1. Add():

**public** **boolean** add(E e) {

**final** ReentrantLock lock = **this**.lock;

lock.lock();

**try** {

Object[] elements = getArray();

**int** len = elements.length;

Object[] newElements = Arrays.*copyOf*(elements, len + 1);

newElements[len] = e;

setArray(newElements);

**return** **true**;

} **finally** {

lock.unlock();

}

}

NB : Arrays.*copyOf means creating a new array instance and copying all the elements from the old array instance.*

1. Set() :

**public** E set(**int** index, E element) {

**final** ReentrantLock lock = **this**.lock;

lock.lock();

**try** {

Object[] elements = getArray();

E oldValue = get(elements, index);

**if** (oldValue != element) {

**int** len = elements.length;

Object[] newElements = Arrays.*copyOf*(elements, len);

newElements[index] = element;

setArray(newElements);

} **else** {

// Not quite a no-op; ensures volatile write semantics

setArray(elements);

}

**return** oldValue;

} **finally** {

lock.unlock();

}

}

1. Remove() :

**public** E remove(**int** index) {

**final** ReentrantLock lock = **this**.lock;

lock.lock();

**try** {

Object[] elements = getArray();

**int** len = elements.length;

E oldValue = get(elements, index);

**int** numMoved = len - index - 1;

**if** (numMoved == 0)

setArray(Arrays.*copyOf*(elements, len - 1));

**else** {

Object[] newElements = **new** Object[len - 1];

System.*arraycopy*(elements, 0, newElements, 0, index);

System.*arraycopy*(elements, index + 1, newElements, index,

numMoved);

setArray(newElements);

}

**return** oldValue;

} **finally** {

lock.unlock();

}

}

1. Remove() :