**МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

Лабораторная работа по ИТП №6

Выполнил: Студент группы

БВТ2204

Сарыков Эрбол

Москва

2023

Задание 1:

Написать программу, которая считывает текстовый файл и выводит на

экран топ-10 самых часто встречающихся слов в этом файле. Для решения

задачи использовать коллекцию Map, где ключом будет слово, а

значением - количество его повторений в файле.

Пример реализации:

import java.io.File;

import java.io.FileNotFoundException;

import java.util.\*;

public class TopWords {

public static void main(String[] args) {

// указываем путь к файлу

String filePath = "C:\\text.txt";

// создаем объект File

File file = new File(filePath);

// создаем объект Scanner для чтения файла

Scanner scanner = null;

try {

scanner = new Scanner(file);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

// создаем объект Map для хранения слов и их количества

\*\*\*\*\*\*

// читаем файл по словам и добавляем их в Map

\*\*\*\*\*\*

// закрываем Scanner

\*\*\*\*\*\*

// создаем список из элементов Map

\*\*\*\*\*\*

// сортируем список по убыванию количества повторений

Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {

@Override

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String,

Integer> o2) {

\*\*\*\*\*\*

}

});

// выводим топ-10 слов

\*\*\*\*\*\*

}

//выводим результат

\*\*\*\*\*\*\*

}

}

}

Задание 2:

Написать обобщенный класс Stack<T>, который реализует стек на основе

массива. Класс должен иметь методы push для добавления элемента в

стек, pop для удаления элемента из стека и peek для получения верхнего

элемента стека без его удаления.

Пример реализации:

public class Stack<T> {

private T[] data;

private int size;

public Stack(int capacity) {

data = (T[]) new Object[capacity];

size = 0;

}

public void push(T element) {

\*\*\*\*\*\*

}

public T pop() {

\*\*\*\*\*\*

}

public T peek() {

\*\*\*\*\*\*

}

}

Пример использования:

Stack<Integer> stack = new Stack<>(10);

stack.push(1);

stack.push(2);

stack.push(3);

System.out.println(stack.pop());

System.out.println(stack.peek());

stack.push(4);

System.out.println(stack.pop());

Задание 3:

Необходимо разработать программу для учета продаж в магазине.

Программа должна позволять добавлять проданные товары в коллекцию,

выводить список проданных товаров, а также считать общую сумму

продаж и наиболее популярный товар. Использовать TreeSet для хранения списка проданных товаров

Задание 1

import java.io.File;

import java.io.FileNotFoundException;

import java.util.\*;

public class TopWords {

    public static void main(String[] args) {

        // тут тоже путь указывай до файла свой

        String filePath = " вот тут ";

        File file = new File(filePath);

        Scanner scanner = null;

        try {

            scanner = new Scanner(file);

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        }

        Map<String, Integer> wordCount = new HashMap<>();

        while (scanner != null && scanner.hasNext()) {

            String word = scanner.next().replaceAll("\\W", "").toLowerCase();

            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);

        }

        if (scanner != null) {

            scanner.close();

        }

        List<Map.Entry<String, Integer>> list = new ArrayList<>(wordCount.entrySet());

        list.sort((o1, o2) -> o2.getValue().compareTo(o1.getValue()));

        for (int i = 0; i < Math.min(10, list.size()); i++) {

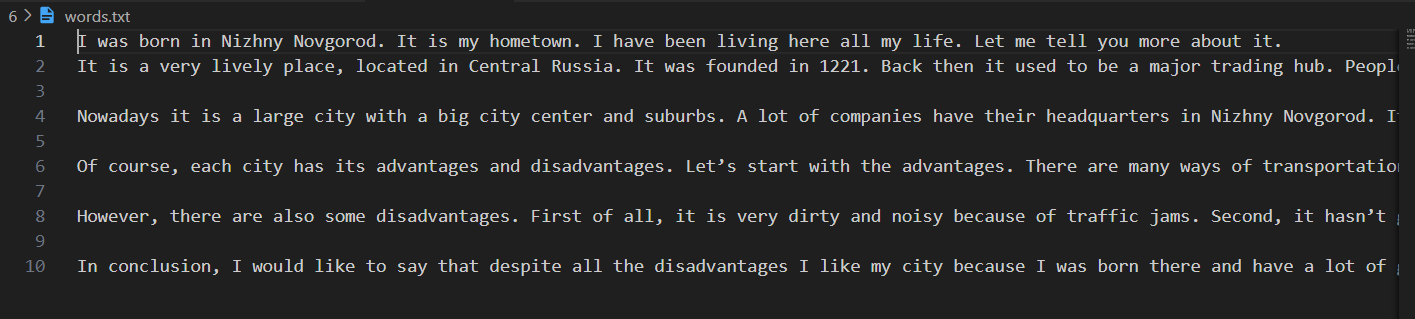
            Map.Entry<String, Integer> entry = list.get(i);

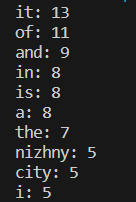
            System.out.println(entry.getKey() + ": " + entry.getValue());

        }

    }

}





1. Метод main

Метод main представляет точку входа в программу.

В этом методе определена строка filePath, которая содержит путь к текстовому файлу, который нужно прочитать и проанализировать.

Затем создается объект File, представляющий собой файл, который будет прочитан.

Создается объект Scanner для чтения данных из файла.

Далее создается пустая HashMap с именем wordCount, которая будет использоваться для хранения слов и их частоты встречаемости.

С использованием цикла while, программа считывает каждое слово из файла, выполняя следующие действия:

Удаляет все неалфавитные символы из слова с помощью метода replaceAll("\\W", "").

Преобразует слово в нижний регистр с помощью метода toLowerCase().

Добавляет слово в wordCount с использованием метода put(). Если слово уже существует в wordCount, увеличивает его частоту на 1 с помощью getOrDefault(word, 0) + 1.

После завершения чтения файла, программа закрывает Scanner.

Создается список list, который содержит записи (элементы) из wordCount. Этот список будет использоваться для сортировки записей по частоте встречаемости слов.

Список list сортируется в порядке убывания частоты встречаемости с использованием анонимного компаратора.

Затем программа выводит наиболее часто встречающиеся слова. В данном случае, выводится до 10 слов с наибольшей частотой встречаемости.

Задание 2

import java.util.EmptyStackException;

public class \_Stack\_ {

    public static void main(String[] args) {

        Stack<Integer> stack = new Stack<>(10);

        stack.push(1);

        stack.push(2);

        stack.push(3);

        System.out.println(stack.pop());

        System.out.println(stack.peek());

        stack.push(4);

        System.out.println(stack.pop());

    }

    public static class Stack<T> {

        private T[] data;

        private int size;

        @SuppressWarnings("unchecked")

        public Stack(int capacity) {

            data = (T[]) new Object[capacity];

            size = 0;

        }

        public void push(T element) {

            if (size == data.length) {

                throw new StackOverflowError("Стек заполнен");

            }

            data[size++] = element;

        }

        public T pop() {

            if (size == 0) {

                throw new EmptyStackException();

            }

            T element = data[--size];

            data[size] = null;

            return element;

        }

        public T peek() {

            if (size == 0) {

                throw new EmptyStackException();

            }

            return data[size - 1];

        }

        public boolean isEmpty() {

            return size == 0;

        }

    }

}



1. Класс Stack<T>

Этот класс представляет собой реализацию стека с использованием параметра типа T.

Он имеет следующие основные методы:

push(T element): Добавляет элемент в вершину стека. Если стек полон, генерируется исключение StackOverflowError.

pop(): Удаляет и возвращает элемент из вершины стека. Если стек пуст, генерируется исключение EmptyStackException.

peek(): Возвращает элемент из вершины стека без удаления. Если стек пуст, генерируется исключение EmptyStackException.

isEmpty(): Проверяет, пуст ли стек.

Он также имеет поля:

private T[] data: Массив для хранения элементов стека.

private int size: Текущий размер стека.

Конструктор Stack(int capacity): Создает стек с заданной вместимостью.

2. Метод main

Метод main представляет точку входа в программу.

В этом методе создается экземпляр класса Stack<Integer>, представляющий стек, который будет работать с целыми числами.

Затем в стек добавляются несколько элементов с помощью метода push().

Выводятся результаты операций pop(), peek(), и еще один элемент добавляется в стек.

Задание 3

import java.util.\*;

class Book implements Comparable<Book> {

    String title;

    double price;

    int soldCopies;

    public Book(String title, double price) {

        this.title = title;

        this.price = price;

        this.soldCopies = 0;

    }

    public void sellCopy() {

        this.soldCopies++;

    }

    public String getTitle() {

        return title;

    }

    public double getPrice() {

        return price;

    }

    public int getSoldCopies() {

        return soldCopies;

    }

    @Override

    public int compareTo(Book other) {

        return this.title.compareTo(other.title);

    }

    @Override

    public String toString() {

        return title + " (Продано копий: " + soldCopies + ", Цена: " + price + ")";

    }

}

public class BookSales {

    private TreeSet<Book> soldBooks;

    public BookSales() {

        this.soldBooks = new TreeSet<>();

    }

    public void addSale(Book book) {

        // есть ли уже такая книга в списке

        if (soldBooks.contains(book)) {

            soldBooks.forEach(b -> {

                if (b.equals(book)) {

                    b.sellCopy();

                }

            });

        } else {

            book.sellCopy();

            soldBooks.add(book);

        }

    }

    public void printSoldBooks() {

        soldBooks.forEach(System.out::println);

    }

    public double totalSalesAmount() {

        return soldBooks.stream().mapToDouble(b -> b.getPrice() \* b.getSoldCopies()).sum();

    }

    public Book mostPopularBook() {

        return Collections.max(soldBooks, Comparator.comparingInt(Book::getSoldCopies));

    }

    public static void main(String[] args) {

        BookSales sales = new BookSales();

        Book book1 = new Book("Война и мир", 500);

        Book book2 = new Book("Гарри Поттер", 400);

        sales.addSale(book1);

        sales.addSale(book2);

        sales.addSale(book1); // продали еще одну копию

        sales.printSoldBooks();

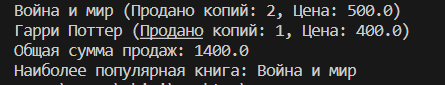
        System.out.println("Общая сумма продаж: " + sales.totalSalesAmount());

        Book popularBook = sales.mostPopularBook();

        System.out.println("Наиболее популярная книга: " + popularBook.getTitle());

    }

}



1. Класс Book

Этот класс представляет собой книгу с атрибутами:

title (название книги)

price (цена книги)

soldCopies (количество проданных копий)

Класс имеет методы:

sellCopy(): Увеличивает количество проданных копий на 1.

getTitle(): Возвращает название книги.

getPrice(): Возвращает цену книги.

getSoldCopies(): Возвращает количество проданных копий.

Реализует интерфейс Comparable, чтобы книги могли быть сравнимы по названию.

2. Класс BookSales

Этот класс представляет собой список проданных книг в виде TreeSet<Book>, который автоматически сортирует книги по названию.

Метод addSale(Book book): Добавляет продажу книги. Если книга уже есть в списке, увеличивает количество проданных копий. Если книги нет в списке, добавляет ее и устанавливает количество проданных копий в 1.

Метод printSoldBooks(): Выводит список проданных книг.

Метод totalSalesAmount(): Вычисляет общую сумму продаж всех книг.

Метод mostPopularBook(): Находит и возвращает наиболее популярную книгу, то есть книгу с наибольшим количеством проданных копий.

3. Метод main

Метод main представляет точку входа в программу.

Создается экземпляр класса BookSales для учета продаж.

Создаются две книги book1 и book2.

Вызываются методы addSale() для учета продаж книг.

Выводится список проданных книг, общая сумма продаж и информация о наиболее популярной книге.