Devoir 2 - BlockChain

Maxime Côté-Gagné (8851539)

Travail présenté à Monsieur Robert Laganière

Dans le cadre du cours CSI 2510

Université d'Ottawa

22 octobre 2018

Devoir 2 BlockChain

Maxime Côté-Gagné
88515139

minerID: mcote074

Classe:

Block:

Block est une classe qui sont des objets de type Block qui on un index(int), un temps(Timestamp), un object Transaction, un nonce(random String) et le previousHash qui est le hash du dernier Block.

Chaque Block est accompagné d'un hash qui lui est donnée à partir du toString() du Block qui est généré dans la classe BlockChain.

Transaction:

Transaction est une classe qui est pour des objets de type Transaction qui ont un sender(string), un receiver(string et un amount(int).

BlockChain:

BlockChain est la classe main de ce programme.

La méthode fromFile() permet de lire un fichier tel que blockchain.txt et de créer des Block avec leur Transaction pour ensuite les ajouté au blockchain.

toFile() permet d'écrire sur un fichier tel que blockchain.txt après la dernière ligne. C'est aussi dans toFile() que la création d'un hash s'effectue.(Proof of Work).

validateBlockchain() permet de voir si la BlockChain est valide ou non en vérifiant si le hash donnée dans le blockchain correspond à celui obtenu par le toString() et aussi vérifi si l'index est à jour.

getBalance() permet de vérifier le nombre de bitcoin(amount,int) que un user(string) de la blockchain à en sa possession.

add() permet d'ajouter un block à la blockchain et d'incrémenter l'index.

copyFileUsingStream() est une méthode qui permet de copier entièrement un fichier texte source à une destination.

main() permet l'exécution du prorgamme.

Proof of work:

```
do{
    found =false;
    int targetStringLength = 1 + random.nextInt(16);
    StringBuilder buffer = new StringBuilder(targetStringLength);

    for(int i = 0; i < targetStringLength; i++) {
         int randomLimitedInt = ASCIIBOT + (int)(random.nextFloat() * (ASCIITOP - ASCIIBOT + 1));
         buffer.append((char) randomLimitedInt);
    }
    nonce = buffer.toString();
    System.out.println("nonce "+nonce);

    //System.out.println(previousHash);//print previous hash

    Block newBlock = new Block(ctrIndex, timestamp, new Transaction(sender,receiver,amount), nonce, previousHash);
    System.out.println(newBlock);
    String nB = newBlock.toString();
    hash = Sha1.hash(nB);
    System.out.println(hash);//print hash with at least 5 zeros
    if(hash.startsWith("00000")){
        found = true;
        System.out.println(ctr);
        break;
    }
    ctr++;

}while(!found);
```

Pour mon algorithme pour le proof of work j'ai décidé de faire une boucle do while, qui est en exécution temps et aussi longtemps que un hash(string) commençant par 5 zero soit trouvé. Donc un buffer contenant le nonce généré dans la boucle intérieur selon une largeur aléatoire et des caractère ASCII aléatoire est généré puis ensuite ajouté à un Block(newBlock ici) anisi que tous les valeurs pour créer ce nouveau block et le hash du précédent block. Ensuite, le hash du block.toString() est effectué. On vérifie si le hash commence avec 5 zero. Si il ne commence pas avec 5 zero on recommence la boucle et le ctr(permettant de savoir le nombre de tour effectué est incrémenté) jusqu'à temps que nous avons un string hash avec 5 zero.

Transactions	number of hash trials	toString()	Hash
1	1578753	2018-10-22 16:49:33.817:lucia:max=10.0*<4$R{<"4T0000000045d77dade8708c8df6ca5e167bc07688	00000aba19371ffb75e4003b6998733360422b61
2	194290	2018-10-22 16:51:40.515:robert:max=10.0h%RQW<|Za>00000aba19371ffb75e4003b6998733360422b61	00000b3567de82a93baef006c46549881c11a630
3	693367	2018-10-22 16:52:19.436:max:netflix=1./T:,'wQ^v}Dd"E00000b3567de82a93baef006c46549881c11a630	00000307eb47227dab92efa17d7207c50fa499be
4	1113776	2018-10-22 16:53:46.416:max:uottawa=3.JwASyGJ~,f7Kv:00000307eb47227dab92efa17d7207c50fa499be	00000c686d937e3f6ffac21366e1ba10184aea27
5	3839348	2018-10-22	000001f1dde88320a3913b2e8c

		16:55:08.529:max:quebec=7.1e9@aubl{00000c686d937e3f6ffac21366e1ba10184aea27	1dfb9d02aebd04
6	1147823	2018-10-22 16:57:49.373:max:canada=3.+BY+PRK000001f1dde88320a3913b2e8c1dfb9d02aebd04	0000027550187095692 9ab4b15b86f5489e6e469
7	94799	2018-10-22 16:58:54.499:robert:ikea=1.%j-5WXtqkUnNXES[00000275501870956929ab4b15b86f5489e6e469	00000ee6424226877657fc66e4683574101737d6
8	1088328	2018-10-22 16:59:46.292:lucia:apple=10.Ely;00000ee6424226877657fc66e4683574101737d6	000007d25534bf1491bd963f01101831e0852cf8
9	68370	2018-10-22 17:01:30.087:max:tim=1.)n6-4._6000007d25534bf1491bd963f01101831e0852cf8	0000034965141edfdb4b3134dbc943ab50f5c974
10	3495776	2018-10-22 17:02:10.974:quebec:robert=3.1qn\/0000034965141edfdb4b3134dbc943ab50f5c974	00000cab91d567dc0702905a0e69657c82be0879

Moyenne d’essais pour chaque transaction: 1331463