Queue Interface[*] and Implementation

```
public interface Queue
{
    // postcondition: returns true if queue is empty, false otherwise
  boolean isEmpty();

    // precondition:  queue is [e1, e2, ..., en] with n >= 0
    // postcondition: queue is [e1, e2, ..., en, x]
  void enqueue(Object x);

    // precondition:  queue is [e1, e2, ..., en] with n >= 1
    // postcondition: queue is [e2, ..., en]; returns e1
    //                throws an unchecked exception if the queue is empty
  Object dequeue();

    // precondition:  queue is [e1, e2, ..., en] with n >= 1
    // postcondition: returns e1
    //                throws an unchecked exception if the queue is empty
  Object peekFront();
}

public class ListQueue implements Queue
{
  private java.util.LinkedList list;

  public ListQueue() { list = new java.util.LinkedList(); }
  public boolean isEmpty() { return list.size() == 0; }
      // Or: ... isEmpty() { return list.isEmpty(); }
  public void enqueue(Object obj) { list.addLast(obj); }
  public Object dequeue() { return list.removeFirst(); }
  public Object peekFront() { return list.getFirst(); }
}
```

[*] Adapted from the College Board's *AP Computer Science AB: Implementation Classes and Interfaces*.