**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования

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

**ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

Кафедра «МКиИТ»

**Информационные технологии и**

**программирование**

**Лабораторная работа по ИТП №6**

Выполнил: студент группы БВТ2205

Белов Егор Олегович

**Задание 1:**

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 = "src/fileEx1.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  
 Map<String, Integer> map = new HashMap<>();  
  
 // читаем файл по словам и добавляем их в Map  
 while (scanner.hasNext()) {  
 String word = scanner.next().toLowerCase();  
  
 if (map.containsKey(word)) {  
 int quantity = map.get(word);  
 quantity++;  
  
 map.put(word, quantity);  
 } else {  
 map.put(word, 1);  
 }  
 }  
  
 // закрываем Scanner  
 scanner.close();  
  
 // создаем список из элементов Map  
 List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());  
  
 // 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) {  
 // Сравнение по значению (Integer) в порядке убывания  
 return -1 \* o1.getValue().compareTo(o2.getValue());  
 }  
 });  
  
 //выводим результат  
 if (list.size() > 10) {  
 String[] result = new String[10];  
  
 for (int i = 0; i < 10; i++) {  
 result[i] = String.*valueOf*(list.get(i));  
 }  
  
 System.*out*.println(Arrays.*toString*(result));  
 } else {  
 String[] result = new String[list.size()];  
  
 for (int i = 0; i < list.size(); i++) {  
 result[i] = String.*valueOf*(list.get(i));  
 }  
  
 System.*out*.println(Arrays.*toString*(result));  
 }  
 }  
}

Создается класс TopWords, содержащий метод main, который является точкой входа программы.

Задается путь к файлу, который нужно проанализировать. В данном случае файл располагается в папке src и имеет имя fileEx1.txt.

Создается объект типа File, который представляет указанный файл.

Создается объект типа Scanner для чтения содержимого файла.

В блоке try-catch происходит попытка создания объекта Scanner с использованием файла. Если файл не найден, будет выведено сообщение об ошибке.

Создается объект типа HashMap с именем map, который будет использоваться для хранения слов и их количества.

В цикле с помощью метода while (scanner.hasNext()) происходит чтение файла по словам.

Каждое слово считывается и приводится к нижнему регистру с помощью метода toLowerCase().

Проверяется, содержится ли слово уже в map. Если да, то увеличивается количество повторений слова на 1 и обновляется значение в map. Если слово не содержится в map, оно добавляется с начальным значением 1.

После чтения всего файла, Scanner закрывается с помощью метода scanner.close().

Создается список list из элементов map с использованием конструктора ArrayList<>(map.entrySet()).

Создается компаратор с помощью анонимного класса Comparator<Map.Entry<String, Integer>>(), который сортирует элементы списка по убыванию количества повторений.

Список list сортируется с помощью метода Collections.sort() и переданным компаратором.

Выводится результат. Если в списке более 10 элементов, выбираются первые 10 элементов и помещаются в массив result. Затем массив result выводится с помощью System.out.println(). Если в списке менее 10 элементов, все элементы добавляются в массив result и выводятся.

**Задание 2:**

import java.util.Arrays;  
  
public class secondExercise {  
 public static void main(String[] args) {  
 Stack<Integer> stack = new Stack<>(5);  
  
 stack.push(1);  
 stack.push(2);  
 stack.push(3);  
 System.*out*.println(stack.view());  
  
 System.*out*.println("Удалён: " + stack.pop());  
 System.*out*.println(stack.view());  
  
 System.*out*.println("Верхний элемент: " + stack.peek());  
  
 stack.push(4);  
 System.*out*.println(stack.view());  
  
 System.*out*.println("Удалён: " + stack.pop());  
 System.*out*.println(stack.view());  
 }  
  
 public static class Stack<T> {  
 private T[] data;  
 private int index = 0;  
  
 public Stack(int capacity) {  
 data = (T[]) new Object[capacity];  
 }  
  
 public void push(T element) {  
 data[index] = element;  
 index++;  
 }  
  
 public T pop() {  
 index--;  
  
 T value = data[index];  
 data[index] = null;  
  
 return value;  
 }  
  
 public T peek() {  
 return data[index - 1];  
 }  
  
 public String view() {  
 return Arrays.*toString*(data);  
 }  
 }  
}

Создается класс secondExercise, содержащий метод main, который является точкой входа программы.

В методе main создается объект типа Stack<Integer> с именем stack. Он представляет стек, который будет хранить целые числа.

С помощью метода push() добавляются три элемента (1, 2, 3) в стек stack.

Выводится содержимое стека с помощью метода view(), который возвращает строковое представление массива элементов стека.

Используется метод pop(), чтобы удалить верхний элемент из стека и вывести его значение.

Выводится обновленное содержимое стека.

С помощью метода peek() получаем значение верхнего элемента стека, но не удаляем его.

Снова используется метод push(), чтобы добавить элемент 4 в стек.

Выводится обновленное содержимое стека.

Используется метод pop() для удаления верхнего элемента стека и вывод его значения.

Выводится финальное содержимое стека после удаления элемента.

Вложенный класс Stack<T> представляет реализацию стека. Он параметризирован типом T, так что может работать со стеком любого типа данных.

В конструкторе класса Stack создается массив data заданной ёмкости capacity для хранения элементов стека.

Метод push() добавляет элемент в стек путем помещения его в массив data на текущую позицию index и инкрементирует index.

Метод pop() удаляет и возвращает верхний элемент стека. Он декрементирует index, получает значение верхнего элемента из массива data, устанавливает соответствующий элемент в null и возвращает значение.

Метод peek() возвращает значение верхнего элемента стека без его удаления.

Метод view() преобразует массив data в строковое представление с помощью Arrays.toString() и возвращает получившуюся строку.

**Задание 3:**

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.LinkedList;  
  
public class thirdExercise {  
 public static void main(String[] args) {  
 Product product1 = new Product("Ice Cream", 89, 5);  
 Product product2 = new Product("Milk", 112, 2);  
 Product product3 = new Product("Cheese", 215, 1);  
 Product product4 = new Product("Popcorn", 2399, 4);  
 Product product5 = new Product("Energos", 10000, 1);  
  
 Stack stack = new Stack(Product.*countOfObject*);  
  
 stack.add(product2);  
 stack.add(product4);  
 stack.add(product4);  
 stack.add(product4);  
 stack.add(product4);  
 stack.add(product4);  
 stack.add(product5);  
 stack.add(product5);  
  
 System.*out*.println("Список проданных товаров: " + stack.getList());  
 System.*out*.println("Общая сумма продаж: " + stack.getTotalPrice());  
 System.*out*.println("Наиболее популярный товар: " + stack.getPopular());  
 }  
  
 public static class Product {  
 private String name;  
 private int cost;  
 private int quantity;  
 private int sold = 0;  
 private static int *countOfObject*;  
  
 public Product(String name, int cost, int quantity) {  
 this.name = name;  
 this.cost = cost;  
 this.quantity = quantity;  
  
 *countOfObject*++;  
 }  
  
 public int getPrice() {  
 return cost;  
 }  
  
 public void decreaseQuantity() {  
 quantity--;  
 sold++;  
 }  
 }  
  
 public static class Stack {  
 public LinkedList<Product>[] data;  
 private int index = 0;  
 private int cost = 0;  
 private int arrayFullness;  
  
 public Stack(int capacity) {  
 data = new LinkedList[capacity];  
 }  
  
 public void add(Product product) {  
 if (product.quantity == 0) {  
 return;  
 }  
  
 for (int i = 0; i < data.length; i++) {  
 if (data[i] != null && data[i].get(0) == product) {  
 product.decreaseQuantity();  
  
 cost += product.getPrice();  
 return;  
 }  
 }  
  
 data[index] = new LinkedList<>();  
 data[index].add(product);  
  
 product.decreaseQuantity();  
  
 cost += product.getPrice();  
  
 index++;  
 arrayFullness++;  
 }  
  
 public String getList() {  
 String[] list = new String[arrayFullness];  
  
 for (int i = 0; i < arrayFullness; i++) {  
 list[i] = data[i].get(0).name;  
 }  
  
 return Arrays.*toString*(list);  
 }  
  
 public int getTotalPrice() {  
 return cost;  
 }  
  
 public String getPopular() {  
 String name = null;  
 int sold = 0;  
  
 for (int i = 0; i < data.length; i++) {  
 if (data[i] == null) {  
 break;  
 }  
  
 Product product = data[i].get(0);  
 if (product.sold > sold) {  
 sold = product.sold;  
 name = product.name;  
 }  
 }  
  
 return name;  
 }  
  
 }  
}

Создается класс thirdExercise, содержащий метод main, который является точкой входа программы.

В методе main создаются объекты типа Product с разными параметрами. Каждый объект представляет продукт с названием (name), стоимостью (cost), и количеством (quantity), а также счетчиком проданных продуктов (sold).

Создается объект типа Stack с именем stack, который представляет стек для отслеживания проданных товаров.

С помощью метода add() добавляются объекты product2, product4 и product5 в стек stack. Метод add() также уменьшает количество доступных товаров (quantity) и увеличивает общую стоимость (cost) проданных товаров.

Выводится список проданных товаров с помощью метода getList(), возвращающего строку из названий проданных товаров.

Выводится общая стоимость продаж с помощью метода getTotalPrice(), возвращающего общую стоимость всех проданных товаров.

Выводится наиболее популярный товар с помощью метода getPopular(), который определяет продукт с наибольшим количеством продаж и возвращает его название.

Также, в коде присутствуют вложенные классы Product и Stack.

Класс Product представляет продукт. Он содержит поля для хранения названия, стоимости, количества в целом и количества проданных товаров. У класса есть конструктор, который инициализирует поля и увеличивает countOfObject - счетчик созданных объектов класса.

У класса Product есть метод getPrice(), который возвращает стоимость продукта, и метод decreaseQuantity(), который уменьшает количество доступных товаров и увеличивает счетчик проданных товаров.

Класс Stack представляет реализацию стека. Он содержит поле data - массив связных списков, представляющих элементы стека. Также есть поля index - текущий индекс элемента, cost - общая стоимость продаж, и arrayFullness - количество элементов в массиве.

У класса Stack есть метод add(), который добавляет продукт в стек. При добавлении, метод проверяет, если количество товара равно 0, то он не добавляется. Затем проверяется каждый элемент в массиве data, чтобы найти уже добавленный продукт. Если продукт найден, уменьшается количество доступных товаров и обновляется общая стоимость продаж. Если продукт не найден, создается новый связный список в массиве data, и продукт добавляется в него.

Метод getList() возвращает строку, содержащую названия проданных товаров, полученные из первого элемента каждого связного списка в массиве data.

Метод getTotalPrice() возвращает общую стоимость продаж, сохраненную в поле cost.

Метод getPopular() находит наиболее популярный товар (тот, у которого было наибольшее количество продаж) и возвращает его название.

**Вывод**

Мы смогли разобраться в том, как устроены стеки, при этом сумели создать программу, ответственную за подсчёт проданных товаров и учёт в принципе.