HashMap不是线程安全的，Hashtable是线程安全的，但效率低，因为Hashtable是使用synchronize的，所有线程竞争同一把锁，而ConcurrentHashMap不仅线程安全而且效率高，因为它包含了一个segment数组，将数据分段存储，给每一段数据配一把锁，也就是所谓的锁分段技术。

HashMap为什么是线程不安全的?

答：HashMap在并发的时候主要有两个方面的问题，首先如果多个线程同时使用put方法添加元素，而且假设正好存在两个put的key发生了碰撞，那么根据HashMap的实现，这两个key会添加到数组的同一位置这样最终就会发生其中一个线程的put的数据被覆盖。第二就是如果多个线程同时检测到元素个数超过数组大小*loadFactor，这样最终只有一个线程扩容后的数组会赋给table，也就是说其他线程的都会丢失，并且各自线程put的数据也丢失。

如何线程安全的使用HashMap?

1.Hashtable

2.ConcurrentHashMap

3.SynchronizedMap

HashTable源码是使用synchronized来保证线程安全的。

```
public synchronized V get(Object key) {
        // 省略实现
    }
public synchronized V put(K key, V value) {
    // 省略实现
    }
```

当一个线程访问HashTable的同步方法时，其他线程如果也要访问同步方法，会被阻塞住。当一个线程使用put方法时，另一个线程不但不可以使用put方法，连get方法都不可以，所以效率很低。

调用synchronizedMap()方法后会返回一个SynchronizedMap类对象，而在SynchronizedMap类中使用了Synchronized同步关键字来保证对Map的操作时线程安全的。

其中ConcurrentHashMap的速度最快。