Thread safe classes

- Atomics
- Synchronized vs Concurrent
- Concurrent collections
- Concurrent queues
- Concurrent maps

Atomics

java.util.concurrent.atomic

- AtomicBoolean
- AtomicInteger
- AtomicLong

Atomics are thread safe but without synchronize.

```
public final int incrementAndGet() {
        while (true) {
            int current = get(); //get() returns current value of volatile variable
            int next = current + 1;
            if (compareAndSet(current,  next)) return next;
        }
}
public final boolean compareAndSet(int expect, int update) {
        return unsafe.compareAndSwapInt(this,  valueOffset,  expect,  update);
}
```

Synchronized vs Concurrent

Synchronize

The resource which is synchronized can't be modified by multiple threads simultaneously.

Concurrent

Allows multiple threads to access different parts of a collection at a given time.

Concurrent collections

CopyOnWriteArrayList

A thread-safe variant of java.util.ArrayList in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

CopyOnWriteArraySet

Set wraper for CopyOnWriteArrayList

ConcurrentSkipListSet

A sorted container that can be accessed by multiple threads. This is essentially the equivalent of TreeSet for concurrent code.

Concurrent Queues

ConcurrentLinkedQueue

An unbounded thread-safe {@linkplain Queue queue} based on linked nodes.

Like most other concurrent collection implementations, this class does not permit the use of null elements.

ArrayBlockingQueue

A classic bounded buffer, in which a fixed-sized array holds elements inserted by producers and extracted by consumers.

Once created, the capacity cannot be changed.

LinkedBlockingQueue

An optionally-bounded blocking queue based on linked nodes.

Linked queues typically have higher throughput than array-based queues but less predictable performance in most concurrent applications.

Concurrent maps

ConcurrentHashMap

A hash table supporting full concurrency of retrievals and high expected concurrency for updates.

ConcurrentSkipListMap

A sorted container that can be accessed by multiple threads. This is essentially the equivalent of TreeMap for concurrent code.