

```












10 package exercise01;
11
12 import java.util.Iterator;
13
14 // suppress unchecked warnings in Java 1.5.0_6 and later
15 @SuppressWarnings("unchecked")
16
17 public class RingBuffer<Item> implements Iterable<Item> {
18     private Item[] a; // queue elements
19     private int N = 0; // number of elements on queue
20     private int first = 0; // index of first element of queue
21     private int last = 0; // index of next available slot
22
23     // cast needed since no generic array creation in Java
24     public RingBuffer(int capacity) {
25         a = (Item[]) new Object[capacity];
26     }
27
28     public boolean isEmpty() { return N == 0; }
29     public int size() { return N; }
30
31     public void enqueue(Item item) throws RingBufferException {
32         if (N == a.length) { throw new RingBufferException("Ring buffer overflow"); }
33         a[last] = item;
34         last = (last + 1) % a.length; // wrap-around
35         N++;
36     }
37
38     // remove the least recently added item - doesn't check for underflow
39     public Item dequeue() throws RingBufferException {
40         if (isEmpty()) { throw new RingBufferException("Ring buffer underflow"); }
41         Item item = a[first];
42         a[first] = null; // to help with garbage collection
43         N--;
44         first = (first + 1) % a.length; // wrap-around
45         return item;
46     }
47
48     public Iterator<Item> iterator() { return new RingBufferIterator(); }
49
50     // an iterator, doesn't implement remove() since it's optional
51     public class RingBufferIterator implements Iterator<Item> {
52         private int i = 0;
53         public boolean hasNext() { return i < N; }
54         public void remove() { throw new UnsupportedOperationException(); }
55
56         public Item next() {
57             if (!hasNext()) throw new NoSuchElementException();
58             return a[i++];
59         }
60     }
61 }
62
63 }

```

```

1 package exercise01;
2
3 @SuppressWarnings("serial")
4 public class RingBufferException extends Exception {
5
6     public RingBufferException(String msg) {
7         super(msg);
8     }
9
10 }
11

```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼  SoftwareTesting_HW_1	 92,1 %	434	37	471
>  src/test/java	 88,5 %	285	37	322
▼  src/main/java	 100,0 %	149	0	149
▼  exercise01	 100,0 %	149	0	149
>  RingBuffer.java	 100,0 %	145	0	145
>  RingBufferException.java	100,0 %	4	0	4