**Лабораторная работа № 5**

**Бинарные файлы.**

Написать приложение, позволяющее создавать целочисленный бинарный файл, выводить его в многострочный редактор текстов или таблицу, а также производить указанные действия (создание, вывод и операции совсем не обязательно должны выполняться последовательно: пользователь может выбрать ранее созданный файл). Количество чисел в создаваемом файле задается случайным образом из диапазона 1..n, где n вводится. Имя файла для создания и обработки запрашивается с помощью диалоговых окон. После создания или выбора файла его элементы выводятся. Программа должна выдавать ответ при любых введенных пользователем данных. **Интерфейс программы (используется фреймворк vaadin), кроме оговоренных моментов произвольный, максимально удобный для пользователя и по возможности эстетичный.**

|  |
| --- |
| 5. Вывести по возрастанию все числа файла, которые взаимно просты с другими числами этого файла. Выводить числа дважды не нужно, даже если оно встречается несколько раз. Считывать элементы файла в массив или другую структуру нельзя, но для других целей ее создать можно. Компоненты файла вводятся с клавиатуры, диапазон вводимых чисел а..b. Считать, что число элементов файла не ограничено конкретным числом. При обработке файл можно прочитать только один раз. |

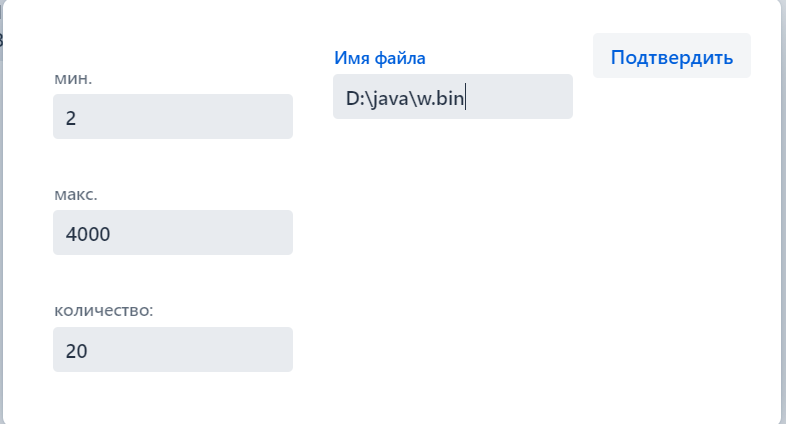


Рис.1. Создание нового файла

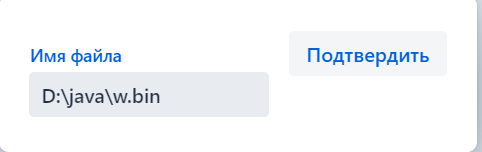


Рис.2. Диалог выбора файла

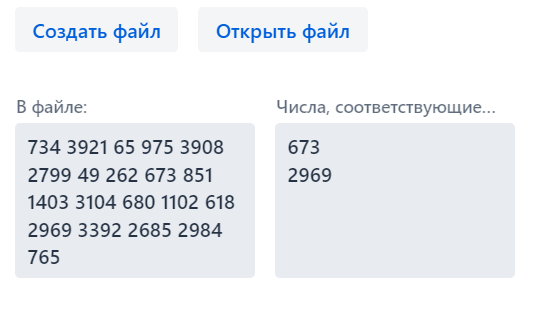


Рис.3. Результат

package org.example;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextArea;  
import com.vaadin.flow.data.provider.ListDataProvider;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
  
import java.io.\*;  
import java.nio.ByteBuffer;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.\*;  
import javax.swing.JFileChooser;  
  
@Route  
public class MainView extends VerticalLayout {  
  
  
 private TextArea text1=new TextArea("В файле:");  
 private TextArea text2=new TextArea("Числа, соответствующие условию:");  
 //private TextArea text3=new TextArea("делители");  
 private TextField textFieldA = new TextField("Введите минимальное значение");  
 private TextField textFieldB = new TextField("Введите максимальное значение");  
 private TreeSet<Integer> ts=new TreeSet<Integer>();  
 private Map<Integer, Integer> divisorsCount = new HashMap<>();  
  
 public MainView() {  
 Button buttonCreateFile = new Button("Создать файл");  
 Button buttonOpenFile = new Button("Открыть файл");  
  
 buttonCreateFile.addClickListener(event -> showDialogForCreatingFile());  
 buttonOpenFile.addClickListener(event -> openExistingFile());  
  
 HorizontalLayout l= new HorizontalLayout(buttonCreateFile, buttonOpenFile);  
 HorizontalLayout l2= new HorizontalLayout(text1, text2);  
 add(l);  
 add(l2);  
  
 // Настройка сетки  
 //grid.setAllRowsVisible(true);  
 //Grid.Column<Integer> число = grid.addColumn(Integer::toString).setHeader("Число");  
 }  
  
 private HorizontalLayout createButtonPanel() {  
 Button buttonCreateFile = new Button("Создать файл");  
 Button buttonOpenFile = new Button("Открыть файл");  
  
 buttonCreateFile.addClickListener(event -> showDialogForCreatingFile());  
 buttonOpenFile.addClickListener(event -> openExistingFile());  
  
 return new HorizontalLayout(buttonCreateFile, buttonOpenFile);  
 }  
  
 private void showDialogForCreatingFile() {  
 Dialog dialog = new Dialog();  
 TextField fileNameField = new TextField("Имя файла");  
 TextField min = new TextField("мин.");  
 TextField max = new TextField("макс.");  
 TextField num = new TextField("количество:");  
 Button confirmButton = new Button("Подтвердить");  
  
 confirmButton.addClickListener(e -> {  
 String fileName = fileNameField.getValue().trim();  
 if (!fileName.isEmpty()) {  
 try {  
 List<Integer> numbers = generateRandomNumbers(Integer.*parseInt*(min.getValue()), Integer.*parseInt*(max.getValue()), Integer.*parseInt*(num.getValue()));  
 writeToFile(fileName, numbers);  
 Notification.*show*("Файл успешно создан.");  
 dialog.close();  
 } catch (Exception ex) {  
 Notification.*show*(ex.getMessage(), 3000, Notification.Position.*MIDDLE*);  
 }  
 } else {  
 Notification.*show*("Пожалуйста, введите имя файла.", 3000, Notification.Position.*BOTTOM\_START*);  
 }  
 });  
 VerticalLayout l1=new VerticalLayout();  
 l1.add(min,max,num);  
 dialog.add(new HorizontalLayout(  
 l1,  
 fileNameField,  
 confirmButton  
 ));  
 dialog.open();  
  
 }  
  
 private void openExistingFile() {  
 Dialog dialog = new Dialog();  
 TextField fileNameField = new TextField("Имя файла");  
 Button confirmButton = new Button("Подтвердить");  
  
 confirmButton.addClickListener(e -> {  
 String fileName = fileNameField.getValue().trim();  
 if (!fileName.isEmpty()) {  
 try {  
 readAndProcessFile(fileName);  
 Notification.*show*("Файл прочитан.");  
 dialog.close();  
 } catch (Exception ex) {  
 Notification.*show*(ex.getMessage(), 3000, Notification.Position.*MIDDLE*);  
 }  
 } else {  
 Notification.*show*("Пожалуйста, введите имя файла.", 3000, Notification.Position.*BOTTOM\_START*);  
 }  
 });  
  
 dialog.add(new HorizontalLayout(  
 fileNameField,  
 confirmButton  
 ));  
 dialog.open();  
  
 }  
 private List<Integer> generateRandomNumbers(int min, int max, int num) {  
 Random random =new Random();  
 List<Integer> numbers = new ArrayList<>();  
 for (int i = 0; i < num; i++) {  
 numbers.add(random.nextInt(max-min+1)+min);  
 }  
 return numbers; // возвращаем заполненный список чисел  
 }  
 private void writeToFile(String fileName, List<Integer> numbers)  
 throws IOException {  
  
 try (FileOutputStream outputStream = new FileOutputStream(fileName)) {  
 for (Integer number : numbers) {  
 byte[] bytes = ByteBuffer.*allocate*(4).putInt(number).array();  
 outputStream.write(bytes);  
 }  
 }  
 }  
 private void readAndProcessFile(String selectedFile) {  
 text1.setValue("");  
 text2.setValue("");  
 ts.clear();  
 //divisors.clear();  
  
 try (DataInputStream inputStream = new DataInputStream(new FileInputStream(selectedFile))) {  
 byte[] bytes = new byte[4]; // Буфер для чтения 4 байтов  
  
 while (inputStream.read(bytes) != -1) { // Читаем блоками по 4 байта  
 int number = ByteBuffer.*wrap*(bytes).getInt(); // Преобразуем байты в целое число  
 processNumber(number);  
 text1.setValue(text1.getValue() + number + " "); // Обновляем текст  
 }  
 removeNonCoprimeNumbers();  
 // Выводим содержимое ts в text2  
 for (Integer i : ts) {  
 text2.setValue(text2.getValue() + i+"\n");  
 }  
// for (Integer i : divisorsCount.values()) {  
// text3.setValue(text3.getValue() + "\n" + i);  
// }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void processNumber(int number) {  
 Set<Integer> currentDivisors = getDivisors(number);  
  
 // Проверяем, есть ли общие делители с уже известными  
 for (int divisor : currentDivisors) {  
 if (divisorsCount.containsKey(divisor)) {  
 divisorsCount.put(divisor, divisorsCount.get(divisor) + 1);  
 } else {  
 divisorsCount.put(divisor, 1);  
 }  
 }  
  
 // Добавляем число в список  
 ts.add(number);  
 }  
  
 private void removeNonCoprimeNumbers() {  
 Iterator<Integer> iterator = ts.iterator();  
  
 while (iterator.hasNext()) {  
 int number = iterator.next();  
 Set<Integer> divisors = getDivisors(number);  
  
 boolean shouldRemove = false;  
 for (int divisor : divisors) {  
 if (divisorsCount.get(divisor) > 1) {  
 shouldRemove = true;  
 break;  
 }  
 }  
  
 if (shouldRemove) {  
 iterator.remove();  
 }  
 }  
 }  
  
 private Set<Integer> getDivisors(int number) {  
 Set<Integer> divisors = new TreeSet<>();  
 for (int i = 1; i <= Math.*sqrt*(number); i++) {  
 if (number % i == 0 && i!=1) {  
 divisors.add(i);  
 if (i != number / i && number / i!=1) {  
 divisors.add(number / i);  
 }  
 }  
 }  
 //divisors.add(number);  
 return divisors;  
 }  
  
}