区块链程序开发怎么做（用Java代码创建第一个区块链程序） 我们都看到了比特币和其他加密货币的规模。虽然这种在线货币的波动性是出了名的，但其背后的技术有可能从内到外扰乱每一个行业。因为区块链有着无限的应用范围，它每天都以新的方式出现。

在这篇文章中，我们将探讨区块链背后的体系结构以及分布式账本的工作原理。一旦你亲眼看到，你就会明白为什么那么多开发者将区块链视为一种新常态。我们还将深入学习如何创建自己的（基本）区块链序列和使用工作证明（挖掘）系统的简短教程。

理解区块链

首先，在我们自己尝试之前，我们都需要在区块链是什么以及它是如何工作的问题上保持一致。块有头信息，还有一组或一个block“块”数据。这些数据通常是加密货币世界中的一项交易，但可以进行调整。然后，链从Genesis块开始，并根据一个块中存储的事务或数据集的数量创建新的块。

一旦达到该阈值，将创建一个新块。这个新的链接到上一个，这就是区块链这个术语的来源。区块链也是不可变的。这是因为每个事务或数据集都涉及SHA-256哈希。块中的内容也被散列。这意味着每个块都有一个唯一的标识符，链接块的散列也存储在图中并散列。

因为区块链是不可变的，所以它们非常安全。基本上不可能搞乱一个。试图伪造交易数据或财产将是一项挑战。更重要的是，随着链条的增长，它变得更加安全。破坏这个系统的技术还不存在，这是个好消息。

区块链有三种类型：

公共——公共区块链是开放给任何人看的。交易和数据都出现在分类账上，这意味着每个人都可以参与共识过程。

联邦——与公共区块链相反，联邦不允许每个人都参与协商一致过程。相反，具有访问分类账权限的节点数量有限。

私有——最后，私有区块链主要用于公司内部。这些仅限于能够访问区块链和执行交易的特定成员。

交易

接下来，让我们谈谈区块链内的交易。区块链技术是分布式的。因为它们只是附加的，所以很容易在网络中的节点之间复制区块链。虽然节点通常进行点对点通信（如比特币），但它们也可以通过HTTP通过API进行分散。

交易可以是任何东西。它可以有一个执行代码或者只是存储信息。随着新智能合约的推出，您可以看到这项技术正在发挥作用。本质上，这些智能合约是计算机协议，用于促进和验证数字合约。他们很可能会在制造业、银行业等行业成为主流。

让我们以比特币为例。对于比特币，有一笔一定金额的交易从所有者帐户转移到另一个帐户。此事务具有公钥和帐户ID以确保其安全。这些事务被添加到网络中，并汇集在一起。虽然他们在一个共享的网络中，但他们不在一个区块或链本身中。

这是怎么回事？归根结底是共识机制。你可能已经知道一种被称为挖掘的机制，它被比特币使用。有一个无休止的共识机制清单，要把它们全部列出需要很长时间。您需要知道的是，它们是收集事务、构建块并将这些块添加到链中以进行验证的算法或模式。

开始

如上所述，区块链是一个区块链或区块列表。每个块都有自己的数字签名以及前面块的数字签名。有些还可能包含事务信息之类的数据。数字签名称为散列。每个块自己的哈希是根据前一个块计算的。一个更改将影响此后的所有哈希。通过计算和比较，我们可以看到区块链是否有效。

因为数据的任何变化都会导致一个断链，让我们首先创建一个类块，它将为Buffic链形成基础。

publicclassBlock{publicStringhash;PublicStringpreviousHash;PrivateStringdata;PrivatelongtimeStamp;//BlockConstructor.PublicBlock(Stringdata,StringpreviousHash){this.data=data;this.previousHash=previousHash;this.timeStamp=newDate().getTime();}上面的代码以字符串哈希开头，这就是我们制作数字签名的地方。如您所见，previousHash将保存最后一个块的哈希，字符串数据将保存整个块数据。因为我们有基础，所以现在我们可以生成一个数字签名。这意味着您必须选择加密算法。对于本例，我们将使用SHA256。为了得到这个结果，我们将导入java.security.MessageDigest。

importjava.security.MessageDigest;publicclassStringUtil{publicstaticStringapplySha256(Stringinput){try{MessageDigestdigest=MessageDigest.getInstance(“SHA-256”);byte[]hash=digest.digest(input.getBytes(“UTF-8”));StringBufferhexString=newStringBuffer();//Thiswillcontainhashashexidecimalfor(in

ti=0;i<hash.length;i++){Stringhex=Integer.toHexString(0xff&hash[i]);if(hex.length()==1)hexString.append(‘0’);hexString.append(hex);}returnhexString.toString();}catch(Exceptione){thrownewRuntimeException(e);}}}这将创建一个StringUtil实用程序类，稍后我们将使用该类。这将获取一个字符串并应用SHA256算法返回生成的签名。有了新的助手，我们可以计算块类中的哈希值。为了计算散列，我们需要使用块的所有部分，我们不想被弄乱。

publicStringcalculateHash(){Stringcalculatedhash=StringUtil.applySha256(previousHash+Long.toString(timeStamp)+data);returncalculatedhash;}publicBlock(Stringdata,StringpreviousHash){this.data=data;this.previousHash=previousHash;this.timeStamp=newDate().getTime();this.hash=calculateHash();/}完美的现在是进行测试的时候了。我们需要创建一些块并在屏幕上显示哈希值。这样，我们就知道一切都在正常工作。因为genesis（first）块中没有上一个块，所以我们将输入“0”作为上一个散列的值。

publicclassTest{publicstaticvoidmain(String[]args){BlockgenesisBlock=newBlock(“Genesisblock”,“0”);System.out.println(“Hashforblock1:”+genesisBlock.hash);BlocksecondBlock=newBlock(“Secondblock”,genesisBlock.hash);System.out.println(“Hashforblock2:”+secondBlock.hash);BlockthirdBlock=newBlock(“Thirdblock”,secondBlock.hash);System.out.println(“Hashforblock3:”+thirdBlock.hash);}}您的输出应该有三个块，每个块都有自己的数字签名。您的数字签名将具有不同的值，具体取决于您的唯一时间戳，但您正在取得进展。现在，是将块存储在ArrayList中的时候了。同时，我们将输入gson并将其视为Json。

importjava.util.ArrayList;importcom.google.gson.GsonBuilder;publicclassTest{publicstaticArrayList<Block>blockchain=newArrayList<Block>();publicstaticvoidmain(String[]args){blockchain.add(newBlock(“Genesisblock”,“0”));blockchain.add(newBlock(“Secondblock”,blockchain.get(blockchain.size()-1).hash));blockchain.add(newBlock(“Thirdblock”,blockchain.get(blockchain.size()-1).hash));StringblockchainJson=newGsonBuilder().setPrettyPrinting().create().toJson(blockchain);System.out.println(blockchainJson);}}现在我们正在做更多的过程。您的输出更接近区块链的预期外观。最后，我们将检查我们的区块链的有效性。我们将在测试类中创建一个ischainvalid（）布尔方法。这将遍历到目前为止我们创建的所有块，并将比较散列，看看它是否等于计算的散列，依此类推。

publicstaticBooleanisChainValid(){BlockcurrentBlock;BlockpreviousBlock;for(inti=1;i<blockchain.size();i++){currentBlock=blockchain.get(i);previousBlock

=blockchain.get(i-1);if(!currentBlock.hash.equals(currentBlock.calculateHash())){System.out.println(“CurrentHashesnotequal”);returnfalse;}if(!previousBlock.hash.equals(currentBlock.previousHash)){System.out.println(“PreviousHashesnotequal”);returnfalse;}}returntrue;}使用这段代码，如果对块有任何更改，那么您将得到一个错误的返回。如果你得到了一个真实的回答，你就成功地做到了。你用一个独特的数字签名将所有存储的数据块连接在一起。你已经准备好开始挖掘了。

最后的想法

因为区块链是去中心化的，所以没有一个权威机构为被接受的交易设置规则。区块链涉及一定程度的信任，因为这些交易存储在开放网络上。尽管这项技术有很多困惑，正如你在上面看到的，它并不像许多人想象的那么复杂。

随着越来越多的开发人员试图解决他们自己的采矿问题，我们看到了更多的工具来帮助他们。Loggly和Microsoft Azure等软件旨在使区块链更容易访问。开发者探索区块链世界有很多原因。首先，正如你在上面所看到的，它相对简单易学，而且在不久的将来，它也将成为一个受欢迎的职业。

虽然区块链不是云中的神奇数据库，但它是复杂技术交易的现代解决方案。毫无疑问，这项技术将继续存在。只有时间才能告诉我们，它在未来将如何继续应用。