

**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:

БПИ2401

Мещеряков Кирилл Владимирович

Руководитель:

Харрасов Камиль Раисович

Москва, 2025

Цель работы:

Изучение и практическое освоение основных коллекций Java Collections Framework, включая интерфейсы List, Set и Map, а также получение навыков работы с обобщенными типами (generics).

Индивидуальное задание:

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 данного функционала представлен на листинге 6.1.

Листинг 6.1. Пример реализации класса TopWord

1. 2. 3. 4. 5. 6. 7. 8. 9.

10. 11.

```
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
```

12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32.
33. 34. 35. 36. 37. 38. 39. 40. 41. 42.

```
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 обобщенного класса `Stack< T>` представлен на листинге 6.2.

Листинг 6.2. Пример реализации класса `Stack< T>`

```
1. public class Stack< T>
```

```
2. {
```

```
3.     private T[] data;
```

```
4.     private int size;
```

```
5. }
```

```
6.     public Stack(int capacity) {
```

```
7.         data = (T[]) new Object[capacity];
```

```
8.         size = 0; }
```

```
9.     public void push(T element) {
```

```
10.         if (size == data.length) {
```

```
11.             return;
```

```
12.         data[size] = element;
```

```
13.         size++;
```

```
14.     }

15.     public T pop() {
```

```
16.         if (size == 0) {
```

```
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.

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.

10. Использовать CopyOnWriteArrayList для хранения списка проданных товаров с возможностью одновременного чтения и записи.

Выполнение:

```
package lab6;  
  
import java.io.File;  
  
import java.io.FileNotFoundException;  
  
import java.util.*;
```

```
public class TopWords {

    public static void main(String[] args) {

        String filePath = "src/lab6/text.txt";

        File file = new File(filePath);

        Scanner scanner = null;

        try {

            scanner = new Scanner(file, "UTF-8");

        } catch (FileNotFoundException e) {

            e.printStackTrace();

            return;

        }

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

        while (scanner.hasNext()) {

            String word = scanner.next().toLowerCase().replaceAll("[^a-яa-z]", "");

            if (!word.isEmpty()) {

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

            }

        }

    }

}
```

```
    }  
}
```

```
scanner.close();
```

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

```
// Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {  
  
//    @Override  
  
//    public int compare(Map.Entry<String, Integer> o1, Map.Entry<String,  
Integer> o2) {  
  
//        return o2.getValue().compareTo(o1.getValue());  
  
//    }  
  
// });
```

```
list.sort(Map.Entry.<String, Integer>comparingByValue().reversed());
```

```
System.out.println("Топ-10 самых частых слов:");
```

```
int count = 0;
```

```
for (Map.Entry<String, Integer> entry : list) {
```

```
    if (count >= 10) break;
```

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

        count++;

    }

}

}
```

```
package lab6;
```

```
public 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 RuntimeException("!B5: ?5@5?>:=5=");  
    }  
  
    data[size] = element;  
  
    size++;  
}
```

```
public T pop() {  
    if (size == 0) {  
        throw new RuntimeException("!B5: ?CAB");  
    }  
  
    size--;  
  
    T element = data[size];  
  
    data[size] = null;  
  
    return element;  
}
```

```
public T peek() {  
    if (size == 0) {  
        throw new RuntimeException("!B5: ?CAB");  
    }  
}
```

```
    return data[size - 1];  
}
```

```
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());  
  
}  
}
```

```
package lab6;
```

```
import java.util.HashMap;
```

```
import java.util.Map;
```

```
import java.util.concurrent.CopyOnWriteArrayList;
```

```
class Product {  
  
    String name;  
  
    double price;  
  
    Product(String name, double price) {  
  
        this.name = name;  
  
        this.price = price;  
  
    }  
}  
  
public class SalesTracker {  
  
    private CopyOnWriteArrayList<Product> sales;  
  
    public SalesTracker() {  
  
        sales = new CopyOnWriteArrayList<>();  
  
    }  
  
    public void addSale(String name, double price) {  
  
        sales.add(new Product(name, price));  
  
    }  
}
```

```
}
```

```
public void showSales() {
```

```
    System.out.println("Проданные товары:");
```

```
    for (Product p : sales) {
```

```
        System.out.println(p.name + " - " + p.price + " руб.");
```

```
    }
```

```
}
```

```
public double getTotalSum() {
```

```
    double total = 0;
```

```
    for (Product p : sales) {
```

```
        total += p.price;
```

```
    }
```

```
    return total;
```

```
}
```

```
public String getMostPopular() {
```

```
    if (sales.isEmpty()) {
```

```
        return "Нет проданных товаров";
```

```
}
```

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

```
for (Product p : sales) {
```

```
    productCount.put(p.name, productCount.getOrDefault(p.name, 0) + 1);
```

```
}
```

```
String popular = "";
```

```
int maxCount = 0;
```

```
for (Map.Entry<String, Integer> entry : productCount.entrySet()) {
```

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

```
        maxCount = entry.getValue();
```

```
        popular = entry.getKey();
```

```
    }
```

```
}
```

```
return popular;
```

```
}
```

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

```
    SalesTracker tracker = new SalesTracker();
```

```
tracker.addSale("Hleb", 50.0);

tracker.addSale("Moloko", 80.0);

tracker.addSale("Hleb", 50.0);

tracker.addSale("Maslo", 150.0);

tracker.addSale("Moloko", 80.0);

tracker.addSale("Hleb", 50.0);


tracker.showSales();

System.out.println("\nObshaya summa prodazh: " + tracker.getTotalSum()
+ " rub.");

System.out.println("Samyj populyarnyj tovar: " +
tracker.getMostPopular());

}

}
```

```
> java -cp bin lab6.TopWords
Топ-10 самых частых слов:
java: 7
и: 3
на: 2
для: 2
программирование: 2
программирования: 2
разработки: 1
разработчики: 1
позволяет: 1
между: 1
```

```
Между: 1  
> java -cp bin lab6.Stack  
3  
2  
4
```

```
> java -cp bin lab6.SalesTracker  
Проданные товары:  
Хлеб – 50.0 руб.  
Молоко – 80.0 руб.  
Хлеб – 50.0 руб.  
Масло – 150.0 руб.  
Молоко – 80.0 руб.  
Хлеб – 50.0 руб.  
  
Общая сумма продаж: 460.0 руб.  
Самый популярный товар: Хлеб
```

Контрольные вопросы:

- ☐ Какие интерфейсы коллекций есть в Java?

Основные интерфейсы: Collection (базовый), List (упорядоченные коллекции с индексами), Set (множества без дубликатов), Queue (очереди), Deque (двусторонние очереди), Map (пары ключ-значение).

- ☐ Какие классы коллекций есть в Java?

ArrayList, LinkedList, HashSet, TreeSet, LinkedHashSet, HashMap, TreeMap, LinkedHashMap, PriorityQueue, ArrayDeque, ConcurrentHashMap, CopyOnWriteArrayList и другие.

- ☐ Что такое итератор?

Итератор — объект, предоставляющий последовательный доступ к элементам коллекции через методы `hasNext()`, `next()` и `remove()`, позволяющий обходить коллекцию без знания её внутренней структуры.

- ☐ Как работают коллекции на основе интерфейса Map?

Map хранит пары ключ-значение, где каждый ключ уникален. Доступ к значениям осуществляется по ключу за $O(1)$ в HashMap или $O(\log n)$ в TreeMap. Не являются наследниками Collection.

☐ Как работают коллекции на основе интерфейса List?

List поддерживает упорядоченную последовательность элементов с возможностью доступа по индексу, допускает дубликаты. ArrayList использует динамический массив, LinkedList — двусвязный список.

☐ Как работают коллекции на основе интерфейса Set?

Set хранит уникальные элементы без дубликатов. HashSet использует хеш-таблицу, TreeSet — красно-черное дерево с сортировкой, LinkedHashSet сохраняет порядок добавления.

☐ Как можно синхронизировать коллекции в Java?

Использовать Collections.synchronizedList/Set/Map() для оборачивания коллекций, применять потокобезопасные классы (ConcurrentHashMap, CopyOnWriteArrayList) или внешнюю синхронизацию через synchronized блоки.

☐ Какие методы предоставляет интерфейс Collection?

add(), remove(), contains(), size(), isEmpty(), clear(), iterator(), addAll(), removeAll(), retainAll(), toArray() и другие базовые операции.

☐ Какие реализации интерфейса List вы знаете?

ArrayList (на основе массива), LinkedList (двусвязный список), Vector (устаревший синхронизированный), Stack (устаревший стек), CopyOnWriteArrayList (потокобезопасный).

☐ Какие реализации интерфейса Set вы знаете?

HashSet, LinkedHashSet, TreeSet, EnumSet, ConcurrentSkipListSet, CopyOnWriteArraySet.

☐ Что такое Comparable и Comparator?

Comparable — интерфейс с методом compareTo() для естественного порядка сортировки объекта. Comparator — отдельный класс с методом compare() для определения альтернативного порядка сортировки.

☐ Что такое параметр типа?

Параметр типа (type parameter) — переменная типа, обозначаемая заглавной буквой (T, E, K, V), которая используется при объявлении обобщенных классов, интерфейсов или методов.

☐ Как параметризовать метод, класс?

Класс: `class Box<T> { }`. Метод: `public <T> void method(T param) { }`.

Параметры типа указываются в угловых скобках перед именем класса или перед возвращаемым типом метода.

☐ Что такое стирание типов?

Type erasure — механизм, при котором компилятор удаляет информацию о параметрах типов после проверки типов, заменяя их на `Object` или границу типа для обратной совместимости с кодом до Java 5.

☐ Как можно обойти ограничения стирания типов?

Использовать `Class<T>` токены, передавать объект `Class` в конструктор, применять рефлексию, использовать супертипы или библиотеки вроде `TypeToken` из `Guava`.

☐ Как работают дженерики с массивами?

Из-за стирания типов и ковариантности массивов возникают проблемы безопасности типов. Массивы хранят информацию о типе элементов в `runtime`, дженерики — нет.

☐ Можно ли создать массив дженериков?

Нельзя напрямую создать массив параметризованного типа (`new T[10]` запрещено). Обходное решение — создание массива `Object` с приведением: `(T[]) new Object[10]`.

☐ Что такое wildcard тип?

Wildcard (?) — неизвестный тип в дженериках. Используется в трёх формах: неограниченный (?), с верхней границей (? extends Type), с нижней границей (? super Type).

☐ В чем разница между `<? extends T>` и `<? super T>`?

<? extends T> (upper bound) — тип или его наследники, используется для чтения. <? super T> (lower bound) — тип или его предки, используется для записи. Принцип PECS: Producer-Extends, Consumer-Super.

Заключение:

В ходе выполнения лабораторной работы были получены практические навыки работы с Java Collections Framework. Реализация задачи анализа частотности слов в текстовом файле продемонстрировала эффективность использования Map для подсчета повторяющихся элементов и применение Comparator для сортировки данных. Создание обобщенного класса Stack<T> позволило углубить понимание механизма generics и их роли в обеспечении типобезопасности при работе с коллекциями различных типов данных.

Выполнение третьего задания по разработке системы учета продаж показало важность правильного выбора структуры данных в зависимости от требований задачи: производительности операций добавления и поиска, необходимости сортировки, потокобезопасности. Сравнение различных реализаций коллекций (ArrayList vs LinkedList, HashMap vs TreeMap, стандартные vs concurrent коллекции) помогло осознать компромиссы между скоростью работы, потреблением памяти и функциональными возможностями.

Освоенные навыки работы с коллекциями и обобщенными типами являются фундаментальными для разработки на Java и будут применяться во всех последующих проектах, требующих эффективной обработки и хранения данных.