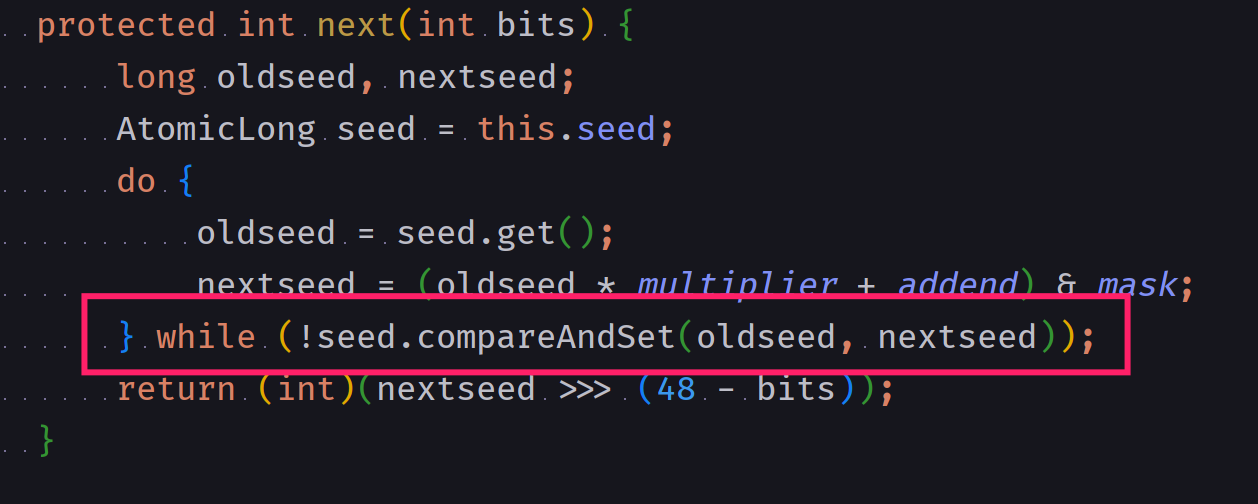
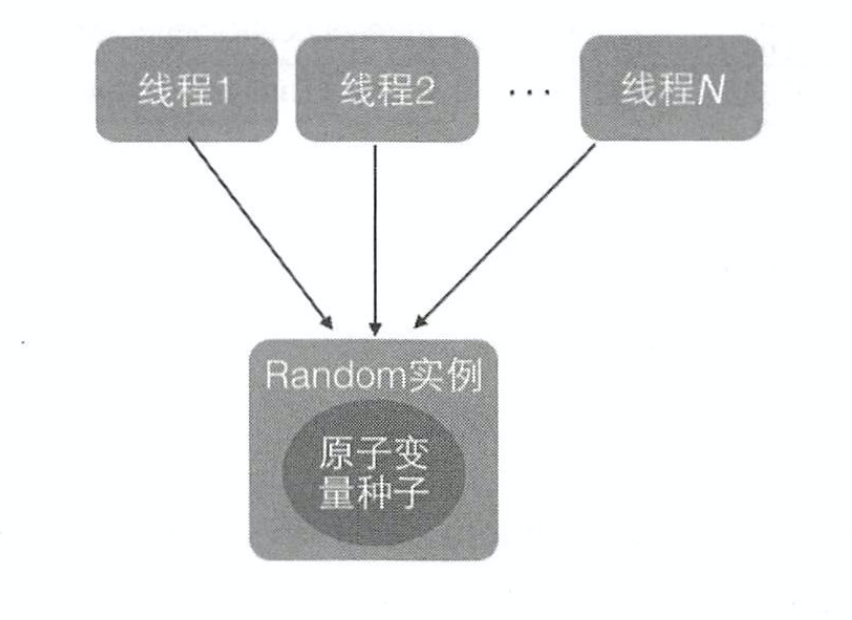
1. java.util.Random类next(int bits)方法实现



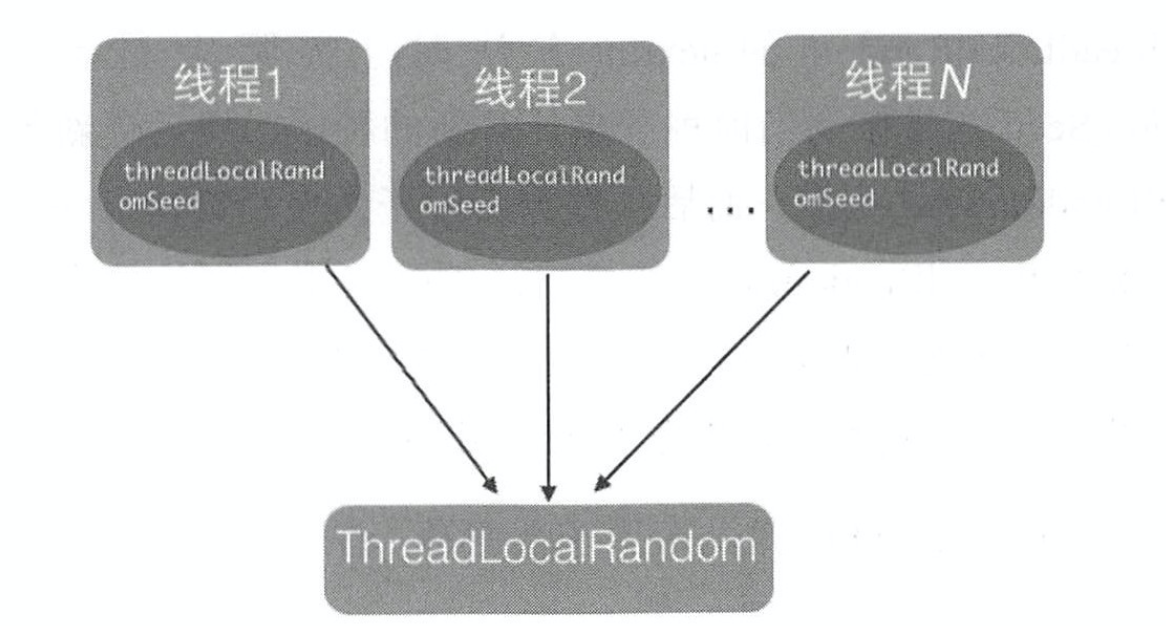
新随机数的生成要经过两个步骤

* 1. 根据老的种子生成新的种子
  2. 根据新种子计算新的随机数

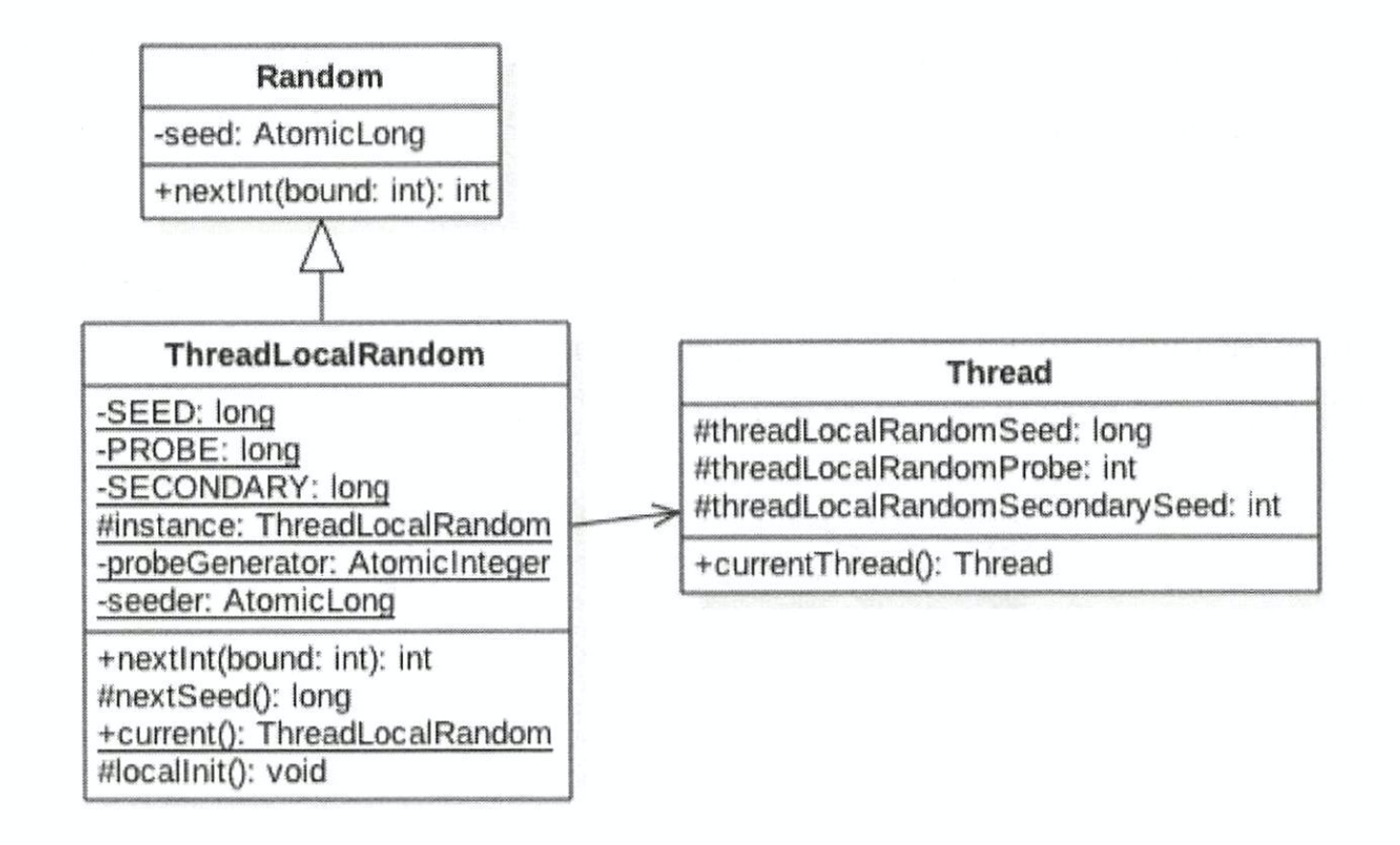
1. 每个Random实例里面都有一个原子性的种子变量private final AtomicLong seed用来记录当前的种子值，当要生成新的随机数时需要根据当前种子计算新的种子并更新回原子变量。多线程下会竞争同一个原子变量的更新操作，由于使用CAS操作，只有一个线程会成功，会造成大量线程进行自旋，降低并发性能，所以ThreadLocalRandom应运而生



1. java.util.concurrent.ThreadLocalRandom，每个线程维护一个种子变量，每个线程生成随机数时都根据自己老的种子计算新的种子，并使用新种子更新老的种子，再根据新种子计算随机数，就不会存在竞争问题了，大大提高并发性能

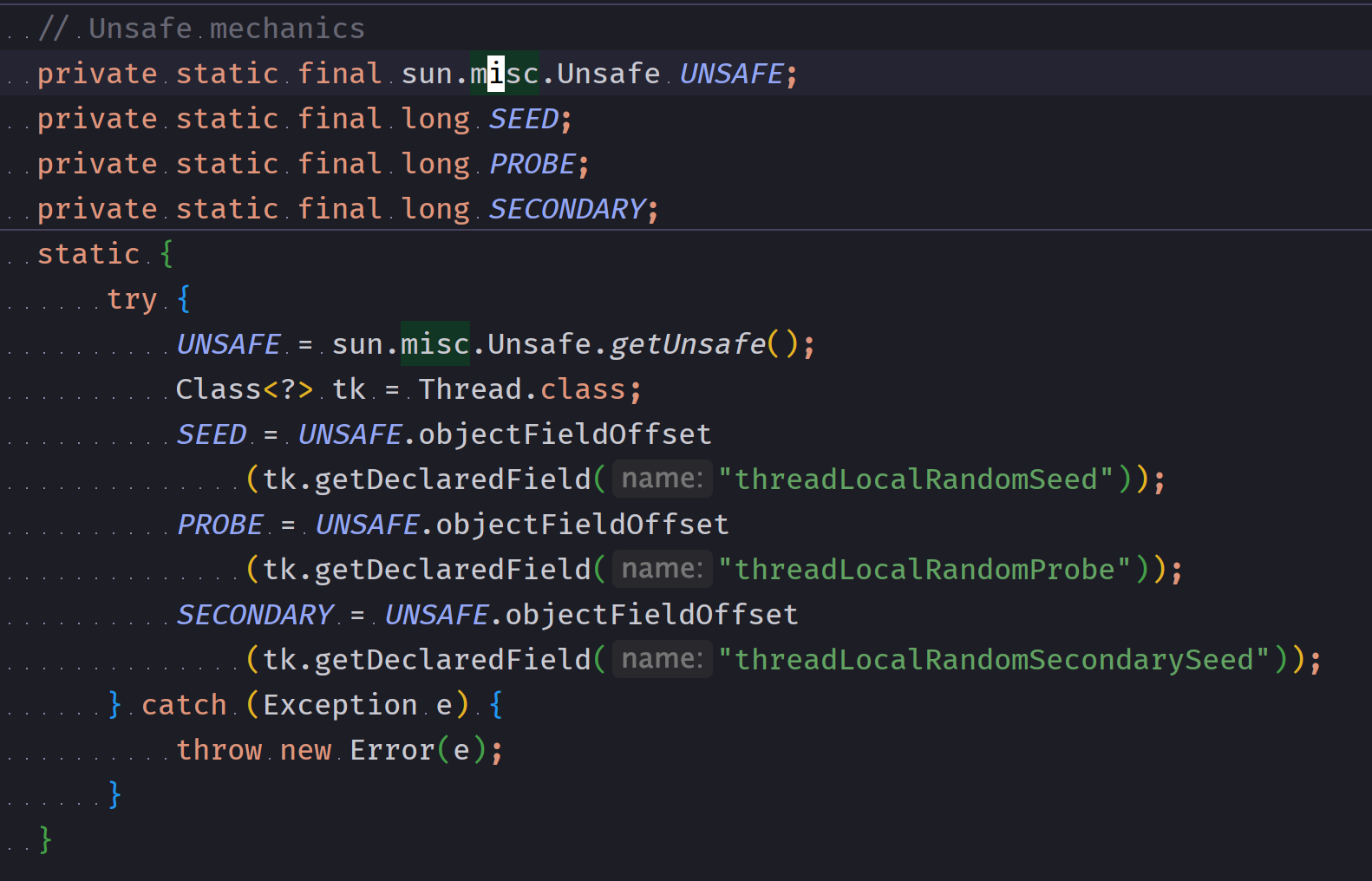


1. ThreadLocalRandom实现原理

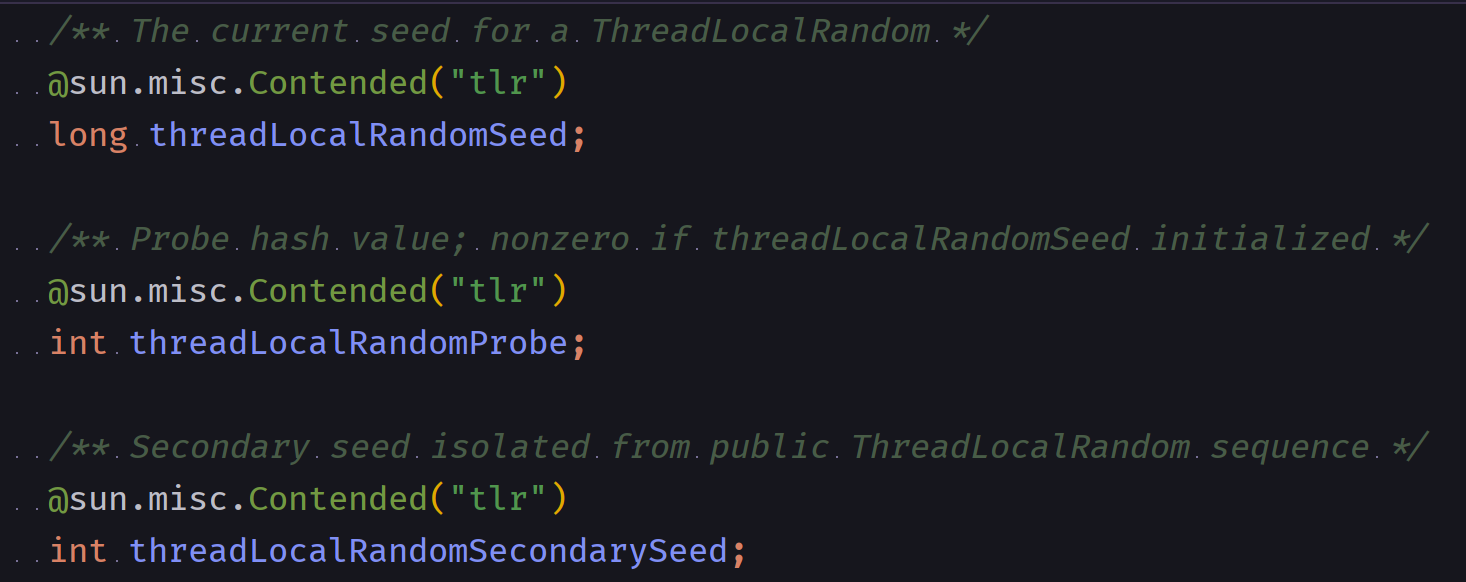


* 1. ThreadLocalRandom类似于ThreadLocal类，当线程调用ThreadLocalRandom的current方法时，ThreadLocalRandom负责初始化调用线程的threadLocalRandomSeed变量，也就是初始化种子

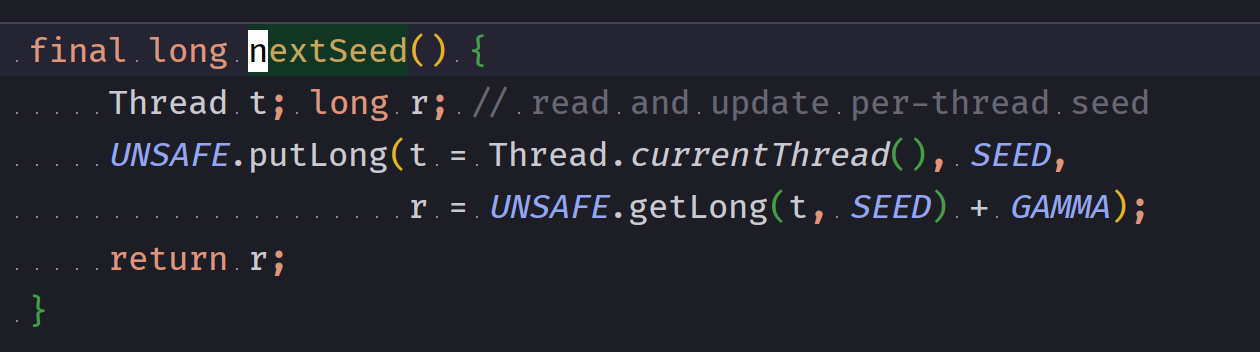
ThreadLocalRandom类



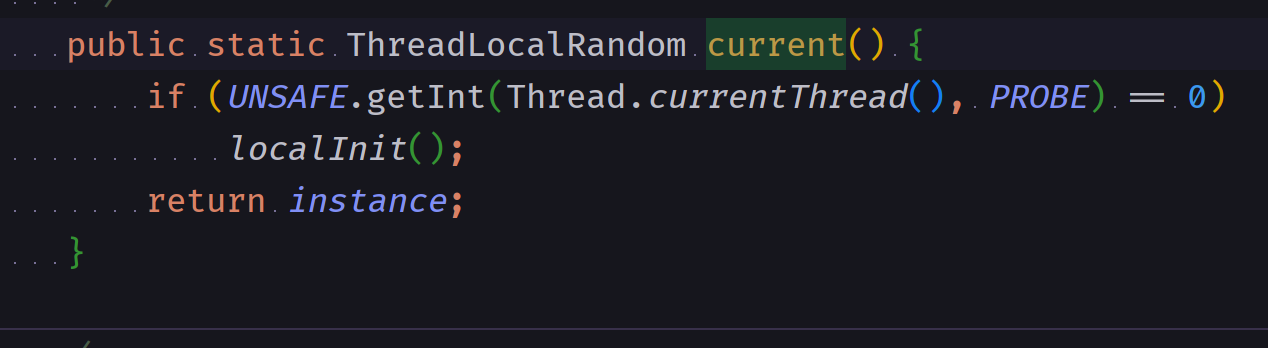
Thread类



* + 1. java.util.concurrent.ThreadLocalRandom#nextSeed，直接更新当前Thread类的threadLocalRandomSeed种子值



* + 1. java.util.concurrent.ThreadLocalRandom#current，延迟初始化



* + 1. java.util.concurrent.ThreadLocalRandom#localInit

