Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2-3

По дисциплине: «Современные системы программирования»

Тема: «Создание меню с использованием стандартных диалогов. Консольное приложение. План тестирования приложения»

Выполнил:

Студент 4 курса

Группы ИИ-16 (2)

Пешко А.С.

Проверил:

Коренкович О. Г.

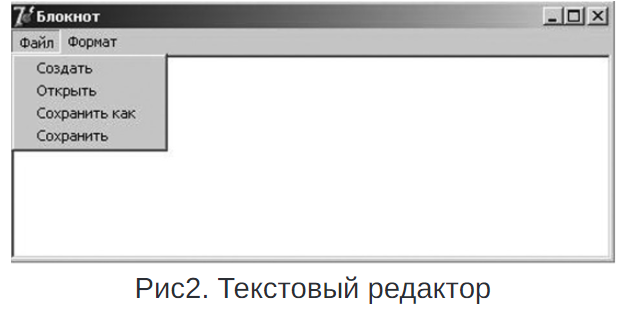
Брест 2021

**Цель работы:** научиться конструировать меню; освоить использование объектов класса и стандартных диалогов; освоить создание консольных приложений; освоить работу с файлами; научиться строить план тестирования приложения.

**Ход работы**

**Часть 1**

**Задача 1.** Создать простейший текстовый редактор, позволяющий вводить и редактировать текст, управлять шрифтом и цветом окна, а также открывать и сохранять файлы.



**Задача 2.**

Добавьте возможность автоматического присвоения файлам расширения xml и отображения в диалоговых окнах только файлов соответствующего типа.

**Задача 3.**

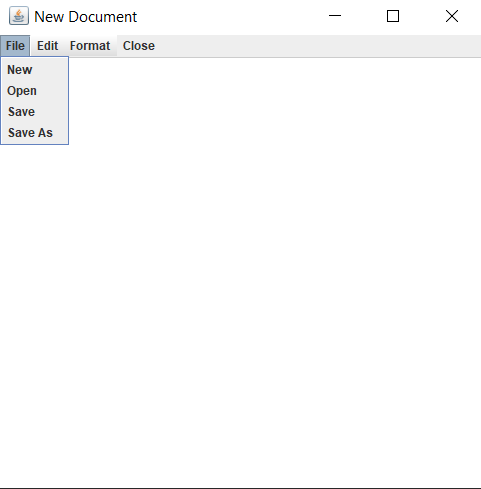
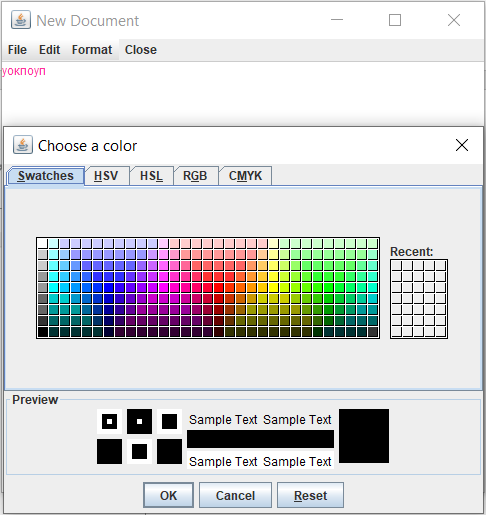
Добавьте к своему проекту отображение имени открытого файла (пути к файлу) в области заголовка формы.

**Задача 4.**

Реализуйте пункт меню Сохранить, позволяющий, если файл ранее был открыт или сохранен, сохранять его по прежнему адресу без дополнительного диалога.

**Код программы:**

public class TextEditor extends JFrame {  
  
 private static final String APP\_PATH = "/home/anastasiya/university/CCP/lab2\_3";  
  
 private JTextArea textArea;  
 private boolean isSaved = false;  
 private String currentFileLocation = "default";  
  
 public static void main(String[] args) {  
 new TextEditor();  
 }  
  
 public TextEditor() {  
  
 JMenuBar menuBar = new JMenuBar();  
 JMenu fileMenu = new JMenu("File");  
 JMenuItem newFileMenuItem = new JMenuItem("New");  
 JMenuItem openFileMenuItem = new JMenuItem("Open");  
 JMenuItem saveFileMenuItem = new JMenuItem("Save");  
 JMenuItem saveAsFileMenuItem = new JMenuItem("Save As");  
  
 newFileMenuItem.addActionListener((e) -> {  
 setTitle("New Document");  
 textArea.setText("");  
 isSaved = false;  
 });  
 openFileMenuItem.addActionListener((e) -> {  
 this.isSaved = true;  
 openFile();  
 });  
 saveFileMenuItem.addActionListener((e) -> {  
 if (isSaved) {  
 save();  
 } else {  
 saveAs();  
 }  
 });  
 saveAsFileMenuItem.addActionListener((e) -> {  
 saveAs();  
 });  
  
 fileMenu.add(newFileMenuItem);  
 fileMenu.add(openFileMenuItem);  
 fileMenu.add(saveFileMenuItem);  
 fileMenu.add(saveAsFileMenuItem);  
  
 JMenu editMenu = new JMenu("Edit");  
 JMenuItem cutEditMenuItem = new JMenuItem("Cut");  
 JMenuItem copyEditMenuItem = new JMenuItem("Copy");  
 JMenuItem pasteEditMenuItem = new JMenuItem("Paste");  
 JMenuItem selectAllEditMenuItem = new JMenuItem("Select All");  
 cutEditMenuItem.addActionListener((e) -> {  
 textArea.cut();  
 });  
 copyEditMenuItem.addActionListener((e) -> {  
 textArea.copy();  
 });  
 pasteEditMenuItem.addActionListener((e) -> {  
 textArea.paste();  
 });  
 selectAllEditMenuItem.addActionListener((e) -> {  
 textArea.selectAll();  
 });  
 editMenu.add(cutEditMenuItem);  
 editMenu.add(copyEditMenuItem);  
 editMenu.add(pasteEditMenuItem);  
 editMenu.add(selectAllEditMenuItem);  
  
 JMenu formatMenu = new JMenu("Format");  
 JMenuItem colorFormatMenuItem = new JMenuItem("Color");  
 JMenuItem fontFormatMenuItem = new JMenuItem("Font");  
 colorFormatMenuItem.addActionListener((e) -> {  
 Color color = JColorChooser.showDialog(null, "Choose a color", Color.black);  
 textArea.setForeground(color);  
 });  
 fontFormatMenuItem.addActionListener((e) -> {  
 AboutDialog aboutDialog = new AboutDialog(this);  
 aboutDialog.setVisible(true);  
 });  
 formatMenu.add(colorFormatMenuItem);  
 formatMenu.add(fontFormatMenuItem);  
  
 JMenuItem closeMenu = new JMenuItem("Close");  
 closeMenu.addActionListener(e -> {  
 if (isSaved || textArea.getText().equals("")) {  
 System.exit(0);  
 } else {  
 saveAs();  
 }  
 });  
  
 menuBar.add(fileMenu);  
 menuBar.add(editMenu);  
 menuBar.add(formatMenu);  
 menuBar.add(closeMenu);  
  
 textArea = new JTextArea();  
 textArea.setLineWrap(true);  
 textArea.setWrapStyleWord(true);  
 setJMenuBar(menuBar);  
 setTitle("New Document");  
 add(textArea);  
 setSize(500, 500);  
 setLocationRelativeTo(null);  
 setVisible(true);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 }  
  
 private void openFile() {  
 JFileChooser jFileChooser = new JFileChooser(APP\_PATH);  
 // add filters  
 FileNameExtensionFilter xmlFilter = new FileNameExtensionFilter("xml files (\*.xml)", "xml");  
 jFileChooser.addChoosableFileFilter(xmlFilter);  
 jFileChooser.setFileFilter(xmlFilter);  
  
 int r = jFileChooser.showOpenDialog(null);  
 if (r == JFileChooser.APPROVE\_OPTION) {  
 currentFileLocation = jFileChooser.getSelectedFile().getAbsolutePath();  
 this.setTitle(currentFileLocation);  
 File file = new File(currentFileLocation);  
 try {  
 String s1 = "", sl = "";  
 FileReader fileReader = new FileReader(file);  
 BufferedReader br = new BufferedReader(fileReader);  
 sl = br.readLine();  
 s1 = br.readLine();  
 while (s1 != null) {  
 System.out.println(sl);  
 sl = sl + "\n" + s1;  
 s1 = br.readLine();  
 }  
 br.close();  
 textArea.setText(sl);  
 } catch (Exception evt) {  
 JOptionPane.showMessageDialog(this, evt.getMessage());  
 }  
 }  
 }  
  
 private void saveAs() {  
 JFileChooser jFileChooser = new JFileChooser(APP\_PATH);  
  
 // add filters  
 FileNameExtensionFilter xmlFilter = new FileNameExtensionFilter("xml files (\*.xml)", "xml");  
 jFileChooser.addChoosableFileFilter(xmlFilter);  
 jFileChooser.setFileFilter(xmlFilter);  
  
 int r = jFileChooser.showSaveDialog(null);  
 if (r == JFileChooser.APPROVE\_OPTION) {  
  
 currentFileLocation = jFileChooser.getSelectedFile().getAbsolutePath();  
 if (!currentFileLocation.endsWith(".xml")) {  
 currentFileLocation += ".xml";  
 }  
  
 this.setTitle(currentFileLocation);  
 File fi = new File(currentFileLocation);  
 try {  
 FileWriter wr = new FileWriter(fi, false);  
 BufferedWriter w = new BufferedWriter(wr);  
 w.write(textArea.getText());  
 w.flush();  
 w.close();  
 this.isSaved = true;  
 } catch (Exception evt) {  
 JOptionPane.showMessageDialog(this, evt.getMessage());  
 }  
 }  
 }  
  
 private void save() {  
 File fi = new File(currentFileLocation);  
 try {  
 FileWriter wr = new FileWriter(fi, false);  
 BufferedWriter w = new BufferedWriter(wr);  
 w.write(textArea.getText());  
 w.flush();  
 w.close();  
 this.isSaved = true;  
 } catch (Exception evt) {  
 JOptionPane.showMessageDialog(this, evt.getMessage());  
 }  
 }

**Часть 2**

**Задача 1.**

Во входном текстовом файле хранится числовой массив объёмом до 1 млн значений. Сами значения являются целыми положительными и не превосходят 1000. Возможны повторяющиеся значения. В выходной текстовый файл нужно вывести эти числа в порядке возрастания.

Структура входного файла:

первая стпроекта.рока в файле показывает длину числового массива (от 1 до 1000000).

Далее идет указанное количество числовых значений от 1 до 1000 каждое по одному числу в строке.

Решение должно быть выполнено в виде консольного приложения, результаты выведены в текстовый файл.

Время работы программы не должно превышать 2 секунд.

**Задача 2.**

Модифицируйте программу, чтобы она выводила данные в порядке убывания

**Задача 3.**

Модифицируйте программу, чтобы она могла по требованию пользователя выводить данные либо по возрастанию, либо по убыванию. Информация о требуемом виде сортировки должна содержаться в том же входном файле.

**Задача 4.**

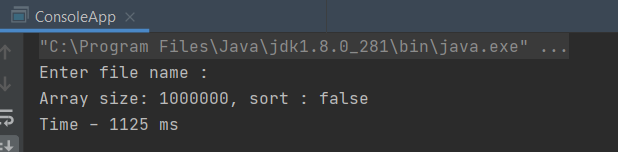
Разработайте программу проверки правильности сортировки

**Код программы:**

public class ConsoleApp {  
  
 private static Integer ARRAY\_SIZE = 1000000;  
 private static final Integer MIN\_VALUE = 0;  
 private static final Integer MAX\_VALUE = 1000;  
 private static boolean sort = false;  
  
 static final String FILE\_BASE = "/home/anastasiya/university/CCP/lab2\_3/resources/";  
 private static String inputFileName = FILE\_BASE + "numbers.txt";  
 private static final String outputFileName = FILE\_BASE + "sorted\_numbers.txt";  
  
 public static void main(String[] args) throws IOException {  
  
 System.out.print("Enter file name : ");  
 Scanner scanner = new Scanner(System.in);  
 String filename = scanner.nextLine();  
  
 doSort(filename, outputFileName);  
  
 }  
  
 public static void doSort(String filename, String outputFileName) throws IOException {  
  
 if (filename == null || filename.isEmpty()) {  
 List<Integer> generatedInts = generateInts();  
 recordIntsInFile(generatedInts, inputFileName);  
 } else {  
 inputFileName = FILE\_BASE + filename;  
 }  
  
 long from = System.nanoTime();  
 List<Integer> ints = readIntsFromFile();  
  
 List<Integer> sortedInts;  
 if (sort) {  
 sortedInts = sortUp(ints);  
 } else {  
 sortedInts = sortDown(ints);  
 }  
  
 recordIntsInFile(sortedInts, outputFileName);  
 long to = System.nanoTime();  
 System.out.println("Time - " + (to - from) / 1\_000\_000L + " ms");  
 }  
  
 private static List<Integer> sortUp(List<Integer> ints) {  
 List<Integer> sortedInts = ints  
 .stream()  
 .sorted()  
 .collect(Collectors.toList());  
  
 return sortedInts;  
 }  
  
 private static List<Integer> sortDown(List<Integer> ints) {  
 List<Integer> sortedInts = ints  
 .stream()  
 .sorted(Comparator.reverseOrder())  
 .collect(Collectors.toList());  
  
 return sortedInts;  
 }  
  
 private static List<Integer> readIntsFromFile() throws FileNotFoundException {  
 LinkedList<Integer> list = new LinkedList<>();  
 Scanner scanner = new Scanner(new File(inputFileName));  
 String[] arrSize = scanner.nextLine().split(" ");  
 System.out.println("Array size: " + arrSize[0] + ", sort : " + arrSize[1]);  
 ARRAY\_SIZE = Integer.parseInt(arrSize[0]);  
 sort = Boolean.parseBoolean(arrSize[1]);  
 while (scanner.hasNextInt()) {  
 list.addLast(scanner.nextInt());  
 }  
 return list;  
 }  
  
 private static List<Integer> generateInts() {  
  
 List<Integer> ints = new ArrayList<>();  
 for (int i = 0; i < ARRAY\_SIZE; i++) {  
 int randomInt = (int) Math.floor(Math.random() \* (MAX\_VALUE - MIN\_VALUE + 1) + MIN\_VALUE);  
 ints.add(randomInt);  
 }  
 return ints;  
 }  
  
 private static void recordIntsInFile(List<Integer> arr, String fileName) throws IOException {  
 FileWriter writer = new FileWriter(fileName);  
 int len = arr.size();  
 writer.write(len + " " + sort + "\n");  
 for (int i = 0; i < len; i++) {  
 writer.write(arr.get(i) + "\n");  
 }  
 writer.close();  
 }  
}

Тестирование:

public class ConsoleAppTest {  
  
 private static final String outputFileName = ConsoleApp.FILE\_BASE + "actual.txt";  
  
 public static void main(String[] args) throws IOException {  
  
 String filename = "test1.txt";  
  
 ConsoleApp.doSort(filename, outputFileName);  
  
 Path path1 = Paths.get(ConsoleApp.FILE\_BASE + "expected.txt");  
 Path path2 = Paths.get(outputFileName);  
 long result = filesCompareByLine(path1, path2);  
  
 if (result == 0) {  
 System.out.println("Test passed!");  
 } else {  
 System.err.println("Test failed!\n Expected and Actual files are NOT identical! Line : " + result);  
 }  
 }  
  
 public static long filesCompareByLine(Path path1, Path path2) throws IOException {  
 try (BufferedReader bf1 = Files.newBufferedReader(path1);  
 BufferedReader bf2 = Files.newBufferedReader(path2)) {  
  
 long lineNumber = 1;  
 String line1 = "", line2 = "";  
 while ((line1 = bf1.readLine()) != null) {  
 line2 = bf2.readLine();  
 if (line2 == null || !line1.equals(line2)) {  
 return lineNumber;  
 }  
 lineNumber++;  
 }  
 if (bf2.readLine() == null) {  
 return 0;  
 } else {  
 return lineNumber;  
 }  
 }  
 }  
}



**Вывод:** научились конструировать меню; освоили использование объектов класса и стандартных диалогов; освоили создание консольных приложений; освоили работу с файлами; научились строить план тестирования приложения.