**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №6

По дисциплине «Информационные технологии и программирование»

Выполнил: Студент группы

БВТ2203

Попов Александр

Москва

2023

**Цель работы:** знакомство с коллекциями в ЯП Java.

**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

**Задание 2:** Написать обобщенный класс Stack, 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

**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.

**Ход работы**

**Задание 1:**

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.stream.Collectors;

*public* *class* TopWords {

*public* *static* void main(String[] *args*) {

        String filePath = "text.txt";

        File file = *new* File(filePath);

        Scanner scanner = null;

*try* {

            scanner = *new* Scanner(file);

        } *catch* (FileNotFoundException *e*) {

            System.out.println("File not found!");

*return*;

        }

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

*try*{

            FileInputStream in = *new* FileInputStream(file);

            int c;

            String s = "";

*while*((in.available())!=1){

                c = in.read();

*if*((int)c!=32){

                    s = s + (char)c;

                } *else* {

*if* (!map.containsKey(s)) {

                        map.put(s, 1);

                    } *else* {

                        int count = map.get(s);

                        map.put(s, count += 1);

                    }

                    s = "";

                }

            }

            in.close();

        } *catch* (IOException *e*) {

            System.out.println("Player file wouldn't load!");

            scanner.close();

*return*;

        }

        scanner.close();

*// 1. Создаю новый отсортированный по значению hashMap*

*// 2. entrySet - возвращает set из элементов исходного map.*

*// 3. stream - поток представляет канал передачи данных из источника данных, понятие потока в программировании размыто;*

*// метод stream создает из set поток данных.*

*// 4. sorted - метод, сортирующий по возрастанию. Принимает в себя компаратор;*

*// Коллекция, которую мы хотим отсортировать, должна реализовывать интерфейс comparator.*

*// 6. Comparable делает наши объекты «сравнимыми» и создает для них наиболее естественный порядок сортировки.*

*// 7. collect - позволяет получить данные не в виде потока, а в виде коллекции.*

*// 8. Collectors.toMap() - используется для сбора элементов Stream в Map.*

*// Я использую перегрузку, которая принимает ключ, значение, функцию - разрешитель конфликтов (хотя она тут избыточна,*

*// т.к. коллекция уже отсортирована и мы используем hashmap) и создается новый LinkedHashMap.*

*// 9. Разница между HashMap и LinkedHashMap заключается в том, что HashMap не сохраняет порядок ввода ключей или элементов,*

*// тогда как LinkedHashMap следует порядку вставки ключей и сохраняет порядок элементов.*

        Map<String, Integer> sortedMap = map.entrySet().stream()

                .sorted((*e1*, *e2*) -> {

*return* *e2*.getValue() - *e1*.getValue();

                })

                .collect(Collectors.toMap(

                        Map.Entry*::*getKey,

                        Map.Entry*::*getValue,

                        (*a*, *b*) -> { *throw* *new* AssertionError(); },

                        LinkedHashMap*::new*

                    ));

        int i=0;

*for* (Map.Entry<String, Integer> entry *:* sortedMap.entrySet()) {

*if* (i>=10) *break*;

i++;

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

        }

    }

}

Файл 1. TopWords.java.

Программа в этом файле считывает слова из файла и выводить список из 10 самых встречающихся слов. Сначала я считываю файл и записываю все слова в map. Затем я, используя поток, сортирую этот словарь и превращаю его в LinkedHashMap. Затем вывожу результат в консоль.

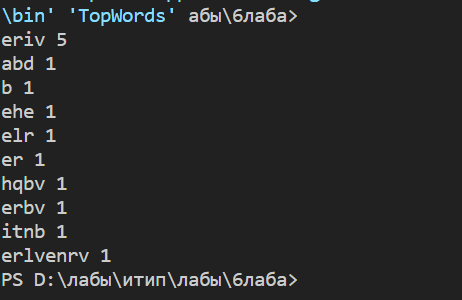


Рис. 1. Результат выполнения программы.

**Задание 2:**

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*) {

*try* {

            data[size] = *element*;

            size++;

        } *catch*(ArrayIndexOutOfBoundsException *e*) {

            System.out.println("Stack is full");

        }

    }

*public* T pop() {

*if* (size == 0) *return* null;

        T el = data[size-1];

        data[size-1] = null;

        size--;

*return* el;

    }

*public* T peek() {

*return* size == 0 *?* null *:* data[size-1];

    }

}

Файл 2. Stack.java.

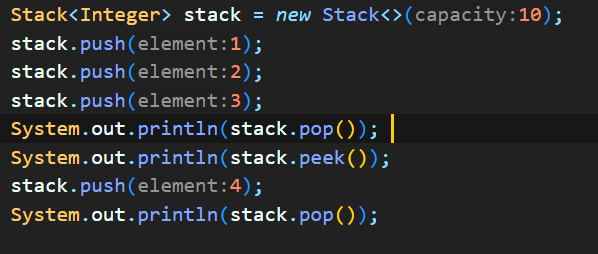
****

Рис. 2. 1-ая часть Main.java.

Я написал класс стэка, который добавляет и удаляет элементы, а также дает возможность получить последний элемент. Затем я тестирую его работу в Main.java.

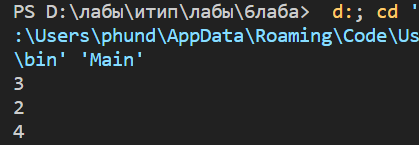
****

Рис. 3. Результат выполнения работы.

**Задание 3:**

class Product implements Comparable<Product> {

*private* String name;

*private* double price;

*private* int count;

*public* Product(String *name*, double *price*) {

        this.name = *name*;

        this.price = *price*;

        this.count = 1;

    }

*public* String getName() {

*return* this.name;

    }

*public* double getPrice() {

*return* this.price;

    }

*public* int getCount() {

*return* this.count;

    }

*public* void incrementCount() {

        this.count += 1;

    }

    @Override

*public* int compareTo(Product *otherProduct*) {

*return* this.name.compareTo(*otherProduct*.getName());

    }

}

Файл 3. Product.java

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.TreeSet;

*public* *class* Sales {

*private* TreeSet<Product> data;

*public* Sales() {

        this.data = *new* TreeSet<Product>();

    }

*public* void addProduct(String *name*, double *price*) {

*if*(!data.add(*new* Product(*name*, *price*))) {

            Iterator<Product> iterator = data.iterator();

*while*(iterator.hasNext()) {

                Product product = iterator.next();

*if*(product.getName() == *name*) {

                    product.incrementCount();

*return*;

                }

            }

        };

    }

*public* void printProducts() {

*for* (Product product *:* data) {

            System.out.println(product.getName() + " " + product.getPrice());

        }

    }

*public* double totalSales() {

        double res = 0;

*for* (Product product *:* data) {

            res += product.getPrice() \* product.getCount();

        }

*return* res;

    }

*public* ArrayList<String> getMostPopularProduct() {

*if* (data.isEmpty()) {

*return* null;

        }

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

*for* (Product product *:* data) {

            map.put(product.getName(), product.getCount());

        }

        ArrayList<String> res = *new* ArrayList<String>();

        int maxCount = 0;

*for* (Map.Entry<String, Integer> entry *:* map.entrySet()) {

*if* (entry.getValue() >= maxCount) {

*if* (entry.getValue() > maxCount) {

                    res.clear();

                    maxCount = entry.getValue();

                }

                res.add(entry.getKey());

            }

        }

*return* res;

    }

}

Файл 4. Sales.java.

Для начала я написал вспомогательный класс Product, который служит основным типом данных для 3 задания. Он содержит наименование, цену и количество проданного продукта. Также он реализовывает класс Comparable, это нужно для сравнения элементов в TreeSet.

Затем я реализовал класс Sales, в котором можно добавлять продукты, выводить список продуктов, а также считать общую сумму продаж и наиболее популярный товар.

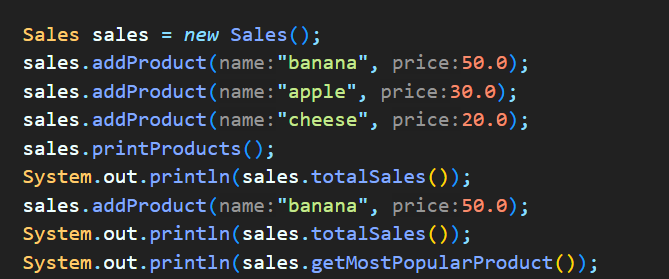
****

Рис. 4. 2-ая часть Main.java.

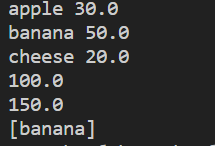
****

Рис. 5. Результат выполнения программы.

**Выводы**

1. Я научился работать с коллекциями, в частности с ArrayList, HashMap и TreeSet.

2. Я написал свой класс стэка.