1) MyRWSynchronizedList.java

```
1  package proxy.e3;
2
3  import java.util.concurrent.locks.ReentrantReadWriteLock;
4
5  class MyRWSynchronizedList implements MyList {
6      MyList list;
7      ReentrantReadWriteLock lock = new ReentrantReadWriteLock();
8
9      MyRWSynchronizedList(MyList list) {
10         this.list = list;
11     }
12
13     @Override
14     public int getCount() {
15         lock.readLock().lock();
16         try {
17             return list.getCount();
18         } finally {
19             lock.readLock().unlock();
20         }
21     }
22
23     @Override
24     public MyObject getAt(int index) {
25         lock.readLock().lock();
26         try {
27             return list.getAt(index);
28         } finally {
29             lock.readLock().unlock();
30         }
31     }
32
33     @Override
34     public void setAt(int index, MyObject data) {
35         lock.writeLock().lock();
36         try {
37             list.setAt(index, data);
38         } finally {
39             lock.writeLock().unlock();
40         }
41     }
42
43     @Override
44     public void insertAt(int index, MyObject data) {
45         lock.writeLock().lock();
46         try {
47             list.insertAt(index, data);
48         } finally {
49             lock.writeLock().unlock();
50         }
51     }
52
53     @Override
54     public void removeAt(int index) {
55         lock.writeLock().lock();
56         try {
57             list.removeAt(index);
58         } finally {
59             lock.writeLock().unlock();
60         }
61     }
62
63     @Override
64     public int findIndex(MyObject data) {
65         lock.readLock().lock();
66         try {
67             return list.findIndex(data);
68         } finally {
```

```
69                 lock.readLock().unlock();
70             }
71         }
72     }
```

2) Example3.java

```
1  package proxy.e3;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class Example3 {
7
8      static void work(MyList list) {
9          for (int i=0; i < 1000; ++i) {
10             list.insertAt(0, new MyInt(999));
11             list.removeAt(0);
12         }
13     }
14
15     static void add(MyList list, int count) {
16         for (int i = 0; i < count; ++i)
17             list.add(new MyInt(i));
18     }
19
20     static void print(MyList list) {
21         System.out.printf("Count: %dWn", list.getCount());
22         for (int i = 0; i < list.getCount(); ++i)
23             System.out.printf("%s ", list.getAt(i));
24         System.out.println();
25     }
26
27     static void doSomething(MyList list) throws Exception {
28         List<Thread> threads = new ArrayList<>();
29         add(list, 100);
30         for (int i = 0; i < 100; ++i) {
31             Thread t = new Thread(() -> work(list));
32             t.start();
33             threads.add(t);
34         }
35         for (Thread t: threads)
36             t.join();
37         print(list);
38     }
39
40     public static void main(String[] args) throws Exception {
41         doSomething(new MyRWSynchronizedList(new MyArrayList()));
42         doSomething(new MyRWSynchronizedList(new MyLinkedList()));
43     }
44 }
```

기존의 코드에서 수정된 부분은, 객체들을 생성하여 조립하는 코드 뿐이다.