Priority Queue

Certainly! In Java, the PriorityQueue class is used to implement a priority queue, which is a queue where elements are ordered based on their priority. The priority is typically determined either by the natural ordering of the elements or by a custom comparator.

Here's an example that demonstrates the usage of PriorityQueue:

```
import java.util.PriorityQueue;

public class PriorityQueueExample {

    public static void main(String[] args) {
        // Create a priority queue of integers
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        // Add elements to the priority queue
        pq.add(5);
        pq.add(2);
        pq.add(8);
        pq.add(1);
        pq.add(3);

        // Remove elements from the priority queue
        while (!pq.isEmpty()) {
            int element = pq.poll();
            System.out.println("Removed: " + element);
        }
    }
}
```

In this example, we create a PriorityQueue object called pq to store integers. We add some elements to the priority queue using the add() method. The elements are automatically ordered based on their natural ordering (in this case, ascending order).

We then use a while loop to remove elements from the priority queue using the poll() method. The poll() method removes and returns the head (i.e., the element with the highest priority) of the priority queue. The loop continues until the priority queue is empty (isEmpty() returns true).

The output of the above example will be:

makefile
Copy code
Removed: 1
Removed: 2
Removed: 3
Removed: 5
Removed: 8
As you can see, the elements are removed in ascending order because the priority queue orders them based on their natural ordering.

You can also provide a custom comparator to the PriorityQueue constructor to define a specific ordering for the elements. This allows you to prioritize elements based on your own criteria.

Note that the PriorityQueue class is part of the java.util package, so you need to import it before using it in your code.

```
package com.hdfc.operator;


//Queue - FIFO
//first in first out

import com.hdfc.collections.User;

import java.util.Comparator;
import java.util.Date;
import java.util.List;
import java.util.PriorityQueue;

class UserIdComparator implements Comparator<User> {

    @Override
    public int compare(User o1, User o2) {
        if (o1.getId() == o2.getId()) {
            return 0;
        } else if (o1.getId() < o2.getId()) {
            return 1;
        } else {
            return +1;
        }
    }
}

//5 2 1 3 6
//1 2 3 4 5
public class PriorityQueueTest {
    public static void main(String[] args) {

        //User class should implements Comparable interface
        // If User class is from library then? Comparator interface
        PriorityQueue<User> priorityQueue = new PriorityQueue<>(new UserIdComparator());
        priorityQueue.add(new User(5, "F", List.of("F"), new Date()));
        priorityQueue.add(new User(3, "C", List.of("C"), new Date()));
        priorityQueue.add(new User(1, "A", List.of("A"), new Date()));
        priorityQueue.add(new User(2, "B", List.of("B"), new Date()));

        System.out.println(priorityQueue.size());
        System.out.println(priorityQueue.poll()); //poll get and remove
        System.out.println(priorityQueue.size());

        System.out.println(priorityQueue.peek()); //just check the first elemetn in queue
        System.out.println(priorityQueue.size());

        /*PriorityQueue<Integer> priorityQueue = new PriorityQueue<>();
        priorityQueue.add(5);
        priorityQueue.add(2);
        priorityQueue.add(1);
        priorityQueue.add(3);
        priorityQueue.add(6);

        System.out.println(priorityQueue.poll());
        System.out.println(priorityQueue.poll());
        System.out.println(priorityQueue.poll());
        System.out.println(priorityQueue.poll());
        System.out.println(priorityQueue.poll());*/

    }
}
```

