**Коллекции в Java: структура данных и их применение**

**Введение в коллекции**

Коллекции в Java предоставляют мощный и гибкий способ работы с группами объектов. Они позволяют хранить, обрабатывать и манипулировать данными с минимальными усилиями. Все коллекции находятся в пакете java.util.

Коллекции включают интерфейсы и классы, которые упрощают работу с такими структурами данных, как списки, множества, карты и очереди.

**Преимущества коллекций:**

* Динамическое управление размерами (в отличие от массивов).
* Простые методы добавления, удаления, поиска и итерации элементов.
* Широкий выбор структур данных для различных задач.

**Иерархия коллекций**

Иерархия начинается с интерфейса **Collection**, который наследуют основные интерфейсы:

1. **List** — упорядоченная коллекция с доступом к элементам по индексу.
2. **Set** — коллекция без дубликатов.
3. **Queue** — коллекция для работы с очередями.

Дополнительно существует интерфейс **Map**, который используется для хранения пар "ключ-значение".

**Основные интерфейсы и их реализации**

**List**  
Списки хранят элементы в определенном порядке и допускают дублирование.

* + ArrayList — динамический массив, эффективный для операций чтения.
  + LinkedList — двусвязный список, эффективный для вставки и удаления элементов.

1. import java.util.ArrayList;
2. public class Main {
3. public static void main(String[] args) {
4. ArrayList<String> list = new ArrayList<>();
5. list.add("Apple");
6. list.add("Banana");
7. list.add("Apple");
8. System.out.println(list); // [Apple, Banana, Apple]
9. }
10. }

**Set**  
Множества хранят только уникальные элементы.

* HashSet — неупорядоченное множество, быстрое для операций добавления и проверки.
* TreeSet — упорядоченное множество на основе бинарного дерева.

import java.util.HashSet;

public class Main {

    public static void main(String[] args) {

        HashSet<Integer> set = new HashSet<>();

        set.add(1);

        set.add(2);

        set.add(2); // Дубликат будет проигнорирован

        System.out.println(set); // [1, 2]

    }

}

**Map**  
Карты позволяют хранить данные в виде пар "ключ-значение".

* HashMap — быстрая реализация без порядка.
* TreeMap — упорядоченная карта.
* import java.util.HashMap;
* public class Main {
* public static void main(String[] args) {
* HashMap<String, Integer> map = new HashMap<>();
* map.put("Alice", 25);
* map.put("Bob", 30);
* System.out.println(map); // {Alice=25, Bob=30}
* }
* }

**Методы коллекций**

Коллекции предоставляют удобные методы для работы с данными:

* add(element) — добавляет элемент.
* remove(element) — удаляет элемент.
* contains(element) — проверяет наличие элемента.
* size() — возвращает количество элементов.

**Применение коллекций**

Коллекции широко используются для обработки данных в реальных приложениях:

* **Хранение и управление списком пользователей.**
* **Фильтрация уникальных элементов из больших наборов данных.**
* **Упрощение работы с очередями запросов или задач.**
* **Создание структуры для хранения и быстрого поиска данных по ключу.**