实验一 区块链系统简单实现

【实验介绍】

本实验将参考比特币中的区块结构，使用Java实现一个简单的区块链系统，以更好地理解区块链的概念。本实验所实现的简易区块链系统名为minichain，该系统模拟比特币的挖矿过程，使用一个工作线程进行交易的打包、Merkle树根哈希值的计算以及相应的挖矿过程（随机替换nonce值，计算出满足难度条件的区块哈希值）。在正确补全相应的功能函数后，运行主程序你将会看到新的区块的产生。

作为第一个实验，代码量不大，也比较简单，主要目的是通过代码的编写，从工程的角度理解区块链。代码中根据比特币中的区块结构（包括区块头和区块体）进行类的设计，建议在完成实验的同时，阅读代码中关于区块结构的代码，一是可以参考预先实现的一些功能函数，对于后续的代码工作有一定的参考价值；二是可以更好地理解区块链的实际构成，达到本实验的目的。

【实验要求】

1. 基本区块结构及其必要属性
2. 计算区块的哈希值
3. 基于POW共识的挖矿过程

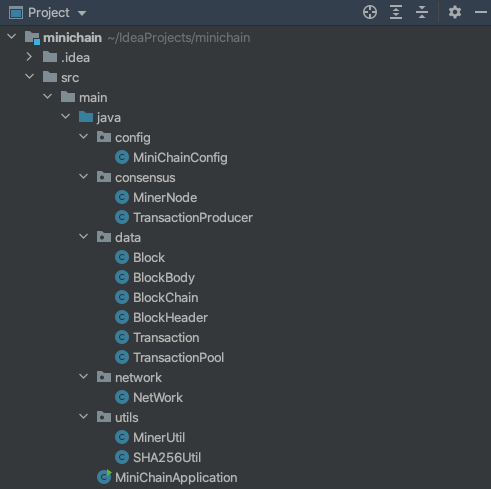
【实验过程】

1. Java环境以及IDE

Java的安装以及环境配置这里就不赘述，有需要的同学可以参考该文章[Windows、Mac以及Ubuntu安装和配置JDK的详细过程](https://jiangnanmax.blog.csdn.net/article/details/113941076)，相应的IDE推荐使用IntelliJ IDEA，其安装就不赘述，自行完成安装。

1. 项目导入及预览

使用IDEA打开项目文件夹，等待加载完成，代码整体结构如下图所示：



这是一个比较简单的小demo，代码量并不大，每个类的详细说明以及待实现的功能函数均在代码中进行了相应的注释，这里做一下整体设计思路的说明。

其中，data包中的类是对交易、区块体、区块头以及区块链的抽象，另外，出于代码实现角度，还额外抽象了一个交易池类，该类主要用于多笔交易的存储，矿工（挖矿线程）就是从该交易池中获取一批次的交易，进行Merkle树根哈希值的计算和区块体的构造等。

相应地，在consensus包中，对矿工进行了类抽象，MinerNode类继承自Thread类，其功能就是从上述交易池中获取一批次的交易，构造出新的区块体、区块头，组合出一个新的区块，然后随机使用nonce值进行区块哈希值的计算，直至哈希值满足难度条件，即可将该区块加入区块链中（即简易模拟了挖矿过程），然后继续生成下一个区块。另外还有一个用于生成随机交易的类，即TransactionProducer类，该类同样继承自Thread类，其功能即随机生成一批次交易放入交易池中，供矿工线程使用。

config包中的MiniChainConfig类中存储了该简易区块链系统的几个关键配置。一个是挖矿的难度设置，即规定了新的区块的哈希值至少以几个0开头才满足难度条件，这里默认设置为4，你可以通过修改难度值来尝试挖矿的速度变化，当然了，这只是一个简单的模拟而已，但是其效果是一致的，该数值设置得越大，挖矿的难度也越大（你可以将难度值设置为5或者6进行尝试，能够明显感受到控制台的输出变慢了）。另一个则是交易池大小的设置，TransactionProducer线程会随机生成交易放入交易池中，直至交易池中的交易数量达到设置的大小，TransactionProducer线程就会进入等待状态，并唤醒MinerNode线程进行交易打包和后续操作；而MinerNode线程则会一次性取出交易池中的所有交易进行交易打包和哈希计算，并在完成其挖矿操作后，进入等待状态，唤醒TransactionProducer线程进行其工作（其实这就是一个最简单的生产者消费者模型，且只有一个生产者线程和消费者线程，如感兴趣，可以尝试增加多个消费线程，也就是多个矿工线程同时进行挖矿）。这里默认将交易池的大小设置为64，不建议对该数值进行修改，设置为2的n次方是为了方便Merkle树根哈希值的计算。

另外，network包中的NetWork类则组合了上述几个功能类，提供两个主要工作线程的启动入口，实现整个系统的运作。utils包则提供了两个工具类（可直接调用），一个是使用SHA256算法进行哈希值计算，另一个则是用于判断区块的哈希值是否满足难度条件。

1. 代码实现

上述内容对该简易区块链系统进行了介绍，相信你已经对该系统有一个比较整体的认识了。接下来就轮到你进行代码实现了，这里已经实现了整体的系统框架，关于类的代码抽象、交易池实现、线程实现以及工具类等均已实现，主要是对MinerNode中的几个功能函数进行了留空，需要你正确补全后系统才能完整的运行起来。

这里需要你进行补全的功能函数分别是MinerNode类中的getBlockBody函数、mine函数和getBlock函数，更加明确的要求以及提示均在代码中进行了注释，详情见源代码。

注意：代码中，针对一个对象（区块、交易等）的哈希值计算，均通过SHA256Util.sha256Digest(Object.toString())完成。代码已经在每一个数据结构的类实现中重写了 toString方法，因此涉及对象的SHA256哈希值计算时，参数请使用相应对象的toString方法获取，这样才能顺利通过单元测试。

1. 单元测试及系统运行效果

这里针对getBlockBody函数提供了一个单元测试，主要是该函数的实现涉及了Merkle

根哈希值的计算，较为重要，因此在补全该函数之后，可以运行test.java.unit中Test类里的测试方法，通过测试即可。

在补全上述三个功能函数后，即可运行整个系统，运行效果如下图所示：

