```
package queues;

import java.util.Enumeration;

/**
 * A queue implemented using arrays.
 *
 * Version 1.
 *
 * @author Alan L. Cox
 * @since 03/15/01
 */
public class ArrayVers1Queue implements IQueue {

    private int      _firstEmptyObject = 0;
    private Object[] _objects = new Object[1];

    /**
     * param object the object to enqueue
     */
    public void enqueue(Object object)
    {
        if (_firstEmptyObject == _objects.length) {

            int j;

            Object[] newObjects = new Object[2*_objects.length];

            for (j = 0; j < _objects.length; j++)
                newObjects[j] = _objects[j];

            _objects = newObjects;
        }
        _objects[_firstEmptyObject] = object;
        _firstEmptyObject++;
    }

    /**
     * @return the dequeued object
     */
    public Object dequeue()
    {
        if (_firstEmptyObject == 0)
            throw new java.util.NoSuchElementException("The queue is empty.");

        Object object = _objects[0];

        _firstEmptyObject--;

        // Java note: i's scope is limited to the for loop and its body.
        for (int i = 0; i < _firstEmptyObject; i++)
            _objects[i] = _objects[i + 1];

        return object;
    }

    /**
     * @return an Enumeration of the queue.
     */
    public Enumeration enumeration()
    {
        return new Enumeration() {

            private int _nextObject = 0;

            public boolean hasMoreElements()
            {
                return _nextObject < _firstEmptyObject;
            }

            public Object nextElement()
            {
                return _objects[_nextObject++];
            }
        };
    }
}
```

```
package queues;

import java.util.Enumeration;

/**
 * A queue implemented using arrays.
 *
 * Version 2.
 *
 * @author Alan L. Cox
 * @since 03/15/01
 */
public class ArrayVers2Queue implements IQueue {

    private int      _firstObject = 0;
    private int      _lastObjectPlusOne = 0;
    private Object[] _objects = new Object[1];
    private int      _objectsQueued = 0;

    /**
     * param object the object to enqueue
     */
    public void enqueue(Object object)
    {
        if (_objectsQueued == _objects.length) {

            int j;

            Object[] newObjects = new Object[2*_objects.length];

            _lastObjectPlusOne += _objects.length;

            for (j = _firstObject; j < _lastObjectPlusOne; j++)
                newObjects[j] = _objects[j % _objects.length];

            _objects = newObjects;
        }
        _objects[_lastObjectPlusOne] = object;
        _objectsQueued++;
        _lastObjectPlusOne++;
        _lastObjectPlusOne %= _objects.length;
    }

    /**
     * @return the dequeued object
     */
    public Object dequeue()
    {
        if (_objectsQueued == 0)
            throw new java.util.NoSuchElementException("The queue is empty.");

        Object object = _objects[_firstObject];
        _objectsQueued--;
        _firstObject++;
        _firstObject %= _objects.length; // Java note: the mod operator

        return object;
    }

    /**
     * @return an Enumeration of the queue.
     */
    public Enumeration enumeration()
    {
        return new Enumeration() {

            private int _nextObject = _firstObject;

            public boolean hasMoreElements()
            {
                return _nextObject < _lastObjectPlusOne;
            }

            public Object nextElement()
            {
                return _objects[_nextObject++];
            }
        };
    }
}
```

```
package queues;

import java.util.Enumeration;

/**
 * A queue interface
 *
 * A queue manages its objects in a First In, First Out (FIFO) order.
 *
 * @author Alan L. Cox
 * @since 03/15/01
 */
public interface IQueue {

    /**
     * Adds a new object to the queue.
     */
    public void enqueue(Object data);

    /**
     * Removes and returns the oldest object from the queue.
     */
    public Object dequeue();

    /**
     * Returns an Enumeration of the stack.
     */
    public Enumeration enumeration();
}
```

```
package stacks;

import java.util.Enumeration;

/**
 * A stack interface
 *
 * A stack manages its objects in a Last In, First Out (LIFO) order.
 *
 * @author Alan L. Cox
 * @since 03/15/01
 */
public interface IStack {

    /**
     * Adds a new object to the stack.
     */
    public void push(Object data);

    /**
     * Removes and returns the newest object from the stack.
     */
    public Object pop();

    /**
     * Returns an Enumeration of the stack.
     */
    public Enumeration enumeration();
}
```