UnorderedListPriorityQueue.java

```java
/* Vincent Chan
 * masc0264
 */
package data_structures;

import java.util.Iterator;

public class UnorderedListPriorityQueue<E> implements PriorityQueue<E> {
  /* Functions Included
   * ====PUBLIC====
   * constructors       //Ln. 26
   * insert(object)     //Ln. 31
   * remove()           //Ln. 37
   * peek()             //Ln. 43
   * contains(Object)   //Ln. 63
   * size()             //Ln. 68
   * clear()            //Ln. 73
   * isEmpty()          //Ln. 78
   * isFull()           //Ln. 83
   * iterator()         //Ln. 89
   */

  //Variable declarations
  UnorderedList<E> list;

  //Constructor
  public UnorderedListPriorityQueue() {
    list = new UnorderedList<E>();
  } //End constructor

  //Inserts a new object. Returns true if insertion is succesful
  public boolean insert(E object) {
    list.addFirst(object);
    return true;
  } //End insert()

  //Removes the object with highest priority
  //Returns null if the list is empty
  public E remove() {
    return list.remove(peek());
  }//End remove()

  //Gets the object with the highest priority
  //the longest but does NOT remove it.
  //returns null if empty
  public E peek() {
    if(list.isEmpty()) return null;
    int comp;
    E lowPri = null;
    for(E current : list) {
      if(lowPri==null) {
        lowPri = current;
        continue;
      }
      comp = ((Comparable<E>)lowPri).compareTo(current);
      if(comp>0) {
        lowPri = current;
      }
    }
    return lowPri;
  } //End peek()

  //Returns true if the queue contains the specified element
  public boolean contains(E obj) {
```

```
65      return list.find(obj)!=null;
66    } //End contains()
67
68    //Returns the number of objects in the queue.
69    public int size() {
70      return list.size();
71    }
72
73    //returns to an empty list.
74    public void clear() {
75      list = new UnorderedList<E>();
76    } //End clear()
77
78    //Returns true if queue is empty
79    public boolean isEmpty() {
80      return list.isEmpty();
81    } //End isEmpty()
82
83    //Returns true if full. otherwise false.
84    //Linked lists are never full.
85    public boolean isFull() {
86      return false;
87    } //End isFull()
88
89    //Returns an iterator of the objects in the PQ,
90    public Iterator<E> iterator() {
91      return list.iterator();
92    } //End iterator()
93  } //End UnorderedListPriorityQueue
```