Exercise 7.1

1. see CasHistogram.java for implementation
   No Class State Escape is possible because all the state variables are private and the count array is declared as final, ensuring that it cannot be assigned to another object after initialization. Other threads can only access the bins array through the provided methods, which are synchronized with AtomicInteger.
   Safe publication is guaranteed because the count array is declared as final.
   no variable is immutable.

2. see TestHistogram.java for implementation

3. Performance testing:
   ```
   Using 1 threads:
           CAS Histogram            11575399,7 ns  191927,65        32
           Histogram2              12907112,5 ns 1331130,36        32
   Using 2 threads:
           CAS Histogram             8012980,5 ns  456934,58        64
           Histogram2              12462089,4 ns 5803547,26        32
   Using 4 threads:
           CAS Histogram             7351609,4 ns  913166,55        64
           Histogram2              10530119,4 ns 2361459,11        32
   Using 8 threads:
           CAS Histogram             7035580,5 ns 1724625,78        64
           Histogram2              12133223,8 ns  883105,71        32
   Using 16 threads:
           CAS Histogram             5589375,6 ns  680768,39        64
           Histogram2               11112511,6 ns  470297,05        32
   ```

we can clearly see that the CAS based implementation works significantly better especially when the number of threads increases.
The results are consistent with what we expected:

- CasHistogram benefits from non-blocking atomic operations (compareAndSet to update the counts), which allow for better parallelism and reduced contention, especially as the number of threads increases. that is also because they allow multiple threads to update different bins concurrently without locking the entire data structure.
- While Histogram2 uses synchronized methods. This can introduce contention and potentially slow down the performance when multiple threads are accessing the histogram concurrently.

Exercise 7.2

1. see ReadWriteCASLock for implementation
2. see ReadWriteCASLock for implementation
3. see ReadWriteCASLock for implementation
4. see ReadWriteCASLock for implementation

5. see TestLocks for implementation
6. see TestLocks for implementation