JAVA并发编程随笔【一】PriorityBlockingQueue优先级队列

一、优先级队列PriorityBlockingQueue必须是实现Comparable接口，队列通过这个接口的compare方法确定对象的priority。当前和其他对象比较，如果compare方法返回负数，那么在队列里面的优先级就比较搞

    比较规则：当前对象和其他对象做比较，当前优先级大就返回-1，优先级小就返回1

二、优先级队列是一个基于堆的无界并发安全的优先级队列。

三、优先级队列不允许null值，不允许未实现Comparable接口的对象。

四、优先级中传入的实体对象

---------------------

作者：随风yy

来源：CSDN

原文：https://blog.csdn.net/yaomingyang/article/details/77618816

版权声明：本文为博主原创文章，转载请附上博文链接！

**public** **class** User **implements** Comparable<User> {

**private** Integer priority;

**private** String username;

**public** Integer getPriority() {

**return** priority;

}

**public** **void** setPriority(Integer priority) {

**this**.priority = priority;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

@Override

**public** **int** compareTo(User user) {

//        System.out.println("比较结果"+this.priority.compareTo(user.getPriority()));

**return** **this**.priority.compareTo(user.getPriority());

}

}

测试类：

**import** java.util.Random;

**import** java.util.concurrent.PriorityBlockingQueue;

**public** **class** PriorityBlockQueueDemo {

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

PriorityBlockingQueue<User> queue = **new** PriorityBlockingQueue<User>();

**for** (**int** i = 0; i < 12; i++) {

User user = **new** User();

**int** max = 20;

**int** min = 10;

Random random = **new** Random();

**int** n = random.nextInt(max) % (max - min + 1) + min;

user.setPriority(n);

user.setUsername("我的第".concat(String.*valueOf*(i)).concat("天"));

queue.add(user);

}

**for** (**int** i = 0; i < 12; i++) {

User u = queue.poll();

System.***out***.format("优先级是：%d \t %s \n", u.getPriority(), u.getUsername());

}

}

}

