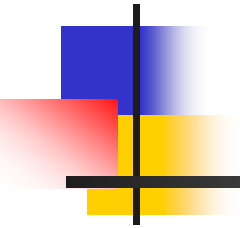
# An Efficient Algorithm for Concurrent Priority Queue Heaps -- Implementation

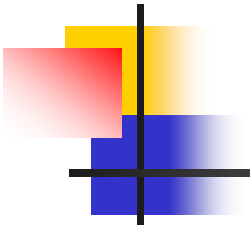


**Shouzheng Yang**

**CSE6490A**

**Class Presentation 2**

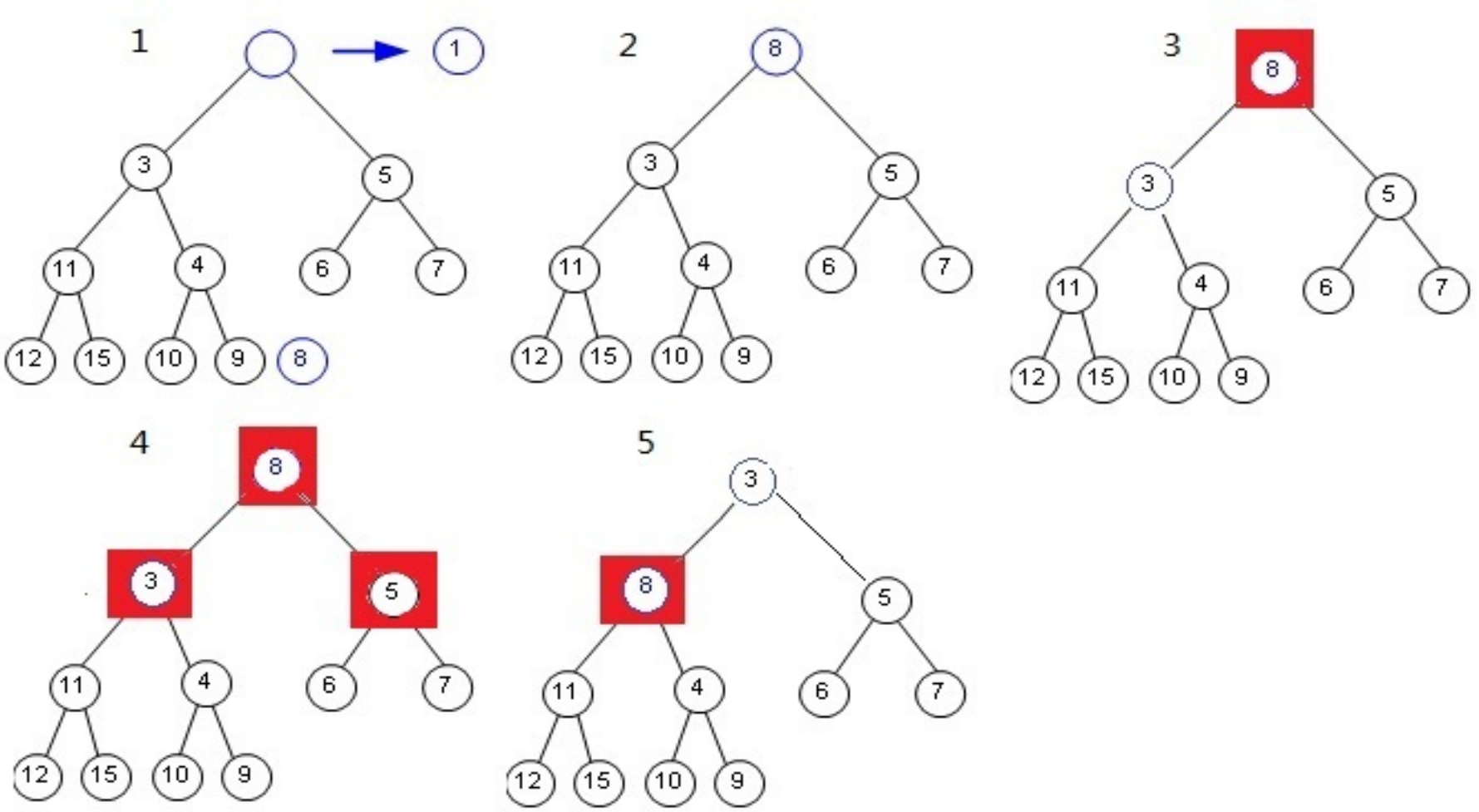
**Mar 14, 2011**



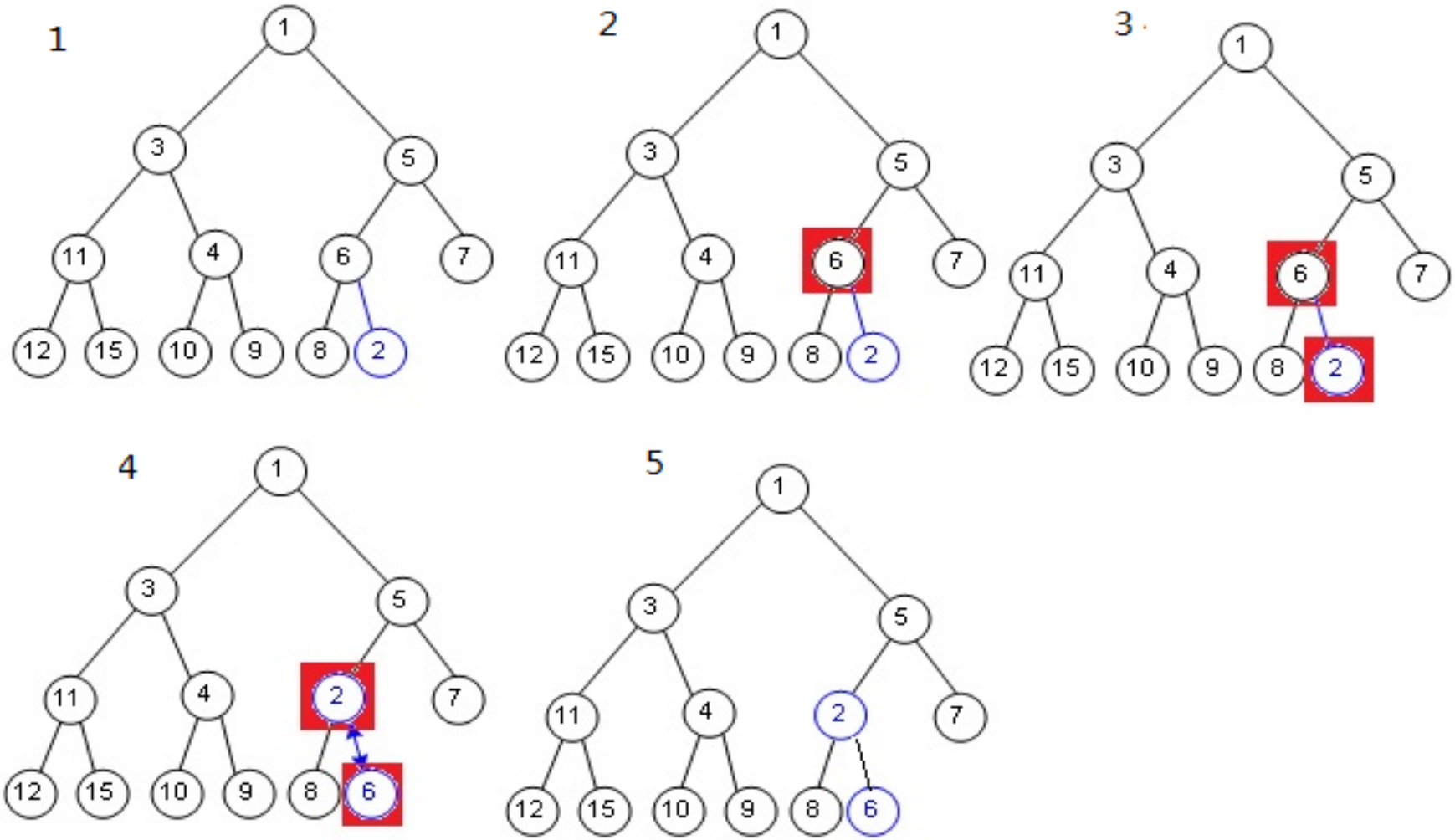
# Outline

- Algorithm Review
- Implementation
  - Class diagram
  - Mechanisms used for concurrency
  - Implementation refinement
- Testing
  - Input
  - Results
  - Analysis

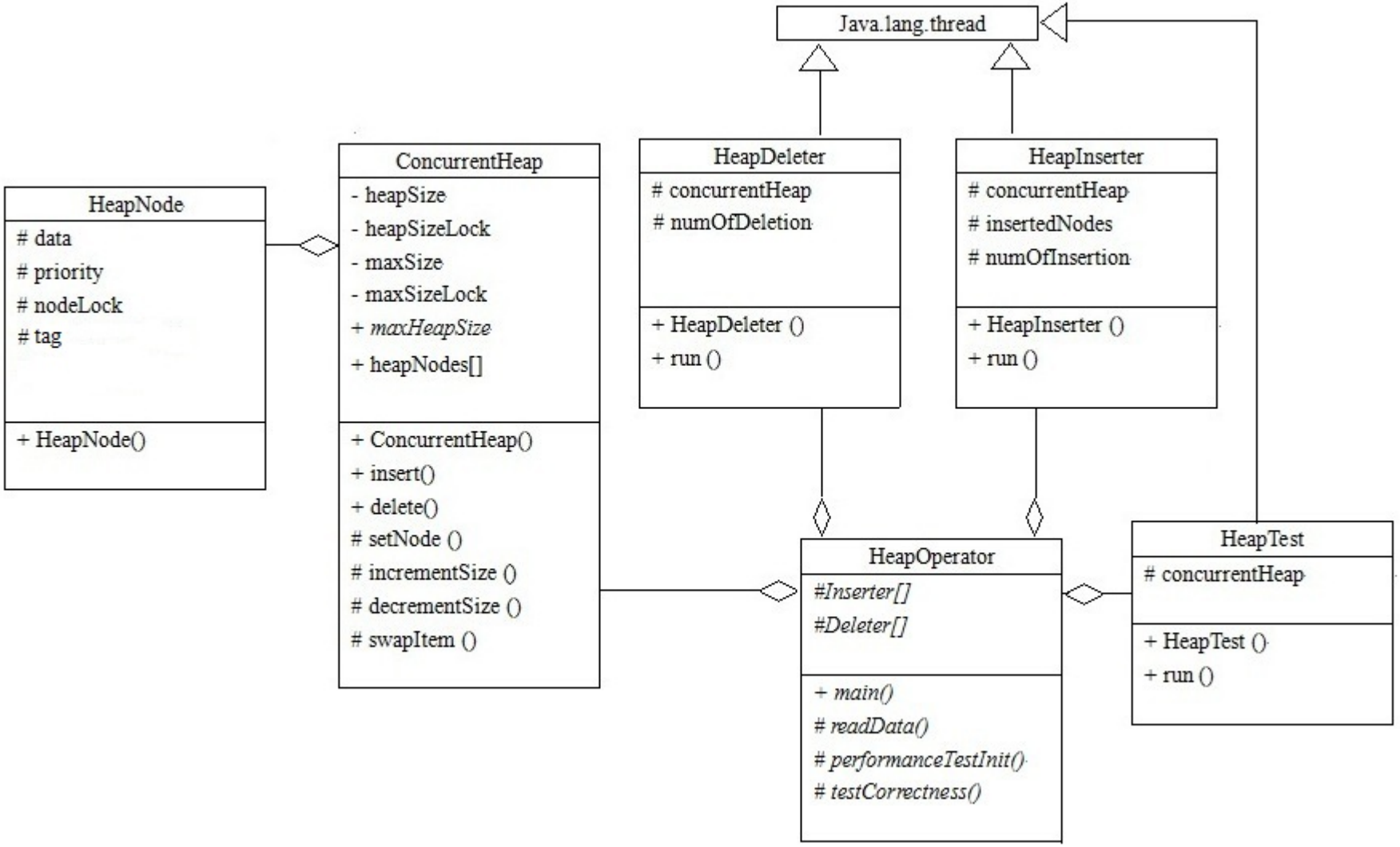
# Algorithm Review: Deletion



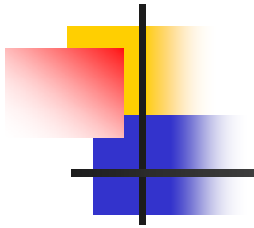
# Algorithm Review: Insertion



# Class Diagram

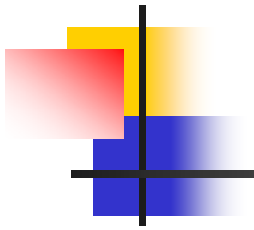


# Mechanisms Used for Concurrency



- Java.util.concurrent.locks.ReentrantLock
  - Fairness parameter
  - My own experience
  - try {
    reentrantLock.lock();
    do some stuff;
    }
    finally{
    reentrantLock.unlock();
    }
- Volatile keyword

# Mechanisms Used for Concurrency

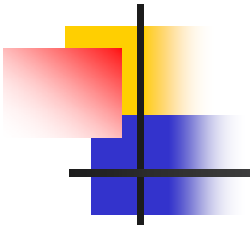


- Atomic variables

```
insertionCount=new AtomicInteger();
insertionCount.incrementAndGet();
```

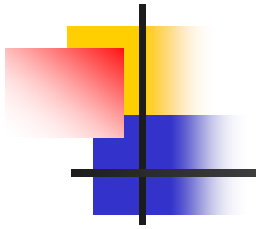
- CyclicBarrier

```
insertionStart=new CyclicBarrier(numOfInserters,
    new Runnable() {
      public void run() {
            insertionStartTime.set(System.nanoTime());
      }
});
```



# Code Refinement

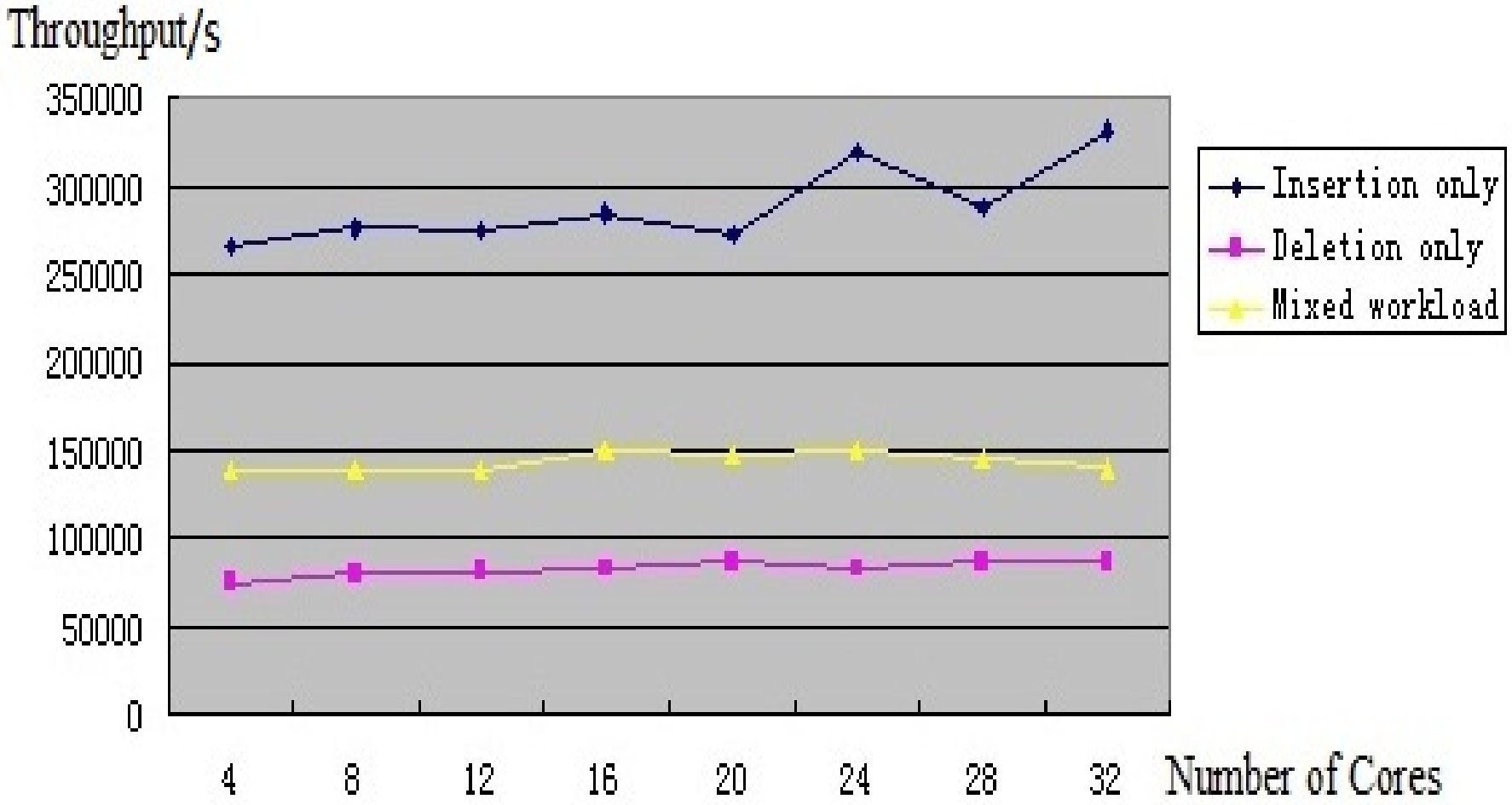
- All possible cases are explicitly stated.
- Lock won’t be acquired until we do immediately need it.



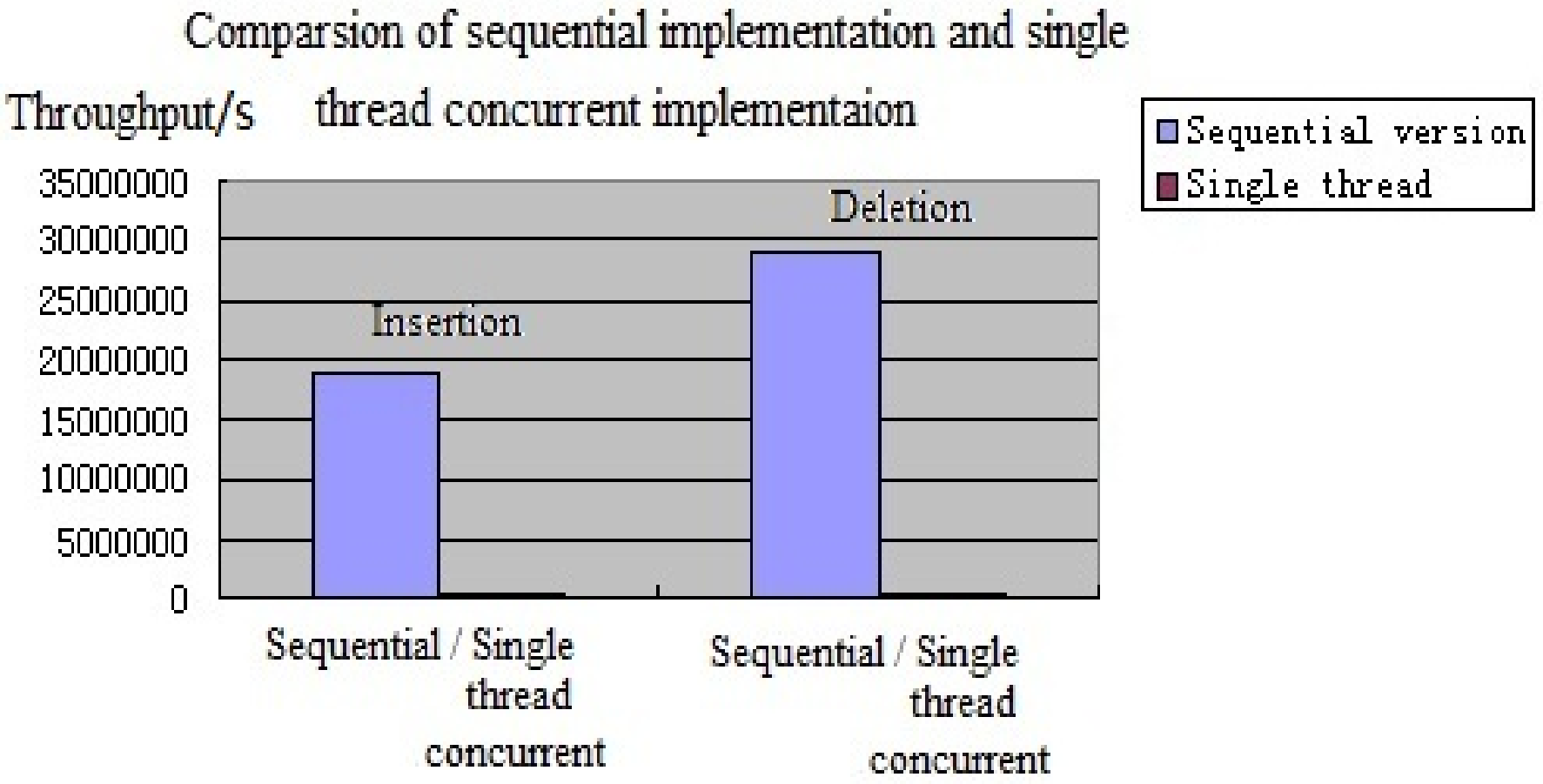
# Testing

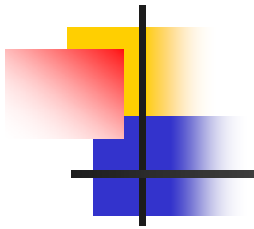
- Input
  - A huge number of random integers generated by Java.util.Random.
  - Read into memory before running the essential concurrent code.
- Experiment
  - Correctness
  - Throughput
  - Overhead of the concurrent implementation

# Result of Throughput



# Overhead of the Concurrent Implementation





# Looking Ahead

- ReentrantLock
  - Strongly suggested by Java API and Java Concurrency in Practice.

```
try {
   aReentrantLock.lock();
   do some stuff;
}
finally{
   aRentrantLock.unlock();
}
```

# Reference


- Hunt, G., Michael, M., Parthasarathy, S., Scott, M.: An efficient algorithm for concurrent priority queue heaps. Information Processing Letters 60(3) 151-157 ISSN: 0020-0190 1996, Elsevier.
- Java™ Platform Standard Ed. 6 API
- Joshua Bloch Joseph Bowbeer David Holmes Brian Goetz, Tim Peierls and Doug Lea. Java concurrency in practice. page 282, 2006.

Thank you very much!