Comparable结合PriorityQueue

`Comparable`接口是Java中的一个接口，它包含一个方法`compareTo()`，用于比较当前对象与另一个对象的大小关系。该接口允许类的实例具有内在的自然顺序。如果一个类实现了`Comparable`接口，就意味着该类的对象可以相互比较，可以用于排序等操作。

`Comparable`接口的定义如下：

**public interface Comparable<T> {**

**int compareTo(T o);**

**}**

```

**`compareTo()`**方法返回一个整数值，具体含义如下：

- 如果当前对象小于另一个对象，返回负整数。

- 如果当前对象等于另一个对象，返回0。

- 如果当前对象大于另一个对象，返回正整数。

通过实现`Comparable`接口，你可以控制对象的排序顺序。例如，如果你有一个`Person`类，你可以让它实现`Comparable`接口，然后在`compareTo()`方法中定义按照年龄、姓名等属性进行比较的逻辑。

以下是一个使用`Comparable`接口的示例：

```java

**class Person implements Comparable<Person> {**

**String name;**

**int age;**

**public Person(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**@Override**

**public int compareTo(Person otherPerson) {**

**// 按照年龄升序排序**

**return this.age - otherPerson.age;**

**}**

**}**

在这个示例中，`Person`类实现了`Comparable`接口，并按照年龄进行比较。`compareTo()`方法的返回值决定了排序的顺序。

***假设 当前值 比 其他值 更接近队列出口***

如果***比较方法***的值＞0，正数=> 让其他值和当前值***位置交换***，其他值更接近队列出口。

如果一个类实现了`**Comparable**`接口，并且你将该类的对象加入到`**PriorityQueue**`（优先级队列）中，那么优先级队列会使用`**compareTo()**`方法来确定对象的顺序，从而影响排序。

`**PriorityQueue`**在内部使用堆（heap）数据结构来维护元素的顺序。当你将一个实现了`**Comparable`**接口的对象加入到`**PriorityQueue**`时，优先级队列会根据对象的`**compareTo()**`方法的返回值来调整堆，确保堆顶元素是最小（或最大）的元素，具体取决于你的比较逻辑。

例如，如果你有一个`**Person**`类实现了`**Comparable**`接口，其中的比较逻辑是按照年龄升序排列：

```java

**class Person implements Comparable<Person> {**

**String name;**

**int age;**

**public Person(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**@Override**

**public int compareTo(Person otherPerson) {**

**// 按照年龄升序排序**

**return this.age - otherPerson.age;**

**}**

**}**

```

然后，当你将`Person`对象加入到`PriorityQueue`时：

```java

**PriorityQueue<Person> priorityQueue = new PriorityQueue<>();**

**priorityQueue.add(new Person("Alice", 30));**

**priorityQueue.add(new Person("Bob", 25));**

```

在这个例子中，`**PriorityQueue**`会使用`Person`对象的`**compareTo()**`方法来确定堆中的顺序，确保堆顶的元素是年龄最小的`Person`对象。因此，当你从`**PriorityQueue**`中**取出元素时，会按照年龄升序排列。**