Priority Queue

- Some times, you want a linear data structure that provides neither the first task you added nor the most recent task.
- Consider a system that manages service tickets for an IT department.
- Tickets can be stored in a queue, which has the first-in-first-out behavior.
  - *enqueue* adds a task to end of the queue.
  - *dequeue* removes and returns the oldest task.
- Queue is not sufficient if tickets also carry *priorities*.
- dequeue should return an element with the *highest priority*.
- For elements with the same priority, the oldest one is serviced.
Priority Queue

• Priority defines an ordering for elements stored in the queue.
  • For example, tickets can have numeric values. Numbers define the priority to order tickets.

• The priority queue provides the following operations:
  ▪ boolean add(T v): adds value v to the queue by respecting its priority.
  ▪ T remove(): removes and returns the highest-priority element.
  ▪ T element(): returns, but doesn’t remove, the highest-priority element.
Priority Queue

- Consider adding tickets to an empty priority queue.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Element</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>add</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>add</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>element</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>add</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>element</td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>
Priority Queue

• To establish order among the elements of a priority queue, the elements must be *comparable*.
  ▪ By default, priority queue stores the smallest element at the front of the queue.
  ▪ To decide a different kind of priority, make a *Comparator* object.
  ▪ Supply the object to the constructor of priority queue.

• Sorting is easy. Add all of the elements to priority queue. Then, read them back in order.
Example

```java
public class PriorityQueueDemo {
    public static <T extends Comparable<? Super T>>
    void dumpPQ( String msg, PriorityQueue<T> pq ) {

        System.out.println( msg + ":" );
        while( !pq.isEmpty( ) ) {
            System.out.print( pq.remove( ) + " " ) ;
        }
    }

    public static void main( String[] args ) {
        PriorityQueue<Integer> minPQ = new PriorityQueue<Integer> ( );
        minPQ.add( 4 );
        minPQ.add( 3 );
        minPQ.add( 5 );
        dumpPQ( "minPQ", minPQ ); // 3 4 5
    }
}
```