```java
/**
 *  Program 2
 *  Queue is an implementation of LinearList
 *  CS310-01
 *  3/13/2019
 *  @author  Karl Parks cssc1506
 */

package data_structures;

import java.util.Iterator;

public class Queue<E extends Comparable<E>> implements Iterable<E>{
    private LinearList<E> list;

    public Queue() {
      list = new LinearList<E>();
    }
  /*inserts the object obj into the queue
   */
  public void enqueue(E obj) {
    list.addLast(obj);
    return;
  }

   /* removes and returns the object at the front of the queue
   */
  public E dequeue() {
    return list.removeFirst();
  }

   /* returns the number of objects currently in the queue
   */
  public int size() {
    return list.size();
  }

   /* returns true if the queue is empty, otherwise false
   */
  public boolean isEmpty() {
    return list.isEmpty();
  }

   /* returns but does not remove the object at the front of the queue
```

```
45      */
46     public E peek() {
47       return list.peekFirst();
48     }
49
50      /* returns true if the Object obj is in the queue
51      */
52     public boolean contains(E obj) {
53       return list.contains(obj);
54     }
55
56      /* returns the queue to an empty state
57      */
58     public void makeEmpty() {
59       list.clear();
60     }
61
62      /* removes the Object obj if it is in the queue and
63      * returns true, otherwise returns false.
64      */
65     public boolean remove(E obj) {
66       boolean tmp = list.contains(obj);
67       list.remove(obj);
68       return tmp;
69     }
70
71      /* returns an iterator of the elements in the queue. The elements
72      * must be in the same sequence as dequeue would return them.
73      */
74
75     @Override
76     public Iterator<E> iterator() {
77       return list.iterator();
78     }
79 }
```