NestedMonitor

Simply remove the synchronization in DisplayBuffer!

```
public class DisplayBuffer implements Buffer {
  public synchronized void put( Object o )
      throws InterruptedException
  {
    empty_.down();
    buffer_.put( o );
    updateView();
    full_.up();
  }
  public synchronized Object get()
    throws InterruptedException { ... }
}
```

Priority ReadersWriters

```
public class WritersPriorityReadWrite
  extends SafeReadWrite implements ReadWrite
{
  int waitingWriters_ = 0;
  public WritersPriorityReadWrite(ReaderWriterPanel view)
  { super(view); }
  public synchronized void acquireRead()
    throws InterruptedException
  {
    while ( waitingWriters_ > 0 || writing_ )
      wait();
    readers_++;
    view_.setReader( readers_ );
  }
```

```
    public synchronized void acquireWrite()
      throws InterruptedException
    {
      waitingWriters_++;
      super.acquireWrite();
    }
    public synchronized void releaseWrite()
    {
      waitingWriters_--;
      super.releaseWrite() ;
    }
  }
```

Fair ReadersWriters

```
public class FairReadWrite
  extends WritersPriorityReadWrite
  implements ReadWrite
{ boolean writersPriority_ = true;
  public FairReadWrite( ... view ) { super(view); }
  public synchronized void acquireRead()
    throws InterruptedException
  {
    while ( (waitingWriters_ > 0 && writersPriority_)
            || writing_ )
      wait();
    readers_++;
    view_.setReader( readers_ );
  }
```

```
    public synchronized void releaseRead()
    {
      writersPriority_ = true;
      super.releaseRead() ;
    }
    public synchronized void releaseWrite()
    {
      writersPriority_ = false;
      super.releaseWrite() ;
    }
  }
```

Safe SingleLaneBridge

Synchronize on the number of red/blue cars on the bridge

```
public class SafeBridge extends DefaultBridge
{
  protected int redCars_ = 0;
  protected int blueCars_ = 0;

  public synchronized void redEnter()
    throws InterruptedException {
    while ( blueCars_ > 0 )
      wait();
    redCars_ ++;
  }
```

```
  public synchronized void redExit() {
    redCars_ --;
    notifyAll();
  }
  public synchronized void blueEnter()
    throws InterruptedException {
    while ( redCars_ > 0 )
        wait();
    blueCars_ ++;
  }
  public synchronized void blueExit() {
    blueCars_ --;
    notifyAll();
  }
}
```

Fair SingleLaneBridge

Swap priority each time you get on the bridge.

```
public class FairBridge
  extends SafeBridge
{
    protected int redWaiting_ = 0;
    protected int blueWaiting_ = 0;
    protected boolean redTurn_ = true;
    protected boolean blueTurn_ = false;
```

```
  public synchronized void redEnter()
    throws InterruptedException
  {
    redWaiting_++;
    while ( blueCars_ > 0
          || ( blueWaiting_ > 0 && blueTurn_ ) )
        wait();
    redWaiting_--;
    blueTurn();
    redCars_ ++;
  }
  protected void blueTurn() {
    redTurn_ = false; blueTurn_ = true;
  }
...
```

Golf

Use simple grocery store ticket scheme.

```
public class FairAllocator extends SimpleAllocator
{ protected int counter_ = 0;
  protected int nowServing_ = 0;
  public FairAllocator(int n) { super(n); }
  synchronized public void get(int n)
    throws InterruptedException {
    int ticket = getTicket();
    while ( !myTurn(ticket) || n > available_ )
      wait();
    available_ -= n;
    leave();
  }
```

```
    protected int getTicket() {
      return counter_++;
    }
    protected boolean myTurn(int ticket) {
      return ticket == nowServing_;
    }
    protected void leave() {
      nowServing_++;
      notifyAll();
    }
    public String getTitle() {
      return "FairAllocator";
    }
  }
```