Fraktur</font>
        <pinned>Да</pinned>
    </book>
    <book>
        <title>На дороге</title>
        <author>Джек Керуак</author>
        <font>Papyrus</font>
        <pinned>Нет</pinned>
    </book>
</library>
```

Код программы:

1.Основной класс

```
package lab6;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;

public class OOPlab6 {
    private JFrame bookList;
    private DefaultTableModel model;
    private JButton save, add, edit, delete, load;
    private JScrollPane scroll;
    private JTable books;

    public void show() {
        bookList = new JFrame("Информация о книгах");
        bookList.setSize(600, 400);
        bookList.setLocation(100, 100);
        bookList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        String[] columns = {"Название книги", "Автор", "Шрифт", "Закреплена?"};
        String[][] data = {
            {"Война и мир", "Лев Толстой", "Arial", "Нет"},
            {"1984", "Джордж Оруэлл", "Calibri", "Да"},
            {"Прощай оружие!", "Эрнест Хемингуэй", "Garamond", "Нет"},
            {"Убить пересмешника", "Харпер Ли", "Fraktur", "Да"},
            {"На дороге", "Джек Керуак", "Papyrus", "Нет"}
        };
        model = new DefaultTableModel(data, columns);
        books = new JTable(model);
        books.setAutoCreateRowSorter(true);
        scroll = new JScrollPane(books);
        bookList.getContentPane().add(scroll, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        save = new JButton("Сохранить");
        load = new JButton("Загрузить");
        add = new JButton("Добавить");
        edit = new JButton("Редактировать");
        delete = new JButton("Удалить");

        buttonPanel.add(save);
        buttonPanel.add(load);
        buttonPanel.add(add);
        buttonPanel.add(edit);
        buttonPanel.add(delete);
        bookList.getContentPane().add(buttonPanel, BorderLayout.SOUTH);

        save.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
                saveToXMLFile();
                JOptionPane.showMessageDialog(bookList, "Данные сохранены.");
            }
        });

        load.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                loadFromXMLFile();
                JOptionPane.showMessageDialog(bookList, "Данные загружены.");
            }
        });

        add.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JTextField titleField = new JTextField();
                JTextField authorField = new JTextField();
                JTextField fontField = new JTextField();
                String[] options = {"Да", "Нет"};
                JComboBox<String> pinnedField = new JComboBox<>(options);

                JPanel panel = new JPanel(new GridLayout(0, 1));
                panel.add(new JLabel("Название книги:"));
                panel.add(titleField);
                panel.add(new JLabel("Автор:"));
                panel.add(authorField);
                panel.add(new JLabel("Шрифт:"));
                panel.add(fontField);
                panel.add(new JLabel("Закреплена?"));
                panel.add(pinnedField);

                int result = JOptionPane.showConfirmDialog(bookList, panel, "Добавить
новую книгу",
                        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
                if (result == JOptionPane.OK_OPTION) {
                    String title = titleField.getText().trim();
                    String author = authorField.getText().trim();
                    String font = fontField.getText().trim();
                    String pinned = (String) pinnedField.getSelectedItem();

                    if (!title.isEmpty() && !author.isEmpty() && !font.isEmpty()) {
                        model.addRow(new Object[]{title, author, font, pinned});
                        JOptionPane.showMessageDialog(bookList, "Добавлена новая
книга.");
                    } else {
                        JOptionPane.showMessageDialog(bookList, "Пожалуйста,
заполните все поля.", "Ошибка", JOptionPane.WARNING_MESSAGE);
                    }
                }
            }
        });

        edit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int selectedRow = books.getSelectedRow();
                    editBook(selectedRow, "Новое название", "Новый автор", "Новый
шрифт", "Нет");
                    JOptionPane.showMessageDialog(bookList, "Книга
отредактирована.");
                } catch (InvalidBookOperationException ex) {
```

```
                    JOptionPane.showMessageDialog(bookList, ex.getMessage(), "Ошибка
редактирования", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        delete.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int selectedRow = books.getSelectedRow();
                    deleteBook(selectedRow);
                    JOptionPane.showMessageDialog(bookList, "Книга удалена.");
                } catch (BookDeletionException ex) {
                    JOptionPane.showMessageDialog(bookList, ex.getMessage(), "Ошибка
удаления", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        bookList.setVisible(true);
    }

    private void saveToXMLFile() {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.newDocument();

            Element root = doc.createElement("Books");
            doc.appendChild(root);

            for (int i = 0; i < model.getRowCount(); i++) {
                Element book = doc.createElement("Book");
                root.appendChild(book);

                Element title = doc.createElement("Title");
                title.appendChild(doc.createTextNode(model.getValueAt(i,
0).toString()));
                book.appendChild(title);

                Element author = doc.createElement("Author");
                author.appendChild(doc.createTextNode(model.getValueAt(i,
1).toString()));
                book.appendChild(author);

                Element font = doc.createElement("Font");
                font.appendChild(doc.createTextNode(model.getValueAt(i,
2).toString()));
                book.appendChild(font);

                Element pinned = doc.createElement("Pinned");
                pinned.appendChild(doc.createTextNode(model.getValueAt(i,
3).toString()));
                book.appendChild(pinned);
            }

            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(new File("books_data.xml"));

            transformer.transform(source, result);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void loadFromXMLFile() {
        try {
            File file = new File("books_data.xml");
            if (!file.exists()) return;

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(file);

            NodeList nodeList = doc.getElementsByTagName("Book");
            model.setRowCount(0); // Очистка текущих данных

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;

                    String title =
element.getElementsByTagName("Title").item(0).getTextContent();
                    String author =
element.getElementsByTagName("Author").item(0).getTextContent();
                    String font =
element.getElementsByTagName("Font").item(0).getTextContent();
                    String pinned =
element.getElementsByTagName("Pinned").item(0).getTextContent();

                    model.addRow(new Object[]{title, author, font, pinned});
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void editBook(int rowIndex, String title, String author, String font,
String pinned) throws InvalidBookOperationException {
        if (rowIndex < 0 || rowIndex >= model.getRowCount()) {
            throw new InvalidBookOperationException("Не выбрана книга");
        }
        model.setValueAt(title, rowIndex, 0);
        model.setValueAt(author, rowIndex, 1);
        model.setValueAt(font, rowIndex, 2);
        model.setValueAt(pinned, rowIndex, 3);
    }

    public void deleteBook(int rowIndex) throws BookDeletionException {
        if (rowIndex < 0 || rowIndex >= model.getRowCount()) {
            throw new BookDeletionException("Не выбрана книга");
        }
        model.removeRow(rowIndex);
    }

    public static void main(String[] args) {
        new OOPlab6().show();
    }
```

```
}

class InvalidBookOperationException extends Exception {
    public InvalidBookOperationException(String message) {
        super(message);
    }
}

class BookDeletionException extends Exception {
    public BookDeletionException(String message) {
        super(message);
    }
}
```

2.Класс загрузки данных

```
package lab7;

import java.io.File;
import java.util.concurrent.CountDownLatch;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.swing.table.DefaultTableModel;
import org.w3c.dom.*;

public class DataLoader implements Runnable {
    private CountDownLatch latch;

    public DataLoader(CountDownLatch latch) {
        this.latch = latch;
    }

    @Override
    public void run() {
        try {
            System.out.println("Загрузка данных из XML...");
            File xmlFile = new File("books.xml");
            if (!xmlFile.exists()) {
                System.out.println("Файл XML не найден. Создайте файл books.xml.");
                latch.countDown();
                return;
            }

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(xmlFile);

            doc.getDocumentElement().normalize();
            NodeList nodeList = doc.getElementsByTagName("Book");

            DefaultTableModel model = new DefaultTableModel(new String[]{"Название",
"Автор", "Шрифт", "Закреплена?"}, 0);

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);

                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;
                    String title =
element.getElementsByTagName("Title").item(0).getTextContent();
                    String author =
element.getElementsByTagName("Author").item(0).getTextContent();
```

```
                    String font =
element.getElementsByTagName("Font").item(0).getTextContent();
                    String pinned =
element.getElementsByTagName("Pinned").item(0).getTextContent();

                    model.addRow(new Object[]{title, author, font, pinned});
                }
            }

            System.out.println("Данные загружены: " + model.getRowCount() + "
книг.");
            latch.countDown(); // Сообщаем, что поток завершен
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.класс редактирования данных

```
package lab8;

import java.io.File;
import java.util.concurrent.CountDownLatch;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;

public class DataEditor implements Runnable {
    private CountDownLatch latch1;
    private CountDownLatch latch2;

    public DataEditor(CountDownLatch latch1, CountDownLatch latch2) {
        this.latch1 = latch1;
        this.latch2 = latch2;
    }

    @Override
    public void run() {
        try {
            latch1.await(); // Ждем завершения загрузки данных
            System.out.println("Редактирование данных...");

            File xmlFile = new File("books.xml");
            Document doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(xmlFile);
            doc.getDocumentElement().normalize();

            // Пример: Добавляем новую книгу
            Element root = doc.getDocumentElement();
            Element newBook = doc.createElement("Book");

            Element title = doc.createElement("Title");
            title.appendChild(doc.createTextNode("Новая книга"));
            newBook.appendChild(title);

            Element author = doc.createElement("Author");
            author.appendChild(doc.createTextNode("Новый автор"));
            newBook.appendChild(author);

            Element font = doc.createElement("Font");
```

```
            font.appendChild(doc.createTextNode("Courier"));
            newBook.appendChild(font);

            Element pinned = doc.createElement("Pinned");
            pinned.appendChild(doc.createTextNode("Нет"));
            newBook.appendChild(pinned);

            root.appendChild(newBook);

            // Сохраняем изменения в XML-файл
            Transformer transformer =
TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(new File("books.xml"));
            transformer.transform(source, result);

            System.out.println("XML-файл обновлен.");
            latch2.countDown(); // Сообщаем, что поток завершен
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4.класс создания отчетов:

```
package lab8;

import java.io.*;
import java.util.concurrent.CountDownLatch;

public class ReportGenerator implements Runnable {
    private CountDownLatch latch;

    public ReportGenerator(CountDownLatch latch) {
        this.latch = latch;
    }

    @Override
    public void run() {
        try {
            latch.await(); // Ждем завершения формирования XML
            System.out.println("Генерация HTML-отчета...");

            File htmlFile = new File("report.html");
            try (BufferedWriter writer = new BufferedWriter(new
FileWriter(htmlFile))) {
                writer.write("<!DOCTYPE html>");
                writer.write("<html><head><title>Отчет о
книгах</title></head><body>");
                writer.write("<h1>Отчет о книгах</h1>");
                writer.write("<table
border='1'><tr><th>Название</th><th>Автор</th><th>Шрифт</th><th>Закреплена?</th></tr>
");
                writer.write("<tr><td>Война и мир</td><td>Лев
Толстой</td><td>Arial</td><td>Нет</td></tr>");
                writer.write("<tr><td>1984</td><td>Джордж
Оруэлл</td><td>Calibri</td><td>Да</td></tr>");
                writer.write("</table>");
                writer.write("</body></html>");
            }
```

```
            System.out.println("HTML-отчет создан: report.html");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Ссылка на репозиторий Git-hub:

TsatskisArtem/For-oop8