**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

до лабораторної роботи №2

**на тему:** *“Робота з текстовою інформацією”*

**з дисципліни** *“Кросплатформне програмування”*

**Лектор:**

доц. каф. ПЗ

Дяконюк Л. М.

**Виконав:**

ст. гр. ПЗ-33

Юшкевич. А.І.

**Прийняв:**

доц. каф. ПЗ

Баштовий А. В.

«\_\_» \_\_\_\_\_\_\_\_\_ 2024р.

∑=\_\_\_\_\_

Львів – 2024

**Тема.** Робота з текстовою інформацією.

**Мета.** Навчитися використовувати основний інструментарій мови для пошуку, зміни, форматування текстової інформації, зокрема здобути навички роботи з датами та часовими змінними в різних часових зонах.

**Теоретичні відомості**

Важливе місце в обробці інформації займає робота з текстами. Як і багато чого, текстові рядки в мові Java являються об’єктами. Вони представляються екземплярами класу *String* або класу *StringBuffer*.

Класи *String, StringBuffer, StringBuilder* визначені в пакеті java.lang і доступні автоматично без оголошення імпорту. Всі три класи реалізують інтерфейс CharSequence.

На відміну від *String*, об'єкти типу *StringBuffer і StringBuilder* можуть бути змінені, вставляючи і додаючи рядки і символи. В багатьох випадках, коли треба змінити довжину рядка типу *String,* компілятор Java неявно перетворює його до типу *StringBuffer*, змінює довжину, потім перетворює назад в тип String.

**Конструктори класу String**

1. String() – створюється об’єкт з пустим рядком;
2. String (String str) – із одного об’єкта створюється інший, тому цей конструктор використовується рідко;
3. String (StringBuffer str) – перетворена копія об’єкту класу BufferString;
4. String(byte[] byteArray) – об’єкт створюється із масиву байтів byteArray;
5. String (char [] charArray) – об’єкт створюється із масиву charArray символів Unicode;
6. String (byte [] byteArray, int offset, int count) – об’єкт створюється із частини масиву байтів byteArray, починається з індексу offset і містить count байтів;
7. String (char [] charArray, int offset, int count) – те ж саме, але масив складається із символів Unicode;
8. String(byte[] byteArray, String encoding) – символи, записані в масиві байтів, задаються в Unicode-рядку з врахування кодування encoding;
9. String(byte[] byteArray, int offset, int count, String encoding) – те ж саме, але тільки для частини масиву.

**Завдання**

1. **Варіант 1:** Розробіть програму, яка обчислює і виводить кількість робочих годин між двома датами з урахуванням робочого графіка. Робочий графік включає в себе робочі дні та неробочі дні, і на кожний робочий день задається робочий час (наприклад, 9:00 - 17:00).

**Додаткові умови:**

1. Робочий графік може включати робочі дні та неробочі дні для кожного місяця окремо.
2. Враховуйте, що робочі дні можуть відрізнятися від країни до країни та від компанії до компанії. Наприклад, в певних країнах неробочими днями можуть бути святкові дні або вихідні дні, які відзначаються в інший час.
3. **Варіант 1.** Знайти в тексті правильні записи цілих, дійсних з плаваючою крапкою та в експоненціальній формі чисел

**Хід роботи**

***GUI.java***

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.time.\*;  
import java.time.format.DateTimeFormatter;  
import java.util.HashMap;  
import java.util.Map;  
  
public class GUI extends JFrame {  
  
 private JTextField startDateField;  
 private JTextField endDateField;  
 private JTextField[] workStartFields;  
 private JTextField[] workEndFields;  
 private Map<JCheckBox, Integer> includeDayCheckBoxes;  
 private JTextArea resultArea;  
  
 // Робочий графік (день тижня -> [початок роботи, кінець роботи])  
 private Map<DayOfWeek, LocalTime[]> workSchedule;  
  
 public GUI() {  
 setTitle("Калькулятор робочих годин");  
 setSize(500, 500);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setLayout(new BorderLayout());  
  
 // Панель для введення дат  
 JPanel datePanel = new JPanel();  
 datePanel.setLayout(new GridLayout(2, 2));  
 datePanel.add(new JLabel("Початкова дата (yyyy-MM-dd):"));  
 startDateField = new JTextField();  
 datePanel.add(startDateField);  
 datePanel.add(new JLabel("Кінцева дата (yyyy-MM-dd):"));  
 endDateField = new JTextField();  
 datePanel.add(endDateField);  
  
 add(datePanel, BorderLayout.*NORTH*);  
  
 // Панель для введення робочих годин для кожного дня тижня  
 JPanel schedulePanel = new JPanel();  
 schedulePanel.setLayout(new GridLayout(7, 4));  
 workStartFields = new JTextField[7];  
 workEndFields = new JTextField[7];  
 includeDayCheckBoxes = new HashMap<>();  
  
 DayOfWeek[] daysOfWeek = DayOfWeek.*values*();  
 for (int i = 0; i < 7; i++) {  
 schedulePanel.add(new JLabel(daysOfWeek[i].toString()));  
 workStartFields[i] = new JTextField("09:00");  
 workEndFields[i] = new JTextField("17:00");  
  
 JCheckBox checkBox = new JCheckBox();  
 checkBox.setSelected(true);  
 checkBox.addActionListener(new DayIsCheckedListener());  
 includeDayCheckBoxes.put(checkBox, i);  
  
 schedulePanel.add(workStartFields[i]);  
 schedulePanel.add(workEndFields[i]);  
 schedulePanel.add(checkBox);  
 }  
  
 add(schedulePanel, BorderLayout.*CENTER*);  
  
 JPanel bottomPanel = new JPanel();  
 bottomPanel.setLayout(new GridLayout(2, 1));  
  
 JButton openExceptionDaysButton = new JButton("Святкові дні");  
 openExceptionDaysButton.addActionListener(new OpenHolidaysListener());  
 bottomPanel.add(openExceptionDaysButton);  
  
  
 // Кнопка для обчислення  
 JButton calculateButton = new JButton("Обчислити");  
 calculateButton.addActionListener(new CalculateListener());  
 bottomPanel.add(calculateButton);  
  
 // Панель для виведення результатів  
 resultArea = new JTextArea();  
 bottomPanel.add(resultArea);  
 add(bottomPanel, BorderLayout.*SOUTH*);  
 }  
  
 private class OpenHolidaysListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 HolidaysUI holidaysUI = new HolidaysUI();  
 }  
 }  
  
 private class DayIsCheckedListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JCheckBox checkBox = (JCheckBox) e.getSource();  
 int index = includeDayCheckBoxes.get(checkBox);  
 if(checkBox.isSelected()){  
 workStartFields[index].setEnabled(true);  
 workEndFields[index].setEnabled(true);  
 }else{  
 workStartFields[index].setEnabled(false);  
 workEndFields[index].setEnabled(false);  
 }  
 }  
 }  
 private class CalculateListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try {  
 // Зчитуємо введені дати  
 LocalDate startDate = LocalDate.*parse*(startDateField.getText(), DateTimeFormatter.*ISO\_LOCAL\_DATE*);  
 LocalDate endDate = LocalDate.*parse*(endDateField.getText(), DateTimeFormatter.*ISO\_LOCAL\_DATE*);  
  
 // Створюємо графік роботи  
 workSchedule = new HashMap<>();  
 DayOfWeek[] daysOfWeek = DayOfWeek.*values*();  
  
 for (int i = 0; i < 7; i++) {  
 if(!workStartFields[i].isEnabled()) continue;  
  
 LocalTime startTime = LocalTime.*parse*(workStartFields[i].getText());  
 LocalTime endTime = LocalTime.*parse*(workEndFields[i].getText());  
 workSchedule.put(daysOfWeek[i], new LocalTime[]{startTime, endTime});  
 }  
  
 // Обчислюємо кількість робочих годин  
 long totalWorkHours = WorkHoursCalculator.*calculateWorkHours*(startDate, endDate, workSchedule);  
  
 // Виводимо результат  
 resultArea.setText("Загальна кількість робочих годин: " + totalWorkHours);  
  
 } catch (Exception ex) {  
 resultArea.setText("Помилка: " + ex.getMessage());  
 }  
 }  
 }  
}

***HolidaysUI.java***

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class HolidaysUI extends JFrame {  
  
 private String datesFile = "Dates.txt";  
 private JTextArea datesText = new JTextArea();  
 public HolidaysUI() {  
  
 setTitle("Святкові дні");  
 setSize(500, 500);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 setLayout(new BorderLayout());  
  
 datesText = new JTextArea(GetDatesFromFileAsString());  
  
 JScrollPane datesScrollPane = new JScrollPane(datesText);  
 datesScrollPane.setVerticalScrollBarPolicy(JScrollPane.*VERTICAL\_SCROLLBAR\_ALWAYS*);  
  
  
 JPanel textPanel = new JPanel();  
 textPanel.setLayout(new BorderLayout());  
 textPanel.add(datesScrollPane);  
 add(textPanel, BorderLayout.*CENTER*);  
  
 JPanel buttonsPanel = new JPanel();  
 buttonsPanel.setLayout(new GridLayout(1, 2));  
  
 JButton save = new JButton();  
 save.setText("Save");  
 save.addActionListener(new SaveButtonClickedListener());  
 buttonsPanel.add(save);  
  
 JButton cancel = new JButton();  
 cancel.setText("Cancel");  
 cancel.addActionListener(new CancelButtonClickedListener());  
 buttonsPanel.add(cancel);  
  
 add(buttonsPanel, BorderLayout.*SOUTH*);  
  
 setVisible(true);  
 }  
  
 private class SaveButtonClickedListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 FileHandler fileHandler = new FileHandler(datesFile);  
  
 try{  
 fileHandler.writeFile(datesText.getText());  
 }catch(IOException ioe){  
 System.*out*.println(ioe);  
 }  
  
 dispose();  
 }  
 }  
  
 private class CancelButtonClickedListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 dispose();  
 }  
 }  
  
 private String GetDatesFromFileAsString() {  
 FileHandler fileHandler = new FileHandler(datesFile);  
 String dates = "";  
  
 try {  
 dates = fileHandler.readFile();  
 }catch (IOException e) {  
 dates = e.toString();  
 }  
  
 return dates;  
 }  
  
  
}

***WorkHoursCalculator.java***

import java.io.IOException;  
import java.time.DayOfWeek;  
import java.time.Duration;  
import java.time.LocalDate;  
import java.time.LocalTime;  
import java.util.HashSet;  
import java.util.Map;  
import java.util.Set;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class WorkHoursCalculator {  
  
  
 public static long calculateWorkHours(LocalDate startDate, LocalDate endDate, Map<DayOfWeek, LocalTime[]> workSchedule) {  
 long totalHours = 0;  
  
 // Проходимо по кожному дню  
 for (LocalDate date = startDate; !date.isAfter(endDate); date = date.plusDays(1)) {  
 DayOfWeek dayOfWeek = date.getDayOfWeek();  
  
 Set<LocalDate> holidays = *GetDatesFromFile*();  
  
 // Якщо це робочий день, додаємо години  
 if (workSchedule.containsKey(dayOfWeek) && !holidays.contains(date)) {  
 LocalTime[] workTimes = workSchedule.get(dayOfWeek);  
 if (workTimes != null) {  
 Duration workDuration = Duration.*between*(workTimes[0], workTimes[1]);  
 totalHours += workDuration.toHours();  
 }  
 }  
 }  
  
 return totalHours;  
 }  
  
 private static Set<LocalDate> GetDatesFromFile(){  
 Set<LocalDate> dates = new HashSet<>();  
  
 FileHandler handler = new FileHandler("Dates.txt");  
  
 String text;  
 try{  
 text = handler.readFile();  
 }catch(IOException e){  
 System.*out*.println("Error in reading file");  
 return dates;  
 }  
  
 String regex = "([1-2]\\d{3})\\.(0[1-9]|1[0-2])\\.([0-2][0-9]|3[0-1])";  
 Pattern pattern = Pattern.*compile*(regex);  
 Matcher matcher = pattern.matcher(text);  
  
 while (matcher.find()) {  
 String temp = matcher.group().replace('.', '-');  
 dates.add(LocalDate.*parse*(temp));  
 }  
  
 return dates;  
 }  
}

***FileHandler.java***

import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.nio.CharBuffer;  
  
public class FileHandler {  
 private String fileName;  
  
 public String getFileName() {return fileName;}  
 public void setFileName(String fileName) {this.fileName = fileName;}  
 public FileHandler(String fileName) {  
 this.fileName = fileName;  
 }  
  
 public String readFile() throws IOException {  
  
 FileReader fr = null;  
  
 do{  
 try {  
 fr = new FileReader(fileName);  
 } catch (FileNotFoundException e) {  
 writeFile(fileName);  
 }  
 }while (fr == null);  
  
 int c;  
 String data = new String();  
  
 while ((c = fr.read()) != -1) {  
 data += (char) c;  
 }  
 fr.close();  
  
 return data;  
 }  
  
 public void writeFile(String data) throws IOException {  
 FileWriter fw = new FileWriter(fileName);  
  
 for(int i = 0; i < data.length(); i++){  
 fw.write(data.charAt(i));  
 }  
  
 fw.close();  
 }  
}

***Main.java***

import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 GUI calculator = new GUI();  
 calculator.setVisible(true);  
 });  
 }  
}

Додаю [посилання](https://github.com/Akmitliviy/CPP) на GitHub репозиторій.

**Висновки:** протягом виконання цієї лабораторної роботи я навчився використовувати основний інструментарій мови для пошуку, зміни, форматування текстової інформації.