Devoir n°2: La BlockChain

Miner ID: jzrib059

Description des classes et leur méthodes:

Les classes utilisées afin de bâtir ce programme sont :

(1) `Transaction` : `Transaction` est la classe qui représente une transaction de bitcoin. Cette classe contient les informations vitales à une transaction. C'est-à-dire, le prénom de l'expéditeur, celui du destinataire, ainsi que le montant bitcoin transmis à ce dernier. Cette class contient deux constructeurs. Un dans lequel les variables décrites ci-dessus sont spécifiées et un sans paramètres. En utilisant ce dernier, les variables sont initialisées à partir des méthodes `setSender(String userName)`, `setReceiver(String userName)`, et `setAmount(int amount)`. Les méthodes `getSender()`, `getReceiver()`, et `getAmount()` permettent l'accès aux variables de la classe. Enfin, la méthode `toString()` permet de retourner une chaîne de caractères représentant l'objet instancié.

(2) `Block` : La classe `Block` représente un bloc, soit l'élément individuel contenu dans une blockchain. Un bloc sauvegarde une transaction, la date de son effectuation, ainsi que trois variables dont le but est d'assurer l'intégrité de la blockchain; soit le hashcode de la transaction, le hashcode de la transaction précédente ainsi que la chaîne de chractères « nonce ». Mis à part les « getters » et « setters » des variables données, cette classe contient aussi deux constructeurs, un muni de paramètres, et l'autre non. Le premier permet d'initialiser des transactions actuelles dont la date d'effectuation est créée sur le champ. Le second constructeur permet d'initialiser des blocs effectués dans le passé, auquel cas la date d'effectuation est obtenue à parti du setter de la variable `timestamp`. La classe contient aussi une méthode `toString()` qui représente l'objet instancié.

(3) `BlockChain` : La classe blockchain est la base de données du programme. En effet, c'est la classe qui va sauvegarder toutes les transactions qui ont eu lieu. Dans ce cas, ces transactions seront englobées dans des blocs. Cette classe contient la méthode `add(Block block)` qui ajoute des blocs à la blockchain, et agrandit sa taille en cas de nécessité. La méthode `get(int i)` retourne un bloc à la position i. La classe est aussi munie d'une méthode `getSize()` qui permet d'obtenir la taille de la blockchain en tout temps. La méthode `fromFile(String fileName)` permet la construction d'une blockchain à partir de la lecture d'un fichier contenant les blocs. Quant à la méthode `toFile(String fileName)`, celle-ci permet de transcrire une blockchain dans un fichier `fileName`. La classe contient aussi la méthode `validateBlockChain()` qui détermine si une blockchain est valide ou non. Nous avons aussi la méthode `getBalance(String username, int stop)` qui permet d'obtenir le solde d'un mineur en traversant la blockchain jusqu'au point `stop`. Par ailleurs, la méthode `getBalance(String username)` permet d'obtenir le solde total d'un utilisateur en traversant toute la blockchain. Cette classe a aussi une méthode `main()` qui contient la séquence d'exécution spécifiée.

(4) `NonceGenerator` : Cette classe a pour but de générer des chaînes de caractères aléatoires jusqu'à ce que l'une d'elles fasse en sorte que le hashcode d'un bloc donné commence avec cinq zéros. Cette class contient la méthode `generateNonce(int length)` qui va générer des chaînes de caractères aléatoires de la taille spécifiée par `length`. La méthode `proofOfWork(Block block)` va générer un hashcode valide pour un block donné en faisant appel à `generateNonce(int length)`. Enfin, nous avons la méthode `statistics()` qui compte le nombre d'essais nécessaire dans l'obtention d'un haschode pour

chaque transaction qui y fait appel.

Description de l'algorithme de génération de nonce « Proof Of Work » :

Cet algorithme fait appel à deux méthodes :

`(1) generateNonce(int length)`: cette méthode construit une chaîne de caractères contenant tous les caractères alphanumériques (majuscules et minuscules) ainsi que les caractères spéciaux. Ensuite, dépendant de la longueur de la chaîne (spécifiée par `length`), cette méthode va choisir des caractères de cette chaînes à des index aléatoires et les concaténer pour former la chaîne aléatoire attribuée à la variable `nonce`.

`(2) proofOfWorkGenerator(Block block)`: Cette méthode va générer une chaîne de caractères `nonce` (à partir de `generateNonce(int length)`), l'attribuer à l'object `block`, faire appel à la méthode de cryptage `Sha1` de la façon suivante: `Sha1.hash(block.toString(),Sha1.OUT_HEX)` afin d'obtenir le hashcode de ce bloc, et le vérifie. Si ce dernier ne commence par avec cinq zéros, on entre dans une boucle «`while` » qui continue à générer des listes de caractères pour `nonce`, les attribuer au bloc et tester le hashcode qui en résulte jusqu'à ce que la condition des cinq zéros soit remplie. Ceci tout en comptant le nombre de fois la boucle « `while` » est exécutée (donc le nombre d'essais) grâce à la variable `numberOfTrials`.
Une fois qu'une chaîne de caractères `nonce` résulte en un hascode qui commence avec cinq zéros, on quitte la boucle « `while` », assigne la variable `numberOfTrials` à une liste statique (`transactions`) qui sauvegarde le nombre d'essais par transaction, et assigne le hascode et variable `nonces` trouvés au bloc donné.

Tableau statistique des essais

Nombre de la transaction	Nombre d'essais
Transaction#1	583286
Transaction#2	1402295
Transaction#3	598726
Transaction#4	760446
Transaction#5	1619786
Transaction#6	235992
Transaction#7	1277839
Transaction#8	1053167
Transaction#9	243153
Transaction#10	580446
Moyenne	835513.6

```
Statistics of the hashcode generation:
[583286, 1402295, 598726, 760446, 1619786, 235992, 1277839, 1053167, 243153, 580446]
Numbr of trials of newly added transaction 1 :583286
Numbr of trials of newly added transaction 2 :1402295
Numbr of trials of newly added transaction 3 :598726
Numbr of trials of newly added transaction 4 :760446
Numbr of trials of newly added transaction 5 :1619786
Numbr of trials of newly added transaction 6 :235992
Numbr of trials of newly added transaction 7 :1277839
Numbr of trials of newly added transaction 8 :1053167
Numbr of trials of newly added transaction 9 :243153
Numbr of trials of newly added transaction 10 :580446
```