用Java实现简单的比特币系统

1、本篇背景

近几年区块链的技术很火，连中国的大妈和大爷都站在了科技最前沿的区块链舞台上，如下图。

区块链的感念来源于比特币，比特币自2019年诞生以来，经过9年多的发展从未发生过重大安全事故，可见其技术和用户基础都十分强大。可以说发展得很疯狂，9年多时间比特币价格上涨

1500万倍！

很多人都知道比特币，但是涉及到比特币系统方面的东西就很少人真正了解。因此本篇内容对从技术角度来解释比特币，用Java实现简单的比特币系统，进一步认识其底层的区块链原理。

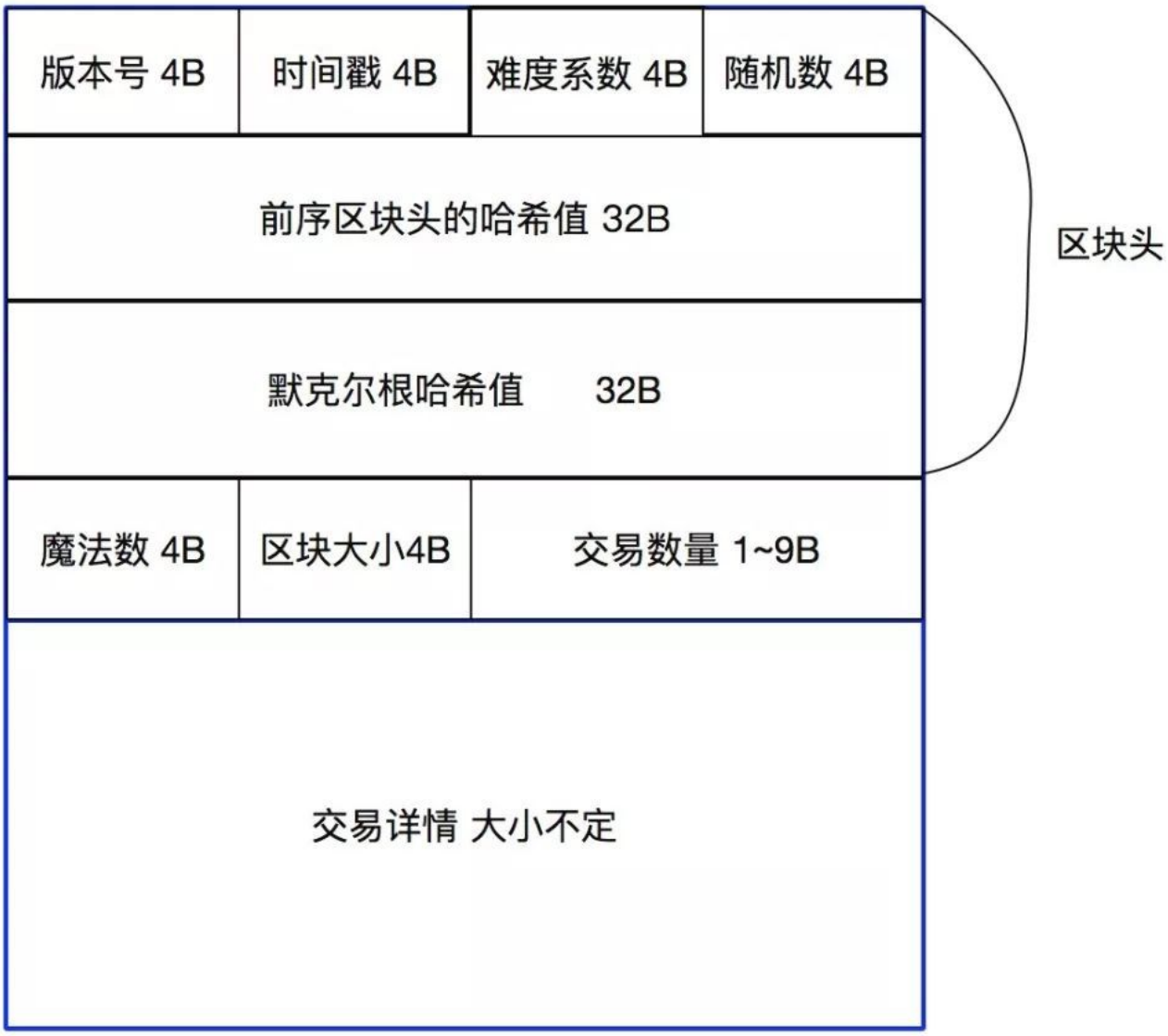
2、区块链结构

比特币是构建在区块链技术之上的一种加密数字货币，区块链顾名思义即由很多区块组成的链条，类似于数据结构中的单链表。

可以把区块链看作成比特币网络中的一个大账本，而每个区块相当于账本中的一页，账本的每一页里都记录了区块头（时间戳、难度系数、随机数等）、交易详情、交易计数器和区块大小等数据。

区块头是每个区块中前80个字节，主要包括上一区块头哈希值，用于保证区块按顺序串联。时间戳，记录该区块的生成时间。难度系数，即该数学题的难度系数，满足SHA256函数输出字符串前几位为零。随机数(Nonce)：全网矿工记录解密该区块相关数学题答案的值，也就是重复了多少次计算才得到满足难度系数的哈希值。
交易详情，记载了每笔交易的转入转出方金额及转出方的数字签名，是每个区块的主要内容。交易计数器，表达每个区块中交易的数量。区块大小，表示每个区块数据的大小，当前每个区块限定在1MB以内，不排除以后有扩大的可能。

比特币的区块结构如下图：



Block Structure of Bitcoin

于是，我们可以用一个JavaBean先来构造一个区块的结构：

```
package com.wenzil.blockchain.bean;

import java.util.List;

/**
 * 区块结构
 */
public class Block {
    // 区块索引号
    private int index;

    // 当前区块的hash值，区块唯一标识
    private String hash;
```

```
    // 生成区块的时间戳
    private long timestamp;

    // 当前区块的交易集合
    private List<Transaction> transactions;

    // 随机数Nonce，计算正确hash值的次数
    private int nonce;

    // 前一个区块的hash值
    private String previousHash;

    // 这里省略了对应的set和get方法

    // 构造方法
    public Block(int index, long timestamp, List<Transaction> transactions,
int nonce, String previousHash, String hash) {
        super();
        this.index = index;
        this.timestamp = timestamp;
        this.transactions = transactions;
        this.nonce = nonce;
        this.previousHash = previousHash;
        this.hash = hash;
    }
}
```

3、交易模型

转账交易即比特币的拥有方之间进行的相互转账行为，我们把这些比特币的拥有方暂时假设为比特币的钱包，钱包有对应的钱包地址，那这些转账交易实际上就是钱包地址之间的转账交易(类似于支付宝用户之间的转账，其实就是支付宝用户名之间的转账)，这些转账交易需要被记录到账本里才算真正的生效。

由于比特币的转账交易设计比较复杂，这里暂时不深入讨论，所以这里用JavaBean设计了一个简单的交易模型如下：

```
package com.wenzil.blockchain.bean;

/**
 * 交易
 */
public class Transaction {
    // 交易唯一标识
```

```
    private String id;

    // 交易发送方钱包地址
    private String sender;

    // 交易接收方钱包地址
    private String recipient;

    // 交易金额
    private int amount;

    // 这里省略了对应的set和get方法

    // 构造方法
    public Transaction(String id, String sender, String recipient, int amoun
t) {
        super();
        this.id = id;
        this.sender = sender;
        this.recipient = recipient;
        this.amount = amount;
    }
}
```

4、模拟挖矿

挖矿到底是怎么回事？为什么那么多人吵着要去挖矿，梦想着一夜暴富？

我们可以简单的把**挖矿比喻成矿工解一道数学难题的过程，只要解对了就能获取比特币系统奖励的一笔比特币**，同时获取了区块链账本新区块的交易记账权，矿工会把比特币系统近期发生的转账交易记录到账本新的一页上，并获取交易的手续费，一旦交易被记录进了账本，交易就算完成了，接收方才能真正收到发送方转账的比特币。那这道数学难题到底长什么样了？

我们看下这个公式：
Hash = SHA256（区块链的最后一个区块的Hash + 需记账交易记录信息 + 随机数）

这个公式已经很明白了，SHA-256是一种哈希加密算法，被加密的前两部分是固定不变的，我们只有依赖于**随机数的不断变化计算出不同的hash**结果，系统要求**hash结果必须要以10个0开头**，这个几率实在是太小太小，我们做测试可以简单一点，比如只要**hash结果满足以4个0开头**，我们就认为解题成功，即挖矿成功了，这时矿工就可以生成一个新的区块把需记账的交易记录全部记录进区块里去，同时再构造一笔系统奖励给自己的比特币的交易(发起方为系统，接收方为矿工，比特币奖励金额假设为50个)，将其也记录进账本，这样通过账本里的交易记录就会发现矿工的余额多了50个比特币了，我们通过如下代码来模拟挖矿：

```
/**
  * 模拟PoW挖矿
  * @param blockchain 整个区块链
  * @param txs 需记账交易记录
  * @param address 矿工钱包地址
  * @return
  */
public static void mineBlock(List<Block> blockchain, List<Transaction> txs,
String address) {
    // 比特币第一个创世区块的奖励设定为50个比特币，此后每新建210,000个区块，奖励减半。
    // 奖励金在2012年当时候从50比特币减半为25比特币，2016年从25个比特币减半为12.5个比特
币。大概2020年就会再减半为6.25比特币。
    // 加入系统挖矿奖励的交易，这里默认挖矿奖励50个比特币。
    Transaction sysTx = new Transaction(CryptoUtil.UUID(), "", address, 50);
    txs.add(sysTx);
     // 获取当前区块链里的最后一个区块
    Block latestBlock = blockchain.get(blockchain.size() - 1);
    // 随机数
    int nonce = 1;
    String hash = "";
    // 开始挖矿
     while(true){
        hash = CryptoUtil.SHA256(latestBlock.getHash() + JSON.toJSONString(t
xs) + nonce);
        if (hash.startsWith("0000")) {
            System.out.println("=====计算结果正确，计算次数为：" +nonce+ "，hash
：" + hash);
                break;
         }
        nonce++;
        System.out.println("计算错误，hash:" + hash);
    }

    // 解出难题，可以构造新区块并加入进区块链里
    Block newBlock = new Block(latestBlock.getIndex() + 1, System.currentTim
eMillis(), txs, nonce, latestBlock.getHash(), hash);
    blockchain.add(newBlock);
    System.out.println("挖矿后的区块链：" + JSON.toJSONString(blockchain));
}
```

对应的SHA256核心代码如下：

```
/***
  * SHA256加密处理
  * @param str 原始字符串
  * @return 返回加密后的哈希值
```

```
     */
    public static String SHA256(String str) {
        // MessageDigest类为应用程序提供的信息摘要算法
        MessageDigest messageDigest;
        String encodeStr = "";
        try {
             // 使用SHA-256算法
             messageDigest = MessageDigest.getInstance("SHA-256");
            // 使用UTF-8编码更新摘要
            messageDigest.update(str.getBytes("UTF-8"));
            // 将bytes数组转换为十六进制值的字符串
             encodeStr = byte2Hex(messageDigest.digest());
        } catch (Exception e) {
             System.out.println("getSHA256 is error" + e.getMessage());
        }
        return encodeStr;
    }
```

5、钱包余额

计算某个钱包地址的余额，其实就是从区块链账本里找出所有该地址作为接收方的交易记录，将这些交易记录的发生金额累加就得到该地址收到的所有比特币金额了，然后找出所有该地址作为发送方的交易记录再次累加则得到改地址发送出去的所有比特币金额了，用收到的比特币金额之和减去发送出去的比特币金额之和就得到该地址的真正的比特币余额了，具体我们看下代码：

```
    /**
     * 查询钱包的余额
     * @param blockchain 整个区块链
     * @param address 钱包地址
     * @return
     */
    public static int getWalletBalance(List<Block> blockchain, String address) {
        int balance = 0;
        for (Block block : blockchain) {
             List<Transaction> transactions = block.getTransactions();
            for (Transaction transaction : transactions) {
                 if (address.equals(transaction.getRecipient())) {
                    balance += transaction.getAmount();
                }
                if (address.equals(transaction.getSender())) {
                     balance -= transaction.getAmount();
                 }
            }
         }
         return balance;
```

```
    }
}
```

6、系统入口

最后，我们看下这个小比特币系统的运行效果，main函数代码如下：

```
public class Main {
    public static void main(String[] args) {
        // 创建一个空的区块链
        List<Block> blockchain = new ArrayList<>();
        // 生成创世区块
        Block block = new Block(1, System.currentTimeMillis(), new ArrayList
<Transaction>(), 1, "1", "1");
        // 加入创世区块到区块链里
        blockchain.add(block);
        System.out.println(JSON.toJSONString(blockchain));

        // 发送方钱包地址
        String sender = "1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa";
        // 接收方钱包地址
        String recipient = "1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h";

        // 创建一个空的交易集合
        List<Transaction> txs = new ArrayList<>();
        // 挖矿
        BlockService.mineBlock(blockchain, txs, sender);

        System.out.println(sender + "钱包的余额为：" + BlockService.getWalletB
alance(blockchain, sender));

        // 创建一个空的交易集合
        List<Transaction> txs1 = new ArrayList<>();
        // 已发生但未记账的交易记录，发送者给接收者转账3个比特币
        Transaction tx1 = new Transaction(CryptoUtil.UUID(), sender, recipie
nt, 3);
        // 已发生但未记账的交易记录，发送者给接收者转账1个比特币
        Transaction tx2 = new Transaction(CryptoUtil.UUID(), sender, recipie
nt, 1);
        txs1.add(tx1);
        txs1.add(tx2);
        // 挖矿
        BlockService.mineBlock(blockchain, txs1, sender);
        System.out.println("钱包'" + sender + "'的余额为：" + BlockService.get
WalletBalance(blockchain, sender));
        System.out.println("钱包'" + recipient + "'的余额为：" + BlockService.
```

```
getWalletBalance(blockchain, recipient));
    }
}
```

7、运行系统

运行结果如下：

```
0bf31c625538676a35be2d6e8c5d569e8e5
计算错误，hash:b2250e0f398c92ec15be17a245edfda25df0002e095ce456dd17b533a8870724
计算错误，hash:325f0a6bfa74f179ff9e9cb7b477c7f607e0502033f079c6839c1ce0b383a55f
计算错误，hash:81296178df9df346d413adcf7b71914d84d6a1a031583cbed2ce8778ab205399
#####省略上N行输出#####
计算错误，hash:d86d5cbb1af248cc03cf998cb88b0243fbcb2ca78d0740de6b713d1066015f5d
计算错误，hash:d7016aa7b22402c5bf8b3130e95b1de434b313ec198107dc65efd70cf21901ce
计算错误，hash:25fd827a293a48a41e4c2c6d3b13ae694c1498510b16bbe623cde8f0b4aad998
=====计算结果正确，计算次数为：43387，hash：0000e513c1b3c61bacaa363ff37324867e2b117bb52c1f37cdd42758211e4055
挖矿后的区块链：[{"hash":"1","index":1,"nonce":1,"previousHash":"1","timestamp":1528433910711,"transactions":[]},{"hash":"0000c4f43738c76edb0e84ab250b3348e5d5bcb7a7506c6482ef1f785c547eed","index":2,"nonce":121049,"previousHash":"1","timestamp":1528433912376,"transactions":[{"amount":50,"id":"e2a92f509ba04a6aa8f17cae7166156b","recipient":"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa","sender":""}]},{"hash":"0000e513c1b3c61bacaa363ff37324867e2b117bb52c1f37cdd42758211e4055","index":3,"nonce":43387,"previousHash":"0000c4f43738c76edb0e84ab250b3348e5d5bcb7a7506c6482ef1f785c547eed","timestamp":1528433912890,"transactions":[{"amount":3,"id":"58553d20bfcc47d1b1295bc2a552af79","recipient":"1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h","sender":"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa"},{"amount":1,"id":"e2c9735880244622af447c47d52522ac","recipient":"1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h","sender":"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa"},{"amount":50,"id":"da049bd316dd4ad4a8f407b64f74acf5","recipient":"1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa","sender":""}]}]
钱包'1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa'的余额为：96
钱包'1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h'的余额为：4

Process finished with exit code 0
```

对运行结果中"挖矿后的区块链"格式化之后，内容如下：

视图 JSON数据

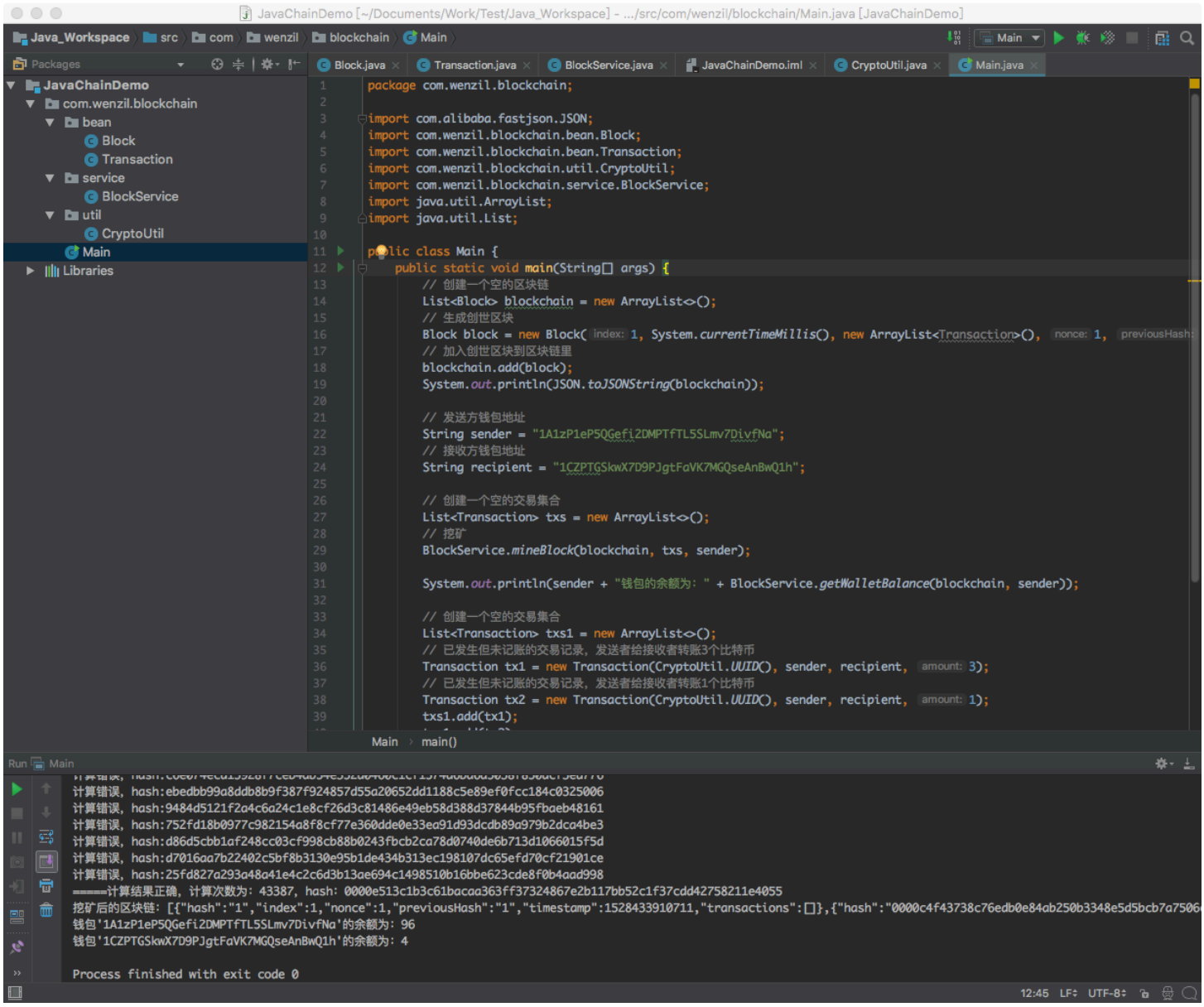
粘贴 复制 格式化 删除空格 删除空格并转义 去除转义


```
[
  {
    "hash": "1",
    "index": 1,
    "nonce": 1,
    "previousHash": "1",
    "timestamp": 1528433910711,
    "transactions": [

    ]
  },
  {
    "hash": "0000c4f43738c76edb0e84ab250b3348e5d5bcb7a7506c6482ef1f785c547eed",
    "index": 2,
    "nonce": 121049,
    "previousHash": "1",
    "timestamp": 1528433912376,
    "transactions": [
      {
        "amount": 50,
        "id": "e2a92f509ba04a6aa8f17cae7166156b",
        "recipient": "1AlzP1eP5QGefi2DMPTfTL5SLmv7DivfNa",
        "sender": ""
      }
    ]
  },
  {
    "hash": "0000e513c1b3c61bacaa363ff37324867e2b117bb52c1f37cdd42758211e4055",
    "index": 3,
    "nonce": 43387,
    "previousHash": "0000c4f43738c76edb0e84ab250b3348e5d5bcb7a7506c6482ef1f785c547eed",
    "timestamp": 1528433912890,
    "transactions": [
      {
        "amount": 3,
        "id": "58553d20bfcc47d1b1295bc2a552af79",
        "recipient": "1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h",
        "sender": "1AlzP1eP5QGefi2DMPTfTL5SLmv7DivfNa"
      },
      {
        "amount": 1,
        "id": "e2c9735880244622af447c47d52522ac",
        "recipient": "1CZPTGSkwX7D9PJgtFaVK7MGQseAnBwQ1h",
        "sender": "1AlzP1eP5QGefi2DMPTfTL5SLmv7DivfNa"
      },
      {
        "amount": 50,
        "id": "da049bd316dd4ad4a8f407b64f74acf5",
        "recipient": "1AlzP1eP5QGefi2DMPTfTL5SLmv7DivfNa",
        "sender": ""
      }
    ]
  }
]
```


8、项目总结

至此，我们就用Java语言基于区块链账本技术实现了一个简单的比特币系统，包含区块链功能，挖矿产生新比特币功能，转账交易功能，查询余额功能，整个项目工程及运行结果如下图：



当然，真正的比特币系统远不止这么简单，比如运用密码学生成比特币钱包与保证交易信息不被篡改，运用P2P通讯技术实现点对点分布式网络等功能，这里只是抛砖引玉。

本篇内容大部分来自"轻风微课"的博客，如有侵权联系我删除文字：
https://my.oschina.net/u/3796575/blog/1791185