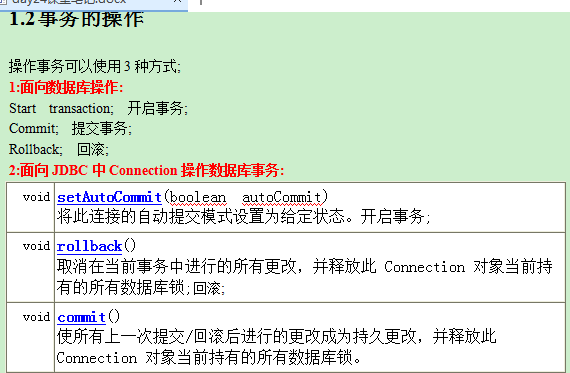
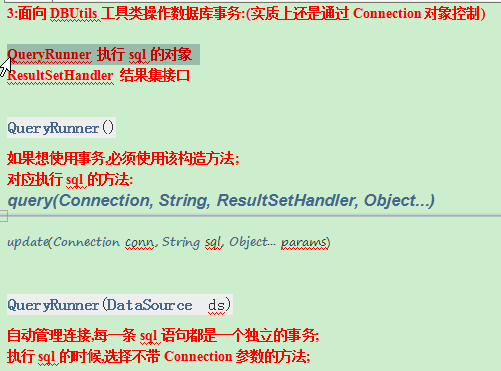
1 笔记见老师的笔记。

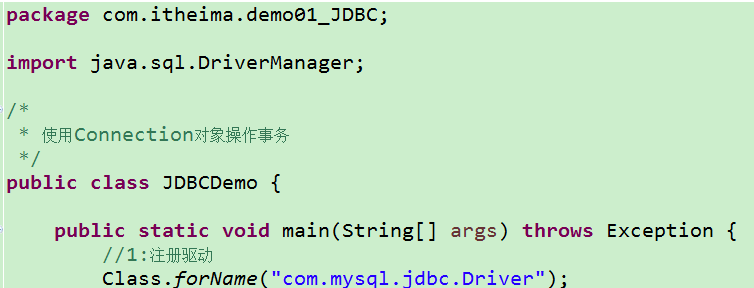


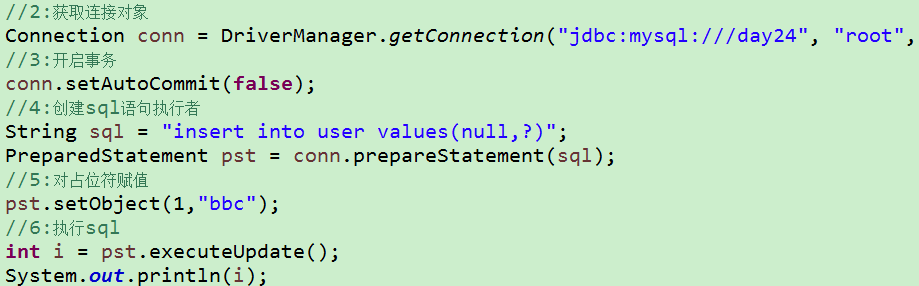


2 desc+表名，use+数据库名。

3 提交事务和回滚都是结束事务。结束之后再插入命令不报错但无效。

4



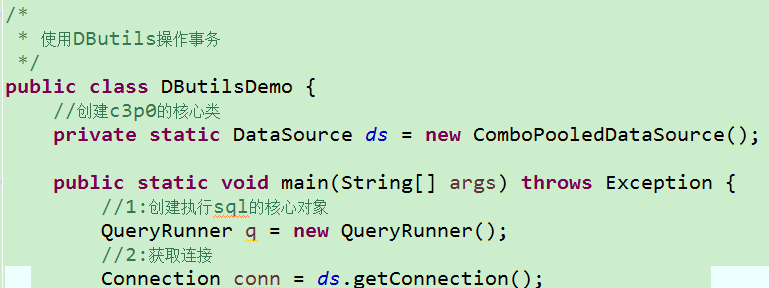


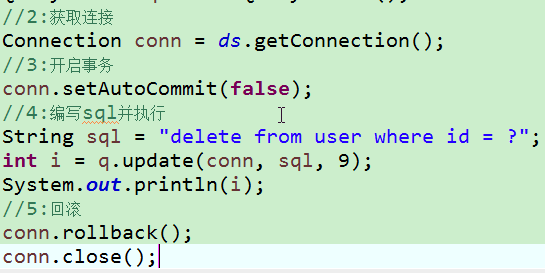


5 resultsethandler?????????????????????????

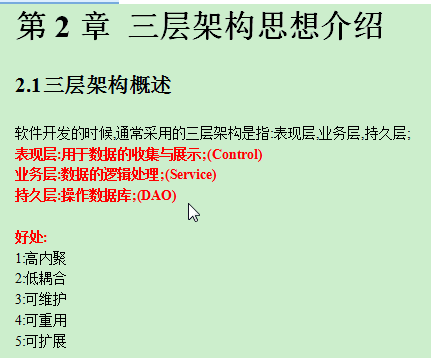
6 dbcp和c3p0都实现dataSource接口。

7



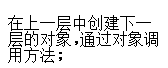


8

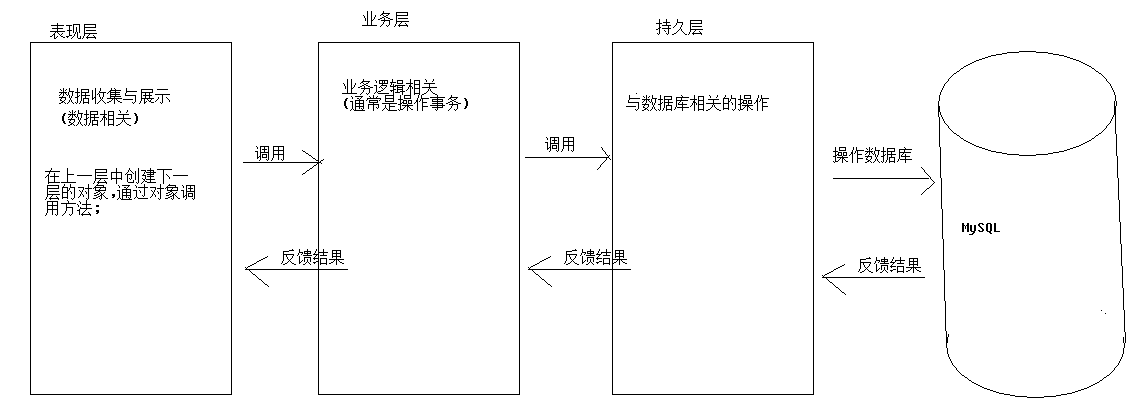


MVC的三层全部都在表现层。

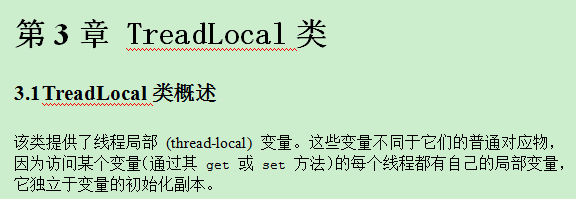
9 三层架构的特点：（建议设计成方法中的局部变量，方便多线程）



10 软件三层架构：



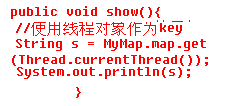
11



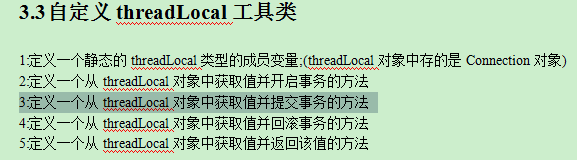
12

模拟ThreadLocal：





13 自定义工具类：（变量是私有的，方法是公有的）

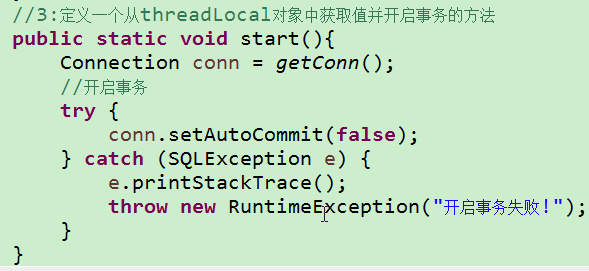


连接池用于获得Collection对象，而ThreadLocal类中存的是Collection对象：（如果有就读取Collection,如果没有就创建新的Collection并且付给ThreadLocal）（**注意每个线程都要用c3p0连接池建立一次连接对象**）（**每次都new一个新的ThreadLocal对象**）

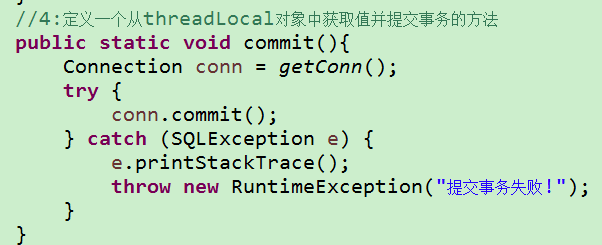


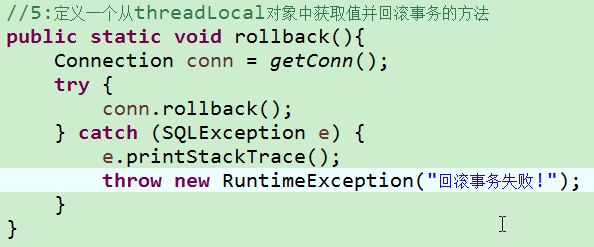


Utils类表示工具，例如ThreadLocalUtils就表示工具类。



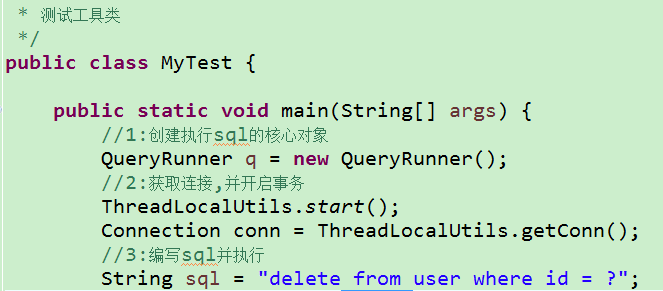
封装的方法唯一的区别就是多了对异常的抛出和处理。

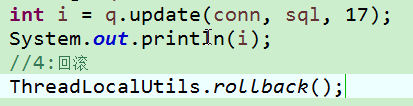




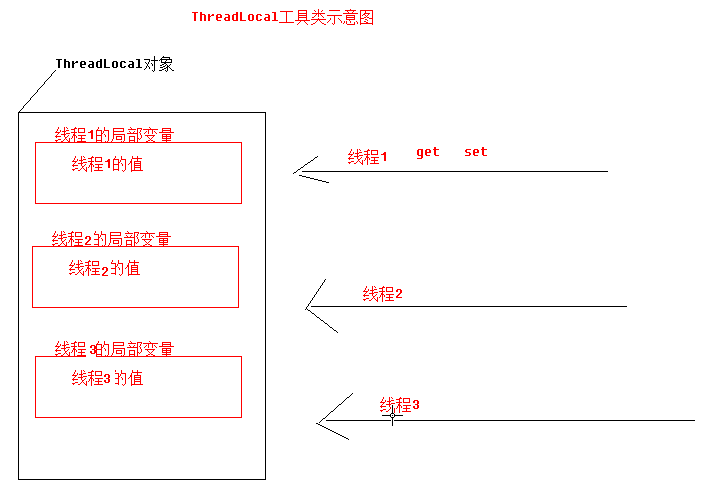
**可见任何操作都要先获取Collection对象，也就是建立连接。**

14 测试工具类：

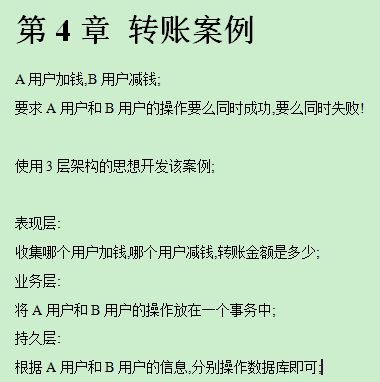




15



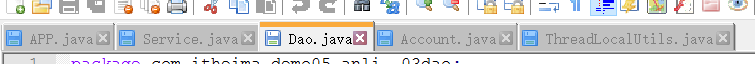
16 转账案例：

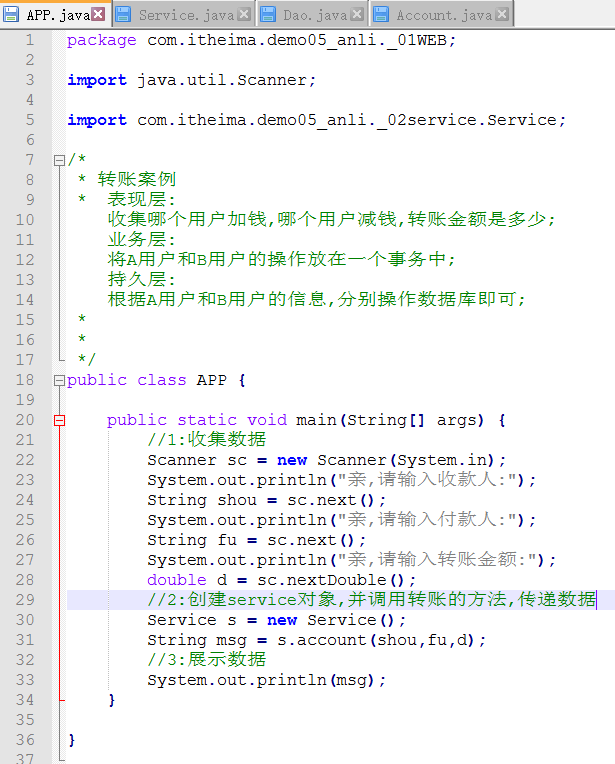


Next()无法写入空格，nextLine()可以写入空格，但是调用之后再用next()和nextInt()会产生异常。

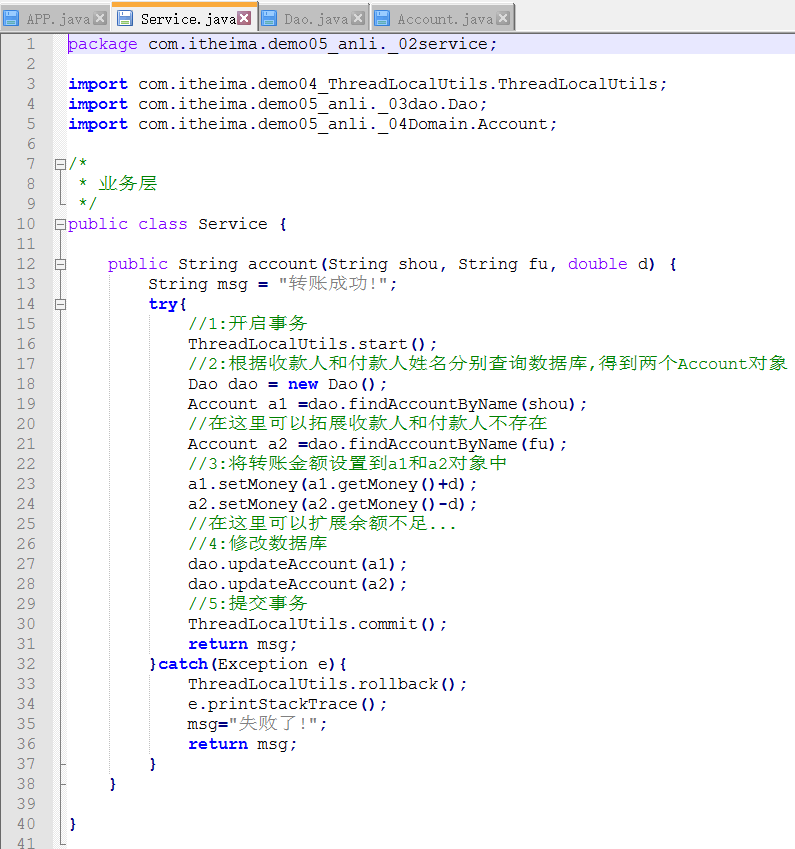
案例看代码01到04.

17 案例：（**第一个是表现层app，第二个是业务层service，第三个是持久层dao**）

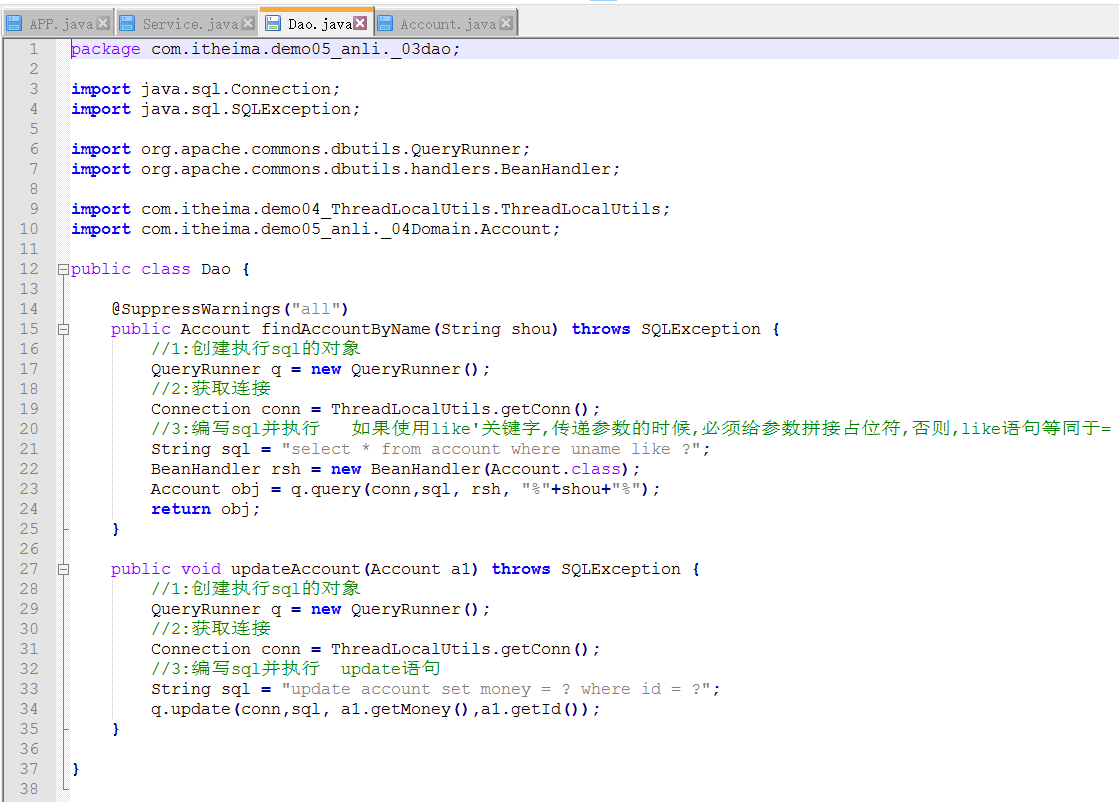




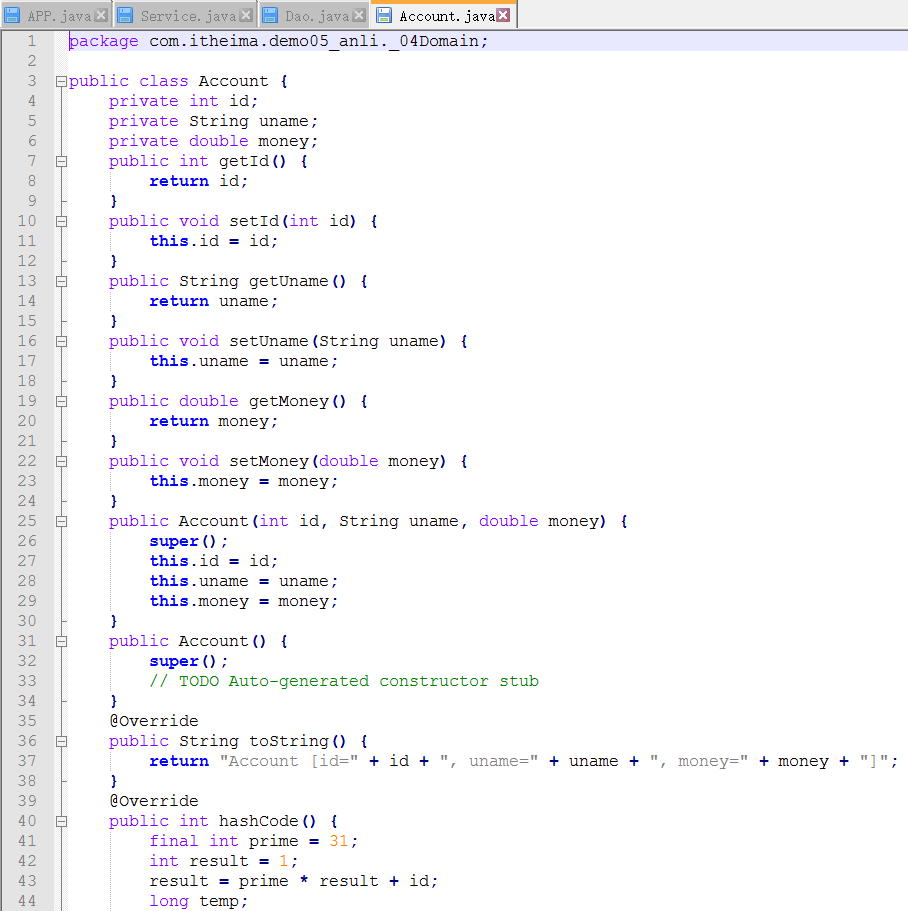
............................................................................................................................................

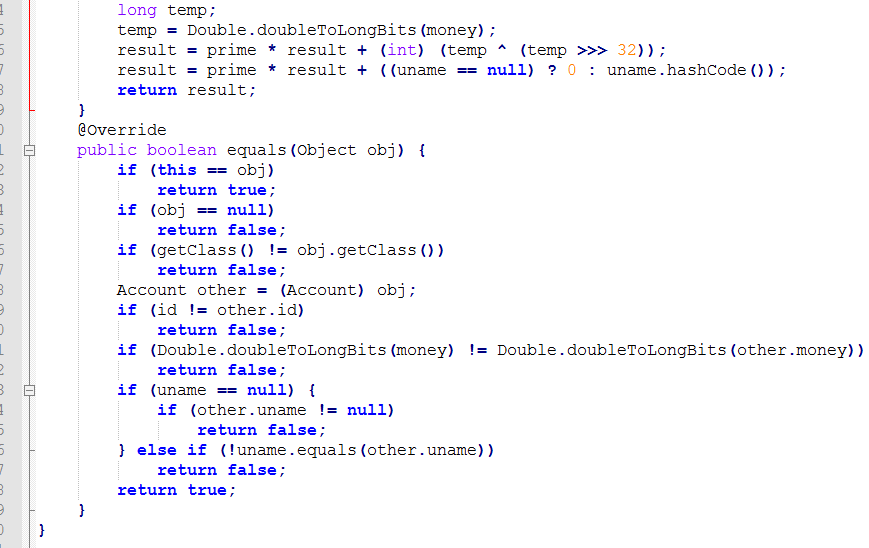


.............................................................................................................................................

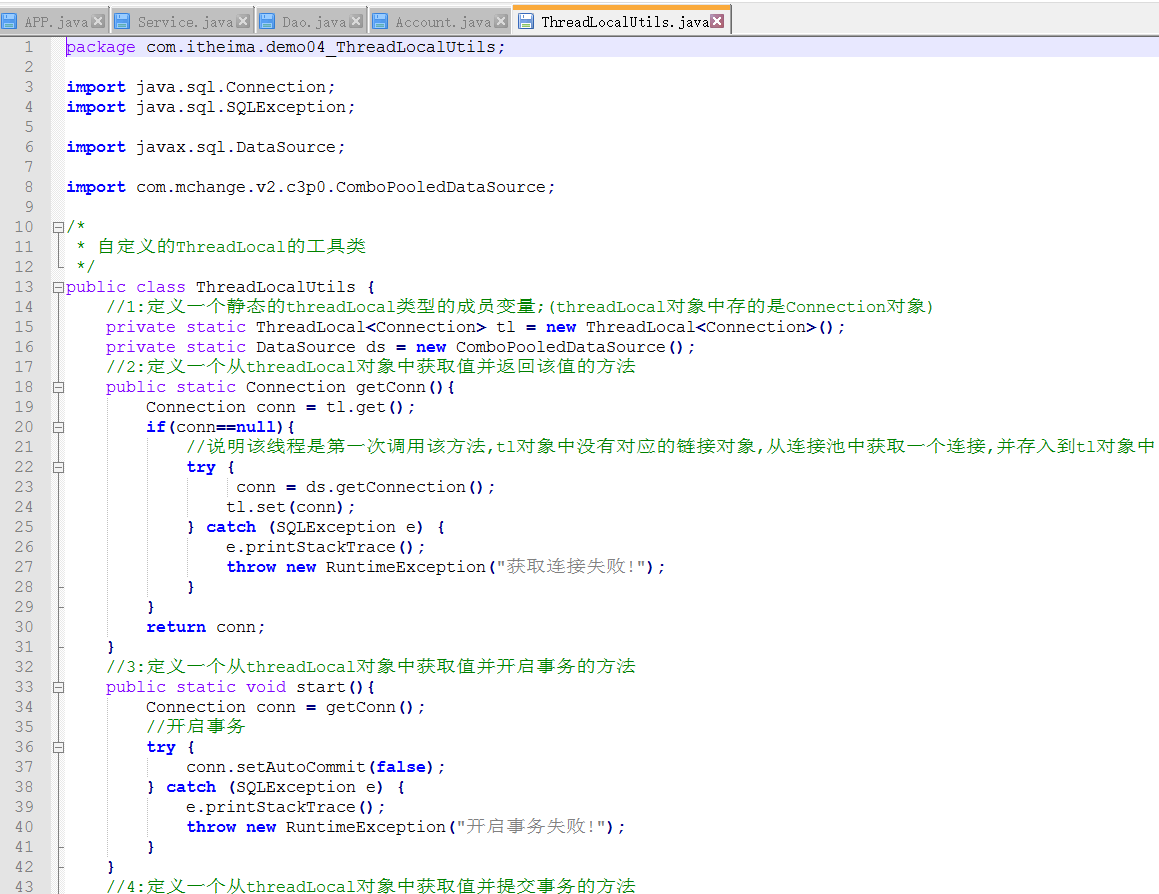


.....................................................................................................................................................



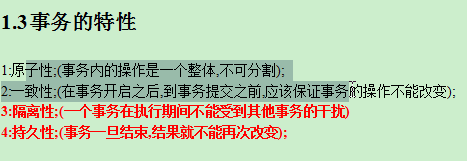


.................................................................................................................................





18 **一致性是一个事务（比如银行转账的减少和增加，不能只减少不增加），隔离性是两个事务。**

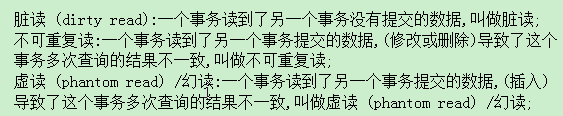


19 **transaction翻译成事务。**



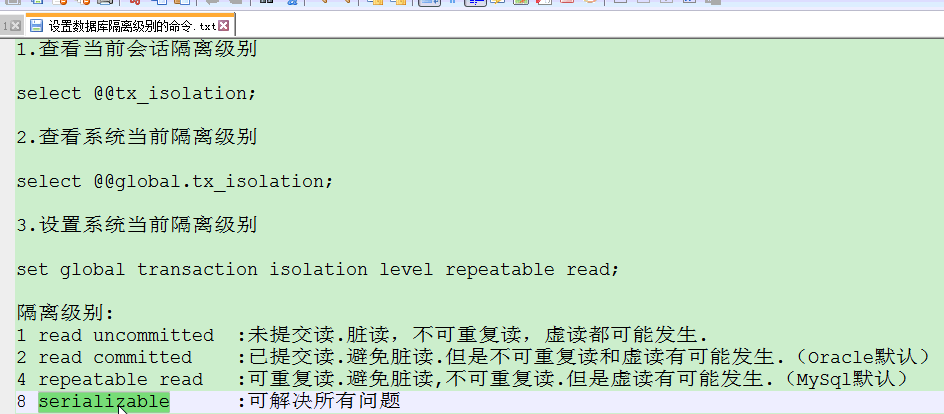
**从上面的文字中可以看出来命名规则是：后面的被限制则前面的必然被限制。**

20  **脏读的意思是读到了不应该读到的数据，不可重复读和虚读的意思是每次读到的内容都不一样，只不过前者是修改和删除（原来的数据），后者是插入（新数据）。**



21 设置完隔离级别要先exit在重新登录数据库才能生效。

22



隔离级别越高越安全，但是效率越低

23