Lot A : HDFS :

Farah Othmane - Ouali Alami Anass

- **Architecture :**
  - **HdfServer.java** : Classe permettant de créer un serveur qui attend des commandes venant d'un HdfsClient.
  - **HdfsClient.java** : Classe qui contient les commandes qui seront exécutées dans les HdfsServer.
  - **Informations.java** : Classe qui définit un objet contenant les données nécessaires pour l'exécution de la commande "WRITE".
  - **Project.java** : Fichier de configuration contenant les machines (dans cette version en local seulement) et les ports à utiliser.
- **HdfsServer.java** :
  - Ouvrir un `ServerSocket`
  - Recevoir une commande cmd
  - Si cmd == "DELETE"
    - Recevoir le nom du fichier
    - Chercher le chunk du fichier
    - Supprimer ce chunk
  - Si cmd == "WRITE"
    - Recevoir l'objet de classe Informations
    - Créer un nouveau fichier vide
    - Selon le type créer un `KVFormat` ou un `LineFormat` (writer)
    - Ecrire dans ce fichier le message contenu dans informations
  - Si cmd == "READ"
    - Ouvrir le fichier en lecture
    - Ajouter dans une ArrayList les KV obtenues à partir de la lecture
    - Envoyer l'ArrayList
- **HdfsClient.java :**
  - **Méthode HdfsWrite:**
    - Ouvrir le fichier selon son type (`KVFormat` ou `LineFormat`)
    - Lire le fichier et mettre les KV obtenues dans une ArrayList
    - Calcule des lignes à écrire dans chaque serveur et les lignes restantes (que nous mettrons dans le dernier serveur)
    - Pour chaque serveur
      - Ouvrir le socket de connexion
      - Envoyer la commande WRITE
      - Créer un objet de class Informations (contenant le type du fichier, son nom,les nombre de lignes à écrire, le message : suite de KV)
      - Envoyer cet objet de classe Informations
      - Fermer la connexion
  - **Méthode HdfsDelete :**
    - Réaliser une boucle sur le nombre de serveurs
    - Dans chaque itération i :
      - Ouvrir un socket de connexion
      - Envoyer "DELETE" au serveur i

                - Envoyer le nom du fichier au serveur i
                - Fermer la connexion

    - **Méthode HdfsRead**
        - Créer un ArrayList de type KV nommé `files`
        - Parcourir tous les serveurs
        - Dans chaque itération i :
            - Ouvrir un socket de connexion
            - Envoyer la commande “READ” puis le nom du fichier au serveur i
            - Enregistrer le ArrayList de fragments reçus à la fin du ArrayList files
        - Créer un fichier de type `KVFormat` nommé `createdFile`
        - Si localFSDestFname est différent de null :
            - Créer le fichier `localFSDestFname`
        - Sinon :
            - Créer le fichier “./hdfsFname”
        - Écrire le contenu du ArrayList `files` dans `createdFile`

Lot B: Hadoop
BERNOUSSI Aymen - MESKINE Hatim

- **WorkerImpl.java:**
    - **Méthode runMap :**
        - Ouvrir le reader et le writer
        - Commencer le map
        - Informer avec le callback que map est fini
        - Fermer les fichier
- **CallBack.java:** Interface contenant deux méthodes FinishMap et waitForRunMap.
    - **FinishMap** : permet de savoir si un traitement map a terminé
    - **waitForRunMap** : permet d’attendre le traitement de tous les map.
- **CallBackImpl.java:** Implémentation de l’interface CallBack en utilisant un sémaphore waitFotAllMaps initialisée à 0 pour être bloquante.
    - **FinishMap** : permet de donner un jeton au sémaphore pour indiquer la terminaison d’un traitement map
    - **waitForRunMap** : permet de prendre un nombre de jetons égal au nombre de serveurs pour bloquer si éventuellement tous les traitements map ne sont pas terminés
- **Job.java:** Classe permettant de lancer les runMap sur tous les serveurs et puis d’appliquer reduce pour obtenir le résultat final.
    - **SetInputFormat :** changer le type du fichier
    - **SetInputFname :** changer le nom du fichier
    - **startJob:**
        - Créer un objet de classe work et le lance

        - work hérite de Thread pour permettre de lancer plusieurs traitements en concurrence
    - **Work :**
        - **Run :**
            - Réaliser une boucle sur le nombre de serveurs
            - Pour chaque itération i :
                - Créer l'URL
                - Chercher le worker correspondant au serveur
                - Selon le type du fichier :
                    - Créer un reader correspondant au type du fichier
                    - Créer un nouveau fichier (writer)
                    - Créer un objet KVFormat correspondant au fichier crée
                - Appliquer un runmap sur le worker
            - Attendre les runmaps
            - Lire les fichiers créés avec HdfsRead
            - Créer un fichier de type KVFormat
            - Réaliser un reduce sur le résultat de HdfsRead pour écrire dans le fichier créé.