Spécifications composant 1

Groupe 6

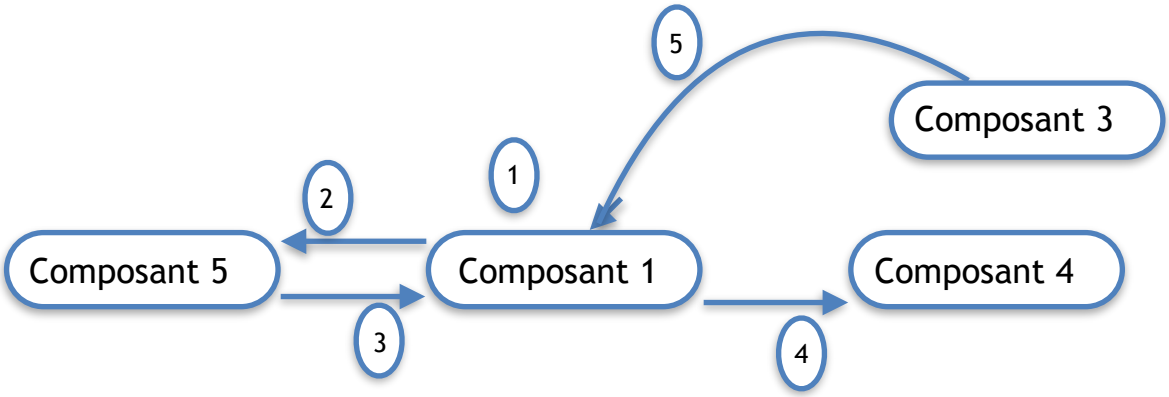
Khalil Bijamane
Laurent Chartrain
Vincent Déchaîne
Romain Moya

Version doc	Date	Auteur(s)	Modifications
1.0	27/01/2015	Jose Luu	Version initiale
1.1	03/03/2016	Groupe 6	Création des spécifications du composant1

Ce document a pour but de rassembler les spécifications concernant le composant 1 du projet de classe.

1. Fonctionnement du composant

L'objectif du Composant 1 est de constituer une interface de fichier pour le blockchain. Pour remplir cette fonction, il va être décliné en plusieurs fonctionnalités différentes comme l'ouverture de fichier, la vérification du fichier, ajouter un block ou bien enrenvoyer un.



1. Ouverture et verrouillage du fichier
2. Appel le Composant 5 pour vérifier chacun des blocs
3. Retour de la fonctionnalité de vérification du bloc
4. Renvoie un bloc de la chaine au Composant 4
5. Ajout de bloc(s) à la chaine provenant du Composant 3

Dans la suite du document, chacune des étapes du schéma sont explicitées afin de mieux comprendre le fonctionnement du composant et son mode de réalisation.

2. Les étapes

2.1. Ouverture du fichier

Notre module ouvre le fichier, pour cela il s'assure de que personne d'autre ne puisse y accéder au même moment et le verrouille.
Cette fonction appelle ensuite la fonction de vérification des blocs avant de terminer l'ouverture du fichier se fait en fonction du résultat de la fonction de vérification.
Si la fonction retourne True, alors on ouvre le fichier. Sinon on empêche son ouverture.

```
Boolean openFile (string path, string nameFile) ;
```

2.2. Vérification des blocs du fichier

Afin de vérifier le fichier, il faut vérifier chacun des blocs présents dans le fichier. Pour cela nous allons faire appel au Composant 5 pour chacun des blocs.Le Composant 5 dont la fonctionnalité principale est la vérification des blocs.

```
Boolean controlBlocks() ;
```

2.2.1 Appel du composant 5

Notre module appelle donc le Composant 5, en lui utilisant la fonction afin d'effectuer la vérification sur le bloc.

```
Boolean nomFonctionComposant5( Block block );
```

Notre module vérifie chacun des blocs du fichier. Si lors de l'appel à la fonction de vérification de bloc elle nous retourne une erreur alors on retourne False, tout en affichant un message d'erreur.
Si l'on ne reçoit aucun False, alors on renvoie True.

2.3. Renvoie d'un bloc

Renvoie le block de la chaine correspondant au numéro passé en argument.

```
Block getBlock (int numeroBlock) ;
```

2.4. Ajout d'un block

La fonction vérifie chacun des blocks de tabBlocks afin de s'assurer qu'ils sont corrects.
De la même façon que pour le point 2.2.1

Si ils sont corrects, alors ils sont rajoutés après le baseBlock dans la chaine.
Retourne True si tout est OK.
False si un des blocks n'est pas valide ou que l'ajout ne s'est pas correctement déroulé.

Boolean addBlocks(Block baseBlock, Hash<Block> blocks) ;

3. Description des erreurs

Pour le composant, chacune des erreurs est gérée par les exceptions (throws) suivantes :

3.1.Etape 2.1

Nom Fonction : Boolean openFile (string path, string nameFile)	
Erreur : mauvais chemin / nom de fichier	***bad path / name of file***
Erreur : fichier non conforme	***File not conform***

3.2.Etape 2.2

Nom Fonction : Boolean controlBlocks();	
Erreur : Block corrompu	***corrupt block***

3.3.Etape 2.3

Nom Fonction : Block getBlock (int numeroBlock) ;	
Erreur : block manquant	***Missing block***
Erreur : Type de numéro incorrect	***Incorrect type of number***

3.4.Etape 2.4

Nom Fonction : Boolean addBlocks(Block baseBlock, hash<Block> blocks) ;	
Erreur : block de base manquant	***Missing baseBlock***
Erreur : Block corrompu	***corrupt block***

4. Plan de tests

4.1.Interaction vis-à-vis des autres composants

On considère les autres composants fonctionnels et renvoyant des données permettant l'exécution de nos fonctions et de notre composant.

4.2.Test du composant

Un main de test se charge d'appeler les fonctions des différents composants et vérifie le les résultats avec ceux attendus. Si le composant n'a pas le comportement souhaité pour le test, un message d'erreur s'affichera. Si tout est correct, il n'y aura donc pas de message affiché d'erreur affiché.
Pour les erreurs les messages suivants s'afficheront

- Boolean openFile (string path, string nameFile)

Si une des valeurs est empêche l'accès ai fichier, alors le message d'erreur suivant doit être renvoyé :

```
"BAD PATH / NAME OF FILE";
```

Si le fichier n'est pas correct alors le message d'erreur suivant doit être renvoyé :
`"FILE NOT CONFORM ";`

- Boolean controlBlocks()

Si un des blocks contrôlés est non conforme alors le message d'erreur suivant doit être renvoyé :
`"CORRUPT BLOCK";`

- Block getBlock (int numeroBlock)

Si le block n'est pas trouvé, le message d'erreur suivant doit être renvoyé :
`"MISSING BLOCK »;`

Si l'argument ne correspond pas au type de nombre des blocks, le message d'erreur suivant doit être renvoyé :
`"INCORRECT TYPE OF NUMBER";`

- Boolean addBlocks(Block baseBlock, Hash<Block> blocks)

Si le block après lequel l'ajout doit s'effectuer n'est pas trouvé alors le message d'erreur suivant doit être renvoyé :
`"MISSING BASEBLOCK ";`

Si un des blocks n'est pas valide alors le message d'erreur suivant doit être renvoyé :
`"CORRUPT BLOCK";`