1. 为什么需要ThreadLocal（ThreadLocal的作用）

如果对象的成员变量不需要在多个线程间共享，且存在多线程访问，可以使用ThreadLocal，将变量设置为线程私有，每个线程保存一份副本，不同线程之间互不干扰。

1. 概述

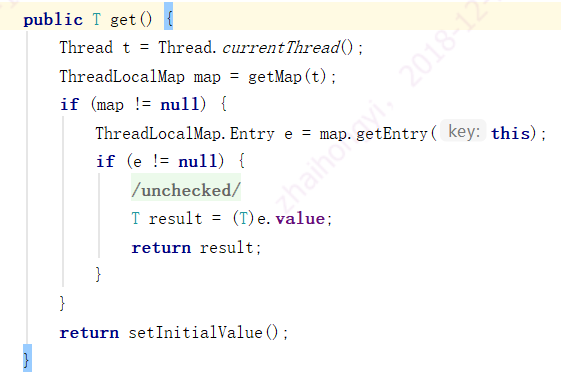
ThreadLocal提供了get(), set(value), remove(), initialValue()方法。

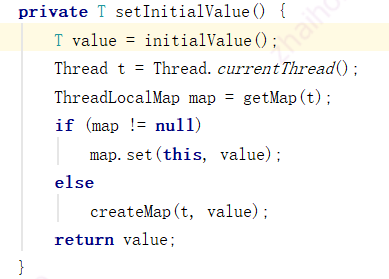
对于每个线程，可能包含多个ThreadLocal对象，因此线程中保存一个map（类型为ThreadLocalMap），map的键值是ThreadLocal，value是用户设置的value。键值是ThreadLocal的原因是，每增加一个ThreadLocal变量就需要在map中多一条记录，所以用ThreadLocal为key。因此使用ThreadLocal的多个线程中每个线程都会包含相同的key。

当调用某个threadLocal变量的get、set方法时，会使用this作为键。

1. 源码分析

3.1 get()





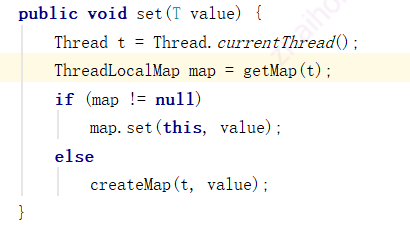
首先get，getMap(t)获取当前线程的ThreadLocalMap（threadLocals）。接着用this，即当前ThreadLocal对象为键值查找value，并返回。如果第一次会设置初始值即setInitialValue，默认会返回null。

默认initialValue返回null，同时会创建map，即new ThreadLocalMap()。

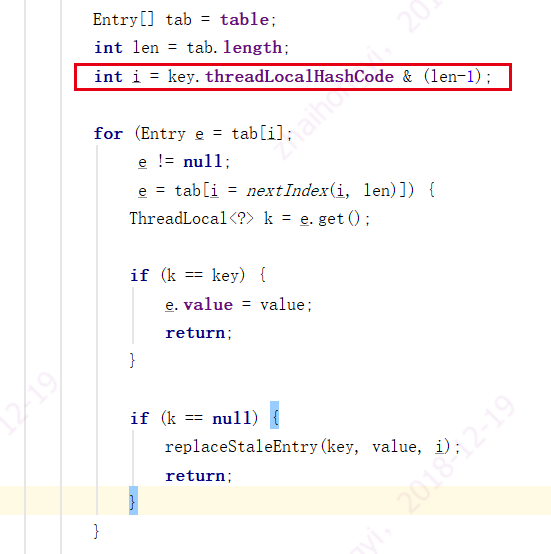
总的来说只要理解为什么用this作为键值，其他都很简单。

3.2 set()

ThreadLocal.set()



ThreadLocalMap.set()



set方法没啥好说的，这里你可以认为map.set(this, value)类似HashMap的put(k, v)方法。但ThreadLocalMap并没有依赖其他类，而使用Entry数组，自己实现读写。每个ThreadLocal都有threadLocalHashCode值，可以认为每个ThreadLocal对象的hashCode都不相同，这个值就是ThreadLocalMap中数组的偏移量。如果出现hash冲突，即e ！= null但key不相同，则继续向后查找Entry。