JDK源码学习

2. ArrayList初始化：

类中有一个共享的static的空数组Object[]，在用户创建一个大小为0的ArrayList时，返回给用户。

3. Thread.setDaemon：

设置该线程是用户线程还是守护线程，守护线程与JVM生命周期相同（比如垃圾收集线程，内存管理线程），优先级较低，用于辅助用户线程执行任务；用户线程就是用户定义的，执行特定任务的线程，当所有用户线程退出后，守护线程也都会退出。

4. @CallerSensitive注释：

这个注解是为了堵住漏洞用的。

曾经有黑客通过构造双重反射来提升权限，原理是当时反射只检查固定深度的调用者的类，看它有没有特权，例如固定看两层的调用者（getCallerClass(2)）。如果我的类本来没足够权限群访问某些信息，那我就可以通过双重反射去达到目的：反射相关的类是有很高权限的，而在 我->反射1->反射2 这样的调用链上，反射2检查权限时看到的是反射1的类，这就被欺骗了，导致安全漏洞。使用CallerSensitive后，getCallerClass不再用固定深度去寻找actual caller（“我”），而是把所有跟反射相关的接口方法都标注上CallerSensitive，搜索时凡看到该注解都直接跳过，这样就有效解决了前面举例的问题

5. JDK提供的动态代理

使用反射的机制，不过调用对象一定要实现某个接口，将接口与扩展的方法作为参数，Java会通过反射，生成代理实例，并调用方法。

7. CountDownLatch：多线程计数器

Java.concurrent类库提供的线程计数器，用于等待估计数目的线程执行完毕后，在开始下面的任务。

CountDownLatch latch = new CountDownLatch(2);

latch.countDown();

latch.await();

8. Executor接口

Java有关线程池的类的最上层的接口，使用生产者-消费者的模式，提交Runnable任务，执行Runnable任务

