CopyOnWrite机制称为写时复制，就是执行修改操作时进行底层数组复制，使得修改操作在新的数组上进行，不妨碍原数组的并发读操作，复制修改完成后更新原数组引用变量。如果有多个调用者同时要求相同的资源，他们会共同获取相同的指针指向相同的资源，直到某个调用者试图修改资源内容时，系统才会真正复制到一份专用副本给该调用者，而其他调用者所见到的最初资源保持不变，这过程对其他调用者都是透明的，此做法的主要优点是如果调用者没有资源，就不会有副本被创建，因此多个调用者只是读取操作时可以共享同一份资源。

其中CopyOnWriteArratList和CopyOnWriteArraySet实现了这个并发容器
好处：因为写时是在复制的一份上操作，所以可以并发的读，不需要加锁，是读写分离的思想，在并发场景中使用。
总结：线程安全的，读操作不需要加锁。
适应于读多写少的情况且脏读的影响不大的并发情况，建议使用CopyOnWrite.

举个CopyOnWriteArrayList例子
构造函数：

```
public CopyOnWriteArrayList(){
  setArray(new Object[0]);
}

public boolean add(E e){
final ReentranLock lock = this.lock;
lock.lock();
try{
   Object [] elements = getArray();
    int len = elements.length;
    Object [] newElements = Arrays.copyof(elements, len + 1); //复制到一个新数组中，容量+1
    setArray(newElements); //将原数组的引用指向新数组
     return true;
}finally {
 lock.unlock();
}
}
```

```
//读操作，不用加锁
public E get(int index){
  return (E) getArray()[index]
}
```