

**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

**Лабораторная работа №6**  
**Работа с коллекциями**

Выполнил: студент группы БВТ2402

Голиков Михаил Вячеславович

Руководитель: Мосева Марина Сергеевна

Москва, 2077

## Цель работы

Целью лабораторной работы является изучение основных структур данных и коллекций в языке программирования Java, освоение принципов работы с обобщёнными типами (generics), а также приобретение навыков использования интерфейсов Map, List и TreeMap для решения практических задач. В ходе работы предполагается реализовать программы для анализа текстовых данных, создания универсального стека и учёта продаж в магазине.

## Индивидуальное задание

В рамках лабораторной работы необходимо реализовать следующие задачи:

### 1. Задача 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.

### 2. Задача 2

Написать обобщённый 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,
- peek для получения верхнего элемента стека без его удаления.

### 3. Задача 3

Разработать программу для учёта продаж в магазине. Программа должна позволять:

- добавлять проданные товары в коллекцию,
- выводить список проданных товаров,
- считать общую сумму продаж,

- определять наиболее популярный товар. Использовать TreeMap для хранения пар «товар – количество продаж».

(Вариант 6).

## Основная часть

В процессе выполнения лабораторной работы были реализованы три отдельные программы, каждая из которых демонстрирует применение различных коллекций и обобщённых типов в Java:

1. **TopWords** — программа для анализа текстового файла и вывода наиболее часто встречающихся слов. Используется HashMap для подсчёта повторений и сортировка списка записей для определения топа слов.
2. **CustomStack** — реализация обобщённого стека на основе массива. Класс Stack<T> поддерживает основные операции: push, pop и peek. Продемонстрирована работа с generics и управление внутренним массивом данных.
3. **Var6** — программа для учёта продаж в магазине. Используется TreeMap для хранения информации о проданных товарах, а также методы для подсчёта общей выручки и определения самого популярного товара.

```
package Lab_6;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class TopWords {
    public static void main(String[] args) {
        String filePath = "FileWithNames.txt";
        File file = new File(filePath);

        Scanner scanner = null;
        try {
```

```

        scanner = new Scanner(file);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
        return;
    }
    // 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 name = scanner.nextLine();
        if (map.containsKey(name)) {
            map.put(name, map.get(name) + 1);
        } else {
            map.put(name, 1);
        }
    }

    try {
        scanner.close();
    } catch (Exception e) {
        //throw new RuntimeException(e);
        e.printStackTrace();
    }
    // создаем список из элементов Map
    List<Map.Entry<String, Integer>> list = new ArrayList<>(map.size());
    for (Map.Entry<String, Integer> entry : map.entrySet()){
        list.add(entry);
    }
    // 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) {
            return (int) o2.getValue() - o1.getValue();
        }
    });
    // выводим топ-10 слов
    int i = 1;
    for (Map.Entry<String, Integer> entry : list){
        System.out.println(entry.getKey() + " " + entry.getValue());
        if (i >= 10) break;
        i++;
    }
}
}

```

## Элемент 1 — Код программы 1

```
C:\Users\proto\...
the 34
you 33
in 33
it 33
as 33
a 33
to 33
that 33
for 33
and 33
```

Элемент 2 — Результат программы 1

```
package Lab_6;

public class CustomStack{
    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());
        System.out.println(stack.peek());
        stack.push(4);
        System.out.println(stack.pop());
    }
}

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

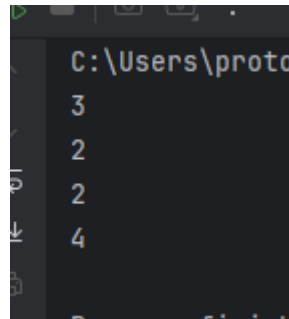
    public Stack(int capacity) {
        data = (T[]) new Object[capacity];
        size = 0;
    }
    public void push (T element){
        data[size] = element;
        size++;
    }
    public T pop() {
        if (size <= 0) return null;
        T element = (T) data[size-1];
        size--;
        return element;
    }
    public T peek() {
```

```

        if (size <= 0) return null;
        return (T) data[size-1];
    }
}

```

### Элемент 3 — Код программы 2



### Элемент 4 — Результат программы 2

```

package Lab_6;

import java.util.*;

public class Var6 {
    public static void main(String[] args){
        List<Map.Entry<String, Integer>> products = new ArrayList<>(5);
        products.add(new AbstractMap.SimpleEntry<>("Banana", 200));
        products.add(new AbstractMap.SimpleEntry<>("Apple", 40));
        products.add(new AbstractMap.SimpleEntry<>("XLEB", 30));
        products.add(new AbstractMap.SimpleEntry<>("Ananas", 2000));
        products.add(new AbstractMap.SimpleEntry<>("Desert Eagle 50.cal",
10000));

        TradingTable myTable = new TradingTable(products);

        myTable.addProd("Banana");

        System.out.println(myTable.getSum());

        myTable.addProd("Banana");
        myTable.addProd("Banana");
        myTable.addProd("Banana");
        myTable.addProd("Desert Eagle 50.cal");
        myTable.addProd("XLEB");
        myTable.addProd("XLEB");

        System.out.println(myTable.getSum());
        System.out.println(myTable.getList().toString());
        System.out.println(myTable.getBestSeller());
    }
}

class TradingTable{

    private TreeMap<String, Integer> deals;
    private List<Map.Entry<String, Integer>> products;

```

```

public TradingTable(List<Map.Entry<String, Integer>> products){
    deals = new TreeMap<String, Integer>();
    this.products = products;
}

public void addProd(String name){
    if (deals.containsKey(name)) deals.put(name, deals.get(name) + 1);
    else deals.put(name, 1);
}

public List<Map.Entry<String, Integer>> getList(){
    List<Map.Entry<String, Integer>> listOfBoughtProds = new
ArrayList<>(deals.size());
    listOfBoughtProds.addAll(deals.entrySet());
    return listOfBoughtProds;
}

public int getSum(){
    int sum = 0;
    List<Map.Entry<String, Integer>> list = getList();
    for (Map.Entry<String, Integer> product : list){
        for (Map.Entry<String, Integer> prodEntry : products) {
            if (prodEntry.getKey().equals(product.getKey())) {
                sum = sum + prodEntry.getValue() * product.getValue();
                break;
            }
        }
    }
    return sum;
}

public Map.Entry<String, Integer> getBestSeller(){
    List<Map.Entry<String, Integer>> list = getList();
    Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {
        @Override
        public int compare(Map.Entry<String, Integer> o1,
Map.Entry<String, Integer> o2) {
            return (int) o2.getValue() - o1.getValue();
        }
    });
    return list.get(0);
}
}

```

### Элемент 5 — Код программы 3

```

C:\Users\proto\.jdk\openjdk-25\bin\java.exe
200
10860
[Banana=4, Desert Eagle 50.cal=1, XLEB=2]
Banana=4
Process finished with exit code 0

```

### Элемент 6 — Результат программы 4

## Заключение

В ходе выполнения лабораторной работы были успешно изучены и применены на практике основные структуры данных и коллекции в Java. Были реализованы программы, демонстрирующие:

- использование Map и List для анализа текстовых данных;
- создание обобщённого класса стека с применением generics;
- организацию учёта данных с помощью TreeMap;
- сортировку и обработку записей для получения статистики.

Работа позволила закрепить навыки работы с коллекциями, обобщёнными типами и алгоритмами обработки данных, что важно для разработки эффективных и структурированных приложений на Java.

**Github:** <https://github.com/Prototype721/Java>