

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

**Ордена трудового Красного Знамени федеральное государственное  
бюджетное**

**образовательное учреждение высшего образования  
«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 6

Выполнил: студент группы БВТ2402

Косякова Олеся Дмитриевна

Москва, 2025

Цель работы: Освоение практического применения основных структур данных и коллекций Java для решения типовых задач обработки информации.

Задание:

- 1) Написать программу, которая считывает текстовый файл и выводит на экран топ-10 самых часто встречающихся слов в этом файле.
- 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.
- 3) 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.

Ход работы:

Задание 1.

Создаем объект File для указанного пути, далее объявляем Scanner для чтения файла и обрабатываем исключение.

```
public class TopWords {  
    Run | Debug  
    public static void main(String[] args) {  
        File file = new File(pathname: "lab6/exm.txt");  
        Scanner scanner = null;  
        try {  
            scanner = new Scanner(file);  
        } catch (FileNotFoundException e) {  
            System.err.println("Файл не найден: " + e.getMessage());  
        }  
    }  
}
```

Создаем коллекцию, в которой будут храниться пары слово и его количество. Далее с помощью цикла читаем каждое следующее слово и обрабатываем его: приводим к нижнему регистру, убираем лишние пробелы, и заменяем все символы не принадлежащие кириллице и латинице на “”. Далее если это слово уже есть в нашей коллекции мы прибавляем единицу к его значению, если нет, то добавляем слово в коллекцию.

```

Map<String, Integer> numOfWords = new TreeMap<>();
while (scanner.hasNext()) {
    String word = scanner.next().toLowerCase().trim().replaceAll(regex: "[^a-zA-Zа-яА-ЯёЁ]", replacement: "");
    if (!word.isEmpty() && word.length() > 1) {
        if (numOfWords.containsKey(word)) {
            int currentCount = numOfWords.get(word);
            numOfWords.put(word, currentCount + 1);
        } else {
            numOfWords.put(word, value: 1);
        }
    }
}
}

```

Создаем новую отсортированную коллекцию. `entrySet().stream()` – преобразует Map в поток ключ-значение, `sorted(Map.Entry.comparingByValue().reversed())` - сортирует по убыванию количества (`Map.Entry.comparingByValue()` – создает компаратор для сравнения значений), `limit(10)` - берет только первые 10 элементов, `collect(...)` - собирает обратно в Map, сохраняя порядок через `LinkedHashMap` (внутри прописываем извлечение ключа, значения, функцию разрешения конфликтов при коллизии и создание конкретной реализации Map для сохранения порядка вставки)

```

scanner.close();
Map<String, Integer> sorted = numOfWords.entrySet().stream()
    .sorted(Map.Entry.<String, Integer>comparingByValue().reversed())
    .limit(maxSize: 10).collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::new));
System.out.println(sorted);

```

## Задание 2.

Создаем класс, который принимает любой тип данных, создаем массив для хранения элементов и переменную, хранящую значение верхнего элемента. Далее в конструкторе создаем массив `Object` и приводим к `T[]`, задаем полю `top` значение `-1`, что обозначает что стек пуст.

```

class Stack<T> {
    private T[] data;
    private int top;

    public Stack(int capacity) {
        data = (T[]) new Object[capacity];
        top = -1;
    }
}

```

Метод `push` реализует добавление элемента в стек, проверяем стек на переполнение и далее увеличиваем значение `top` и далее на это место добавляется элемент. Метод `pop` реализует удаление элемента, для этого уменьшаем индекс массива и таким образом вершиной стека станет предпоследний элемент, а последний будет удален с помощью сборщика мусора.

```

public void push(T element) {
    if (top == data.length - 1) {
        throw new StackOverflowError(s: "Stack переполнен");
    }
    data[++top] = element;
}

public T pop() {
    if (isEmpty()) {
        throw new EmptyStackException();
    }
    return data[top--];
}

public T peek() {
    if (isEmpty()) {
        throw new EmptyStackException();
    }
    return data[top];
}

```

### Задание 3.

Создаем класс Sale, который будет хранить информацию об одной продаже. Объявляем поля название, цена и количество. Далее переопределяем метод toString(), для удобства вывода информации в консоль и прописываем метод, который будет считать сумму продажи товара.

```

public class Sale {
    String productName;
    double price;
    int quantity;

    public Sale(String productName, double price, int quantity) {
        this.productName = productName;
        this.price = price;
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Товар: " + productName +
            ", цена: " + price + " руб." +
            ", количество: " + quantity + " шт." +
            ", сумма: " + (price * quantity) + " руб.";
    }

    public double getTotal() {
        return price * quantity;
    }
}

```

Создаем метод, добавляющий продукты в нашу коллекцию, если пользователь нажмет enter цикл остановится.

```

private static void addSales(Scanner sc, ArrayList<Sale> sales) {
    while (true) {
        System.out.print(s: "Товар: ");
        String name = sc.nextLine().trim();
        if (name.isEmpty())
            break;

        System.out.print(s: "Цена: ");
        double price = sc.nextDouble();

        System.out.print(s: "Кол-во: ");
        int qty = sc.nextInt();
        sc.nextLine();

        sales.add(new Sale(name, price, qty));
    }
}

```

Метод вывода всех продаж реализуем с помощью цикла, который проходится по всему массиву. Метод вывода конечной суммы так же реализуем с помощью цикла и добавляем в переменную значение метода getTotal() каждого товара. Для определения самого популярного товара мы записываем в новую переменную первый объект и далее сравниваем его количество продаж и следующего и записываем в переменную best.

```

private static void printAllSales(ArrayList<Sale> sales) {
    System.out.println(x: "\nВсе продажи:");
    for (Sale s : sales) {
        System.out.println(s);
    }
}

private static void printTotalSum(ArrayList<Sale> sales) {
    double total = 0;
    for (Sale s : sales)
        total += s.getTotal();
    System.out.printf(format: "Итого: ", total);
}

private static void printMostPopular(ArrayList<Sale> sales) {
    Sale best = sales.get(index: 0);
    for (Sale s : sales) {
        if (s.quantity > best.quantity ||
            (s.quantity == best.quantity && s.getTotal() > best.getTotal())) {
            best = s;
        }
    }
    System.out.println(x: "Лидер по количеству:");
    System.out.println(best);
}

```

Вывод: В ходе лабораторной работы мы освоили основные структуры данных и коллекций Java для решения типовых задач обработки информации.

Ссылка на репозиторий: [https://github.com/swydiz/information\\_technology](https://github.com/swydiz/information_technology)