Cập nhật tháng 8 năm 2024

[Bài đọc] Sắp xếp danh sách với Comparable và Comparator

Khi làm việc với các tập hợp (collections) trong Java, việc sắp xếp các phần tử thường là một yêu cầu quan trọng. Java cung cấp hai giao diện chính để hỗ trợ sắp xếp: **Comparable** và **Comparator**. Dưới đây là chi tiết về cách sử dụng và sự khác nhau giữa chúng.

1. **Interface Comparable**
   - Định nghĩa
     - Giao diện Comparable được sử dụng để định nghĩa thứ tự tự nhiên (natural ordering) của các đối tượng trong lớp
   - Phương thức
     - compareTo(T o): So sánh đối tượng hiện tại với đối tượng được truyền vào
       - Trả về **số âm** nếu đối tượng hiện tại nhỏ hơn đối tượng được so sánh
       - Trả về **số 0** nếu hai đối tượng bằng nhau
       - Trả về **số dương** nếu đối tượng hiện tại lớn hơn đối tượng được so sánh
   - Ví dụ

```java
class Student implements Comparable<Student>  6 usages  new *
{
    private int id;  5 usages
    private String name;  2 usages

    public Student(int id, String name)  3 usages  new *
    {
        this.id = id;
        this.name = name;
    }

    public int getId()  no usages  new *
    {
        return id;
    }

    @Override  new *
    public int compareTo(Student other)
    {
        return Integer.compare(this.id, other.id);
    }

    @Override  new *
    public String toString()
    {
        return "Student{id=" + id + ", name='" + name + "'}";
    }
}
```

```java
public static void main(String[] args)  new *
{
    List<Student> students = new ArrayList<>();
    students.add(new Student( id: 3,  name: "Alice"));
    students.add(new Student( id: 1,  name: "Bob"));
    students.add(new Student( id: 2,  name: "Charlie"));

    Collections.sort(students); // Sử dụng Comparable
    System.out.println(students);
}
```

2. **Interface Comparator**

- Định nghĩa
    - Comparator được sử dụng để định nghĩa thứ tự tùy chỉnh (custom ordering) giữa các đối tượng mà không cần sửa đổi lớp của đối tượng
- Phương thức
    - **int compare(T o1, T o2)**: So sánh hai đối tượng
        - Trả về **số âm** nếu o1 nhỏ hơn o2
        - Trả về **số 0** nếu hai đối tượng bằng nhau
        - Trả về **số dương** nếu o1 lớn hơn o2
- Ví dụ

```java
class Student  11 usages  new *
{
    private int id;  4 usages
    private String name;  4 usages

    public Student(int id, String name)  6 usages  new *
    {
        this.id = id;
        this.name = name;
    }

    public int getId()  no usages  new *
    {
        return id;
    }

    public void setId(int id)  no usages  new *
    {
        this.id = id;
    }

    public String getName()  new *
    {
        return name;
    }

    public void setName(String name)  no usages  new *
    {
        this.name = name;
    }

    @Override  new *
    public String toString()
    {
        return "Student{id=" + id + ", name='" + name + "'}";
    }
}
```

```java
class NameComparator implements Comparator<Student>  1 usage  new *
{
    @Override  new *
    public int compare(Student s1, Student s2) {
        return s1.getName().compareTo(s2.getName());
    }
}
```

```java
public static void main(String[] args)  new *
{
    List<Student> students = new ArrayList<>();
    students.add(new Student( id: 3,  name: "Alice"));
    students.add(new Student( id: 1,  name: "Bob"));
    students.add(new Student( id: 2,  name: "Charlie"));

    Collections.sort(students, new NameComparator()); // Sử dụng Comparator
    System.out.println(students);
}
```

3. So sanh Comparable và Comparator

Tiêu chí	Comparable	Comparator

Khai báo	Cần sửa đổi lớp của đối tượng (implements Comparable)	Không cần sửa đổi lớp đối tượng
Số thứ tự	Chỉ có **1** cách sắp xếp (thứ tự tự nhiên)	Hỗ trợ **nhiều** cách sắp xếp (tùy chỉnh)
Phương thức	compareTo(T o)	compare(T o1, T o2)
Ứng dụng	Khi muốn sắp xếp mặc định (ví dụ: sắp xếp theo id)	Khi cần sắp xếp theo nhiều tiêu chí khác nhau
Ví dụ tiêu biểu	Sắp xếp String, Integer, Date	Sắp xếp theo tên, điểm số, hoặc nhiều tiêu chí

4. **Khi nào nên sử dụng**
   - **Dùng Comparable:** Khi muốn định nghĩa thứ tự tự nhiên cho đối tượng và đảm bảo nó luôn có sẵn
   - **Dùng Comparator:** Khi cần sắp xếp theo nhiều tiêu chí khác nhau hoặc không muốn chỉnh sửa lớp của đối tượng

Link tài nguyên đọc thêm: https://www.geeksforgeeks.org/comparable-vs-comparator-in-java