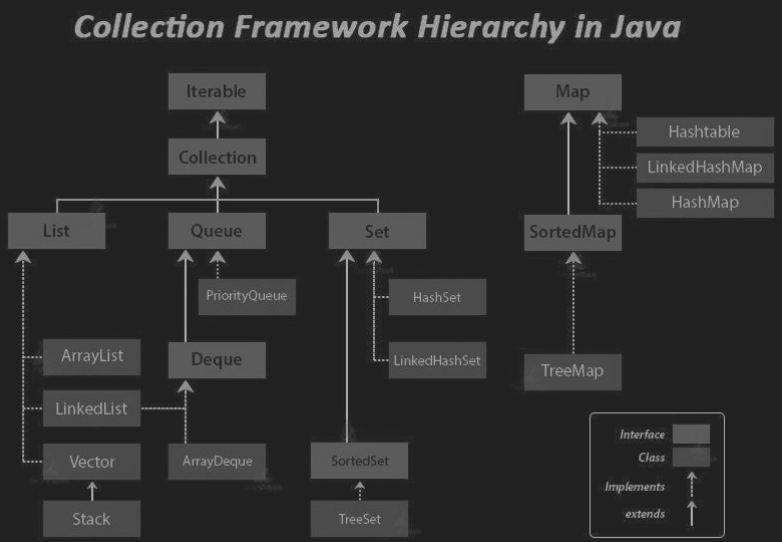
Java Collection II

Java Collection Framework



Java Collection Framework предоставляет архитектуру для хранения и манипулирования группой объектов.

Он включает в себя различные интерфейсы, реализации и алгоритмы для работы с коллекциями.

Еще одна статья покороче: ссылка

Comparator и Comparable

```java
package l35.slides.ex0;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class Employee implements Comparable<Employee> {
    String name;
    double salary;

    Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    @Override
    public int compareTo(Employee other) {
        return Double.compare(this.salary, other.salary);
    }

    @Override
    public String toString() {
        return name + ": " + salary;
    }
}

class EmployeeNameComparator implements Comparator<Employee> {
    @Override
    public int compare(Employee e1, Employee e2) {
        return e1.name.compareTo(e2.name);
    }
}
```

Comparable и Comparator — два интерфейса, используемых для сравнения объектов в Java.

Comparable: Этот интерфейс определяет метод compareTo, который используется для сравнения объекта с другим объектом того же типа. Классы, реализующие Comparable, могут быть автоматически сравнены и упорядочены (например, в TreeSet или TreeMap).

Comparator и Comparable

```
package l35.slides.ex0;

import java.util.ArrayList;
import java.util.List;


public class Main {
   public static void main(String[] args) {
       List<Employee> employees = new ArrayList<>();
       employees.add(new Employee("John", 50000.00));
       employees.add(new Employee("Alice", 60000.00));
       employees.add(new Employee("Bob", 40000.00));

       // Сортировка по зарплате
       Collections.sort(employees);
       System.out.println("Отсортировано по зарплате: " + employees);

       // Сортировка по имени
       Collections.sort(
           employees, new EmployeeNameComparator()
       );
       System.out.println("Отсортировано по имени: " + employees);

       // Сортировка по имени
       Collections.sort(
            employees,
            (e1, e2) -> e1.name.compareTo(e2.name)
       );
       System.out.println("Отсортировано по имени: " + employees);

       // Сортировка по имени
       employees.sort((e1, e2) -> e1.name.compareTo(e2.name));
       System.out.println("Отсортировано по имени: " + employees);
   }
}
```

Comparator: Этот интерфейс определяет метод compare, который используется для сравнения двух объектов.

Comparator удобен, когда вам нужно сравнить объекты в способ, отличный от их естественного порядка сравнения, или когда объекты не реализуют Comparable.

Map и HashMap

```java
package l35.slides.ex2;

import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<String, Integer> map =  new HashMap<>();
        map.put("Apple", 100); // Добавление пары ключ-значение
        map.put("Banana", 200);

        // Получение значения по ключу
        // Выводит "Цена Apple: 100"
        System.out.println("Цена Apple: " + map.get("Apple"));

        // Итерация по ключам карты
        for (String key : map.keySet()) {
            System.out.println(key + " стоит " + map.get(key));
        }
    }
}
```

Map — это объект, который хранит пары ключ-значение. Каждый ключ уникален, и на каждый ключ приходится ровно одно значение.

HashMap — это реализация Map, использующая хеш-таблицу.

HashMap: Эффективно для поиска, вставки и удаления элементов. Не гарантирует порядка элементов.

Контрольная точка

- Понятно ли?

Если все ясно, ставим плюсы, иначе - задаем вопросы.

Домашнее задание

Создайте значит класс Bobr и определите у бобра поля name, age и relativesCount.

Реализуйте интерфейс Comparable<Bobr> и переопределите метод compareTo чтобы сравнивались поля age

Создайте класс NameComparator и RelativesCountComparator (реализуйте интерфейс java.util.Comporator)

Поместите бобров в список (выберите сами какой список, он должен реализовывать интерфейс List)

Отсортируйте список используя Collections.sort

Отсортируйте список используя Collections.sort и созданые компораторы

Отсортируйте список используя Collections.sort используя лямбда выражения вместо компараторов

после всей этой сортировки сформируйте Map (например Hashmap) с данными вида имя бобра (ключ) и количество родственников (значение) и выведите этот map

The end