**Лабораторная работа № 4**

**Текстовые файлы и исключения**

Дан текстовый файл, каждая строка которого имеет предложенный ниже формат. Необходимо вывести его данные в виде таблицы, и выполнить указанную обработку. Имя файла задается пользователем. Если выбранный файл имеет иной формат, то сообщить об этом, используя механизм исключений.

В качестве интерфейса для ввода исходных данных и вывода результатов использовать web-интерфейс организованный с помощью фреймворка Vaadin. Для выбора операции организовать вертикальное меню, для расположения компонентов использовать GridLayout, у страницы должен быть заголовок, например, «Лабораторная работа №4» (стиль заголовка выбрать самостоятельно, заголовок поместить в одну из ячеек GridLayout). GridLayout не поддерживается в последних версия, можно использовать любой другой Grid.

Данные из фала считать двумя способами:

1. стандартное чтение из потока с помощью классов FileReader или Scanner;
2. чтение из потока в функциональном стиле, используя метод lines() класса Files.

Сообщение об исключительной ситуации также отразить на странице.

Ниже приведен небольшой пример создания таблицы с помощью списка объектов Person и вывода некоторой информации о таблице.

package mypack;  
  
public class Person {  
 String name;  
 String profession;  
 String age;  
 public Person(String name, String profession, String age) {  
 this.name = name;  
 this.profession = profession;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getProfession() {  
 return profession;  
 }  
  
 public String getAge() {  
 return age;  
 }  
  
}

package org.example;  
  
  
import java.lang.\*;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.Paragraph;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.tabs.Tab;  
import com.vaadin.flow.component.tabs.Tabs;  
import com.vaadin.flow.router.Route;  
import mypack.Person;  
  
@Route  
public class MainView extends HorizontalLayout {  
  
  
// Создаем таблицу, заполняем её данными из списка  
 class MyGrid{  
 private Grid<Person> grid = new Grid<>(Person.class, false);  
 private List<Person> lst=new ArrayList<>();  
 MyGrid(){  
 grid.addColumn(Person::getName).setHeader("Фамилия").setAutoWidth(true);  
 grid.addColumn(Person::getProfession).setHeader("Профессия").setAutoWidth(true);  
 grid.addColumn(Person::getAge).setHeader("Возраст").setAutoWidth(true);  
 lst.add(new Person("Иванов", "врач", "45"));  
 lst.add(new Person("Петров", "водитель", "25"));  
 lst.add(new Person("Сидоров", "учитель", "47"));  
 lst.add(new Person("Семёнов", "продавец", "35"));  
 lst.add(new Person("Крокодилов", "директор", "55"));  
 grid.setItems(lst);  
 grid.setAllRowsVisible(true);  
 }  
  
 public Grid<Person> getGrid() {  
 return grid;  
 }  
  
 public String toString(){  
 return Integer.*valueOf*(lst.size()).toString();  
 }  
  
  
 }  
 public MainView() {

// Создаем вертикальное меню, связываем каждый пункт меню с компонентом

Map<Tab, Component> tabsToComponents = new HashMap<>();  
 Tab t1 = new Tab("Таблица");  
 MyGrid gr=new MyGrid();  
 tabsToComponents.put(t1, gr.getGrid());  
 Paragraph label = new Paragraph("В таблице "+gr+" строк");  
 Tab t2 = new Tab("Кое-что о таблице");  
 tabsToComponents.put(t2, label);  
  
 Tabs tabs = new Tabs(t1, t2);  
 tabs.setOrientation(Tabs.Orientation.*VERTICAL*);  
 tabs.setHeight("240px");  
 tabs.setWidth("240px");

// Обрабатываем событие выбор пункта меню  
 tabs.addSelectedChangeListener(event -> {  
 tabsToComponents.values().forEach(component -> component.setVisible(false));  
 tabsToComponents.get(tabs.getSelectedTab()).setVisible(true);  
 });  
 tabsToComponents.values().forEach(component -> component.setVisible(false));  
 tabsToComponents.get(t1).setVisible(true);

// Итоговое добавление компонентов на страницу

Div pages = new Div(gr.getGrid(), label);  
 this.add(tabs, pages);  
  
 }

}

**Варианты заданий**

|  |
| --- |
| 1. Библиотека. Подсчитать общее количество книг в библиотеке и количество книг данного автора. Формат строки: Автор (строка из 20 символов) название (строка из 20 символов) количество экземпляров (целое число). |
| 2. Статистика. Вывести фамилию самого высокого мужчины из списка и фамилию женщины, рост которой наиболее близок к среднему. Формат строки: Фамилия (строка из 20 символов) пол (м или ж) рост (целое число) |
| 3. Зарплата. Найти самого высокооплачиваемого работника (находить по средней зарплате за день), а также по запросу фамилий двух работников выяснить, кто из них заработал больше и на сколько. Формат строки: Фамилия (строка из 20 символов) количество отработанных дней (целое) зарплата (действительное число). |
| 4. Склад. Найти общую стоимость всего товара на складе, самый дешевый товар, а также по запросу наименования товара – его количество на складе (один и тот же товар может быть записан несколько раз). Формат строки: Наименование товара (строка из 20 символов) количество (целое) цена (действительное число) |
| 5. Олимпиада. Найти страну с наибольшим количеством золотых медалей, а также по запросу названий двух стран, какая из них завоевала больше очков и на сколько. Формат строки: Страна (строка из 20 символов) кол-во золотых (целое) серебряных (целое) бронзовых (целое) |
| 6. Автомобили. Определить, какая из машин имеет наименьший пробег, а также по запросу фамилий двух владельцев, чей автомобиль «моложе». Формат строки: Фамилия (строка из 20 символов) год выпуска (целое) пробег (целое число) |
| 7. Международный товарооборот. Найти страну с наибольшим внешним долгом на одного жителя, а также по названиям 2-х государств выяснить, у которой из них внешний долг больше и на сколько. Формат строки: Государство (строка из 20 символов) число жителей (целое) долг (целое число) |
| 8. Поликлиника. Подсчитать общее количество различных заболеваний и общее количество обращений данного больного. Формат строки: Фамилия (строка из 20 символов) заболевание (строка из 20 символов) количество обращений (целое число). |
| 9. Школа. Найти суммарную нагрузку всех учителей школы, а также по названию предмета, определить, кто из учителей его преподает. Формат строки: Фамилия учителя (строка из 20 символов) предмет (строка из 20 символов) количество часов в неделю (целое число). |
| 10. Железная дорога. Найти станцию с наибольшим количеством проходящих через нее поездов, а также по запросу названий двух станций, через какую из них проходит больше пассажирских поездов. Формат строки:  Станция (строка из 20 символов) количество пассажирских (целое число) и товарных (целое число) поездов, проходящих через станцию за сутки. |
| 11. Армия. Выяснить, какое количество контрактов заканчивается в этом году, а также по запросу двух фамилий, кому из военнослужащих осталось служить дольше.  Формат строки: Фамилия (строка из 20 символов) год начала службы (целое число) срок контракта (целое число). |
| 12. Студенты. Выяснить, какое количество студентов должны получать стипендию, а также по запросу фамилий двух студентов, у кого из них выше средний балл.  Формат строки: Фамилия (строка из 20 символов) оценка за 1-й экзамен (целое число) оценка за 2-й экзамен (целое число) оценка за 3-й экзамен (целое число). |

5. Олимпиада. Найти страну с наибольшим количеством золотых медалей, а также по запросу названий двух стран, какая из них завоевала больше очков и на сколько.  
Формат строки:  
Страна (строка из 20 символов) кол-во золотых (целое) серебряных (целое) бронзовых (целое)

Класс Country:

package org.example;  
  
import java.util.Objects;  
  
public class Country {  
 private String name;  
 private int goldMedals;  
 private int silverMedals;  
 private int bronzeMedals;  
  
 public Country(String name, int goldMedals, int silverMedals, int bronzeMedals) {  
 this.name = name;  
 this.goldMedals = goldMedals;  
 this.silverMedals = silverMedals;  
 this.bronzeMedals = bronzeMedals;  
 }  
  
 // Геттеры  
 public String getName() {  
 return name;  
 }  
  
 public int getGoldMedals() {  
 return goldMedals;  
 }  
  
 public int getSilverMedals() {  
 return silverMedals;  
 }  
  
 public int getBronzeMedals() {  
 return bronzeMedals;  
 }  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Country country)) return false;  
 return Objects.*equals*(name, country.name);  
 }  
}

MainView:

package org.example;  
  
import com.vaadin.flow.component.Text;  
import com.vaadin.flow.component.button.Button;  
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.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.radiobutton.RadioButtonGroup;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;

@Route  
public class MainView extends VerticalLayout {  
  
 private Grid<Country> grid = new Grid<>(Country.class, false);  
 private ArrayList<Country> countries = new ArrayList<>();  
 public MainView() {  
 setupHeader("Лабораторная работа №4");  
 setupUploadComponents();  
 setupGrid();  
 getChampion();  
 comparison();  
 }  
 private void setupHeader(String title) {  
 // Здесь можно настроить стиль заголовка  
 add(title);  
 }  
 private void setupUploadComponents() {  
 RadioButtonGroup<String> radioGroup = new RadioButtonGroup<>();  
 radioGroup.setLabel("Выберите способ чтения файла:");  
 radioGroup.setItems("Стандартное чтение", "Чтение в функциональном стиле");  
 add(radioGroup);  
 com.vaadin.flow.component.textfield.TextField textField = new com.vaadin.flow.component.textfield.TextField();  
 Button button = new Button("Загрузить файл", event -> {  
 grid.setItems(Collections.*emptyList*());  
 countries.clear();  
 String filename=textField.getValue();  
 if (filename.length()==0){  
 textField.setValue("Введите имя файла!");  
 }  
 else {  
 String selectedOption = radioGroup.getValue();  
 if (selectedOption=="Стандартное чтение") loadFileUsingFileReader(filename);  
 else loadFileUsingFiles(filename);  
 }  
  
 });  
 add(textField);  
 add(button);  
 }  
 private void setupGrid() {  
 grid.setColumns("name", "goldMedals", "silverMedals", "bronzeMedals");  
 add(grid);  
 }  
 private void loadFileUsingFileReader(String fileName) {  
  
 try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {  
 String line;  
 while ((line = br.readLine()) != null) {  
 String[] parts = line.split(" ");  
 if (parts.length != 4) {  
 throw new IllegalArgumentException("Неверный формат файла");  
 }  
 String countryName = parts[0];  
 int gold = Integer.*parseInt*(parts[1]);  
 int silver = Integer.*parseInt*(parts[2]);  
 int bronze = Integer.*parseInt*(parts[3]);  
 countries.add(new Country(countryName, gold, silver, bronze));  
 }  
 grid.setItems(countries);  
 } catch (IOException e) {  
 Notification.*show*("Ошибка чтения файла: " + e.getMessage());  
 } catch (IllegalArgumentException e) {  
 Notification.*show*(e.getMessage());  
 }  
 }  
 private void loadFileUsingFiles(String fileName) {  
 try {  
 Files.*lines*(Path.*of*(fileName)).forEach(line -> {  
 String[] parts = line.split(" ");  
 if (parts.length != 4) {  
 throw new IllegalArgumentException("Неверный формат файла");  
 }  
 String countryName = parts[0];  
 int gold = Integer.*parseInt*(parts[1]);  
 int silver = Integer.*parseInt*(parts[2]);  
 int bronze = Integer.*parseInt*(parts[3]);  
 countries.add(new Country(countryName, gold, silver, bronze));  
 });  
 grid.setItems(countries);  
 } catch (IOException e) {  
 Notification.*show*("Ошибка чтения файла: " + e.getMessage());  
 } catch (IllegalArgumentException e) {  
 Notification.*show*(e.getMessage());  
 }  
 }  
  
 private void getChampion(){  
 Text text1=new Text("");  
 HorizontalLayout layout=new HorizontalLayout();  
 Button button = new Button("Больше золотых медалей:", event -> {  
 Collections.*sort*(countries, Comparator.*comparingInt*(Country::getGoldMedals).reversed());  
 text1.setText("Наибольшее количество золотых медалей получила: "+ countries.get(0).getName());  
  
 });  
 layout.add(button,text1);  
 add(layout);  
 }  
 private void comparison(){  
 HorizontalLayout layout1=new HorizontalLayout();  
 HorizontalLayout layout2=new HorizontalLayout();  
 TextField country1=new TextField();  
 Text text2=new Text(" и ");  
 Text text3=new Text("");  
 TextField country2=new TextField();  
 Button button = new Button("Больше золотых медалей у ", event -> {  
 Country C1=new Country(country1.getValue(),0,0,0);  
 Country C2=new Country(country2.getValue(),0,0,0);  
 int gold1=countries.get(countries.indexOf(C1)).getGoldMedals();  
 int gold2=countries.get(countries.indexOf(C2)).getGoldMedals();  
 if (gold1>gold2){  
 text3.setText("Наибольшее количество золотых медалей получила: "+ country1.getValue());}  
 else if (gold1<gold2) {  
 text3.setText("Наибольшее количество золотых медалей получила: "+ country2.getValue());  
 }  
 else {  
 text3.setText("Обе страны получили равное число медалей");  
 }  
 add(text3);  
  
  
 });  
 layout1.add(country1,text2,country2);  
 add(layout1);  
 layout2.add(button,text3);  
 add(layout2);  
 }  
}