


REMARQUE: l'icône  que vous verrez en haut à droite des sources vous permet de les télécharger dans la version originale et complète. En effet, le copier-coller à partir du pdf ne marche généralement pas. Cliquez dessus ou copiez l'adresse du lien, ça ouvre le navigateur, puis tapez CTRL-A, CTRL-C pour tout copier ; puis collez dans l'autre fenêtre.

1. Introduction

Cette semaine, vous allez travailler sur le cluster Hadoop, utiliser et programmer HDFS (Hadoop File System). C'est l'espace de stockage des fichiers distribués. Il ressemble à l'arbre Unix et il y a des commandes très similaires pour manipuler les fichiers et les dossiers. Il y a aussi une API Java permettant d'écrire des programmes de traitement des fichiers HDFS.

Le but du TP est d'apprendre les concepts et les commandes afin de bien gérer les fichiers sur HDFS. Dans un second temps, vous utiliserez l'API Java pour afficher des fichiers.

2. Travail avec le cluster Hadoop

Démarrez la machine avec le système Linux. Connectez-vous. Le mot de passe initial vous sera donné en séance. Ce compte est spécifique à la salle de TP.

Ouvrez un terminal. Tapez la commande `ssh hadoop`. S'il faut, acceptez la signature ECDSA (demandé seulement la première fois pour accepter définitivement cet hôte). Entrez votre mot de passe. Vous voilà connecté(e) au cluster. Pour vous déconnecter, il suffira de taper `exit` ou `CTRL-D`.

NB: vous ne pourrez avoir qu'une connexion de type texte car le serveur hadoop n'est pas dimensionné pour de nombreuses sessions graphiques. Le travail en mode graphique se fera seulement sur le poste devant vous et la connexion ssh ne servira qu'à lancer les programmes.

Dans le cluster la machine principale s'appelle `master.cluster.iut` (son nom est visible dans votre prompt). `master` est son nom dans le cluster et `hadoop` est son nom vu de l'extérieur¹. Comme pour un artiste, il y a le nom de scène et le nom réel. Le cluster contient quatre autres machines appelées `hadoop1` à `hadoop4`. On pourrait en rajouter autant qu'on veut, mais l'IUT n'a pas vocation à devenir un Data Center. Vous ne pourrez pas vous connecter sur les autres machines car elles ne servent que pour le stockage des données et les calculs.

Lorsque vous êtes sur `hadoop` (alias `master`), vous avez les mêmes fichiers que dans votre compte de la salle de TP. Les fichiers sont partagés par NFS et ils sont en réalité sur un serveur dans la salle des machines. Pour le constater, il suffit de taper `df .`, vous verrez le n°IP du serveur. Ce montage NFS va vous permettre de travailler avec les outils de votre poste client, par exemple compiler un programme et trouver le binaire exécutable à l'identique sur `hadoop`.

Faites un essai rapide :

- Ouvrez un navigateur de fichiers sur votre poste local, créez un nouveau document texte appelé `bonjour.txt`, mettez quelque chose dedans.

¹C'est parce que le nom `hadoop` a été mis dans le DNS de l'IUT et `master` dans le DNS du cluster. On aurait pu uniformiser. De toutes façons, ça ne vous gênera pas beaucoup.

- Dans `hadoop`, listez les fichiers avec `ls`, affichez le fichier `bonjour.txt`, éditez-le avec `nano` ou `vi` (il n'y a aucun éditeur graphique style `gedit` ou `geany` car ça chargerait trop le serveur `hadoop`).
- Affichez le contenu du fichier dans le navigateur de fichiers, vous verrez qu'il a été modifié.
- Si vous créez un fichier dans la fenêtre `ssh`, vous devrez rafraîchir le navigateur de fichier (`CTRL-R`).

3. Manipulations sur HDFS

3.1. Commandes de base de HDFS

La partie intéressante du TP commence maintenant avec l'utilisation du système de fichiers distribués HDFS. Ça va concerner deux comptes : votre compte Linux ordinaire (nommé `local` par Hadoop) et le compte que vous avez sur HDFS. Les commandes `ls`, `mkdir`, etc. concernent le compte local ; les commandes `hdfs dfs -ls`, `hdfs dfs -mkdir`, etc. sont pour le compte sur HDFS. Essayez de ne pas confondre les fichiers des deux comptes.

Voici quelques commandes à essayer rapidement (parce qu'elles sont comme les commandes Unix que vous connaissez déjà) dans la fenêtre `ssh` :

- `hdfs dfs -ls` : elle n'affiche rien pour l'instant car elle s'adresse par défaut à votre dossier HDFS personnel `/user/votre_login` qui est vide.
Le temps de réaction de la commande est dû au grand nombre de jars présents dans son `CLASSPATH`. On n'y peut rien dans cette version de Hadoop.
- `hdfs dfs -ls /` : affiche ce qu'il y a à la racine HDFS. Vous pouvez descendre inspecter les dossiers que vous voyez. Exemple `hdfs dfs -ls /user`. Il n'y a pas de commande équivalente à `cd`, parce qu'il n'y a pas de notion de dossier courant dans HDFS, donc à chaque fois, il faut remettre le chemin complet. C'est une habitude à prendre.
- `hdfs dfs -ls -R -h /var` : affiche les fichiers des sous-dossiers, avec une taille arrondie en Ko, Mo ou Go (l'option `-h = human units` existe aussi sur Unix).
- `hdfs dfs -mkdir dossier` : crée un dossier dans votre espace HDFS, c'est à dire `/user/votre_login/dossier`. Notez que la taille d'un dossier sera toujours 0.

Reprenez le fichier appelé `bonjour.txt`, vérifiez qu'il n'est pas vide.

- Copiez ce fichier sur HDFS par `hdfs dfs -put bonjour.txt`. Utilisez `hdfs dfs -ls -R` pour vérifier.
- `hdfs dfs -cat bonjour.txt` : affiche le contenu. Il n'y a pas de commande `more` mais vous pouvez faire `hdfs dfs -cat bonjour.txt | more`
- `hdfs dfs -tail bonjour.txt` : affiche le dernier Ko du fichier.
- Supprimez ce fichier de HDFS par `hdfs dfs -rm bonjour.txt`.
- Remettez à nouveau ce fichier par `hdfs dfs -copyFromLocal bonjour.txt` (vérifier avec `hdfs dfs -ls`). Cette commande est similaire à `hdfs dfs -put`.
- `hdfs dfs -chmod go+w bonjour.txt` (vérifiez son propriétaire, son groupe et ses droits avec `hdfs dfs -ls`)
- `hdfs dfs -chmod go-r bonjour.txt` (vérifiez les droits)
- `hdfs dfs -mv bonjour.txt dossier/bonjour.txt` (vérifiez avec `hdfs dfs -ls -R`)
- `hdfs dfs -get dossier/bonjour.txt demat.txt` : transfère le fichier de HDFS vers votre compte Linux en lui changeant son nom. Cette commande ne serait pas à faire avec de

vraies méga-données !

- `hdfs dfs -cp dossier/bonjour.txt dossier/salut.txt` (vérifiez)
- `hdfs dfs -count -h /share` : affiche le nombre de sous-dossiers, fichiers et octets occupés dans `/share`. Cette commande correspond à peu près à `du -h` dans Unix.
- `hdfs dfs -rm dossier/bonjour.txt` (vérifiez avec `hdfs dfs -ls dossier`).
- `hdfs dfs -rm -r dossier` (vérifiez avec `hdfs dfs -ls`).

Avez-vous bien compris qu'il y a deux espaces de fichiers :

- les fichiers et dossiers de votre compte Unix, visibles avec `ls` à la fois sur le poste local et dans la fenêtre ssh,
- les fichiers et dossiers HDFS, visibles en faisant `hdfs dfs -ls` dans la fenêtre ssh.

Supprimez les fichiers locaux `bonjour.txt` et `demat.txt`.

La documentation de toutes les commandes est sur [cette page](#).


Pour nommer les fichiers dans un environnement mixte (local et HDFS), on peut utiliser la notation URI : `hdfs://nomcomplet`. Par exemple `hdfs dfs -ls hdfs:///share`. On peut ainsi distinguer `file:///tmp/toto` de `hdfs:///tmp/toto`.

3.2. Téléchargement d'un fichier sur HDFS

Voici maintenant des opérations qui font télécharger un fichier appelé `arbres.csv`, voici son URL <https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/fichiers/arbres.csv>. Il s'agit de statistiques sur les arbres remarquables de Paris.

- Définissez d'abord une variable pour contenir l'URL : 

```
URL=https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/fichiers/arbres.csv
```

- Téléchargez le fichier `arbres.csv` dans votre compte, puis copiez-le sur HDFS : 

```
wget $URL  
hdfs dfs -put arbres.csv
```

Vérifiez sa présence et sa taille sur HDFS puis supprimez-le de votre compte local ainsi que de HDFS.

- Voici une autre façon de placer un fichier sur HDFS sans le stocker dans votre compte : 

```
wget -O - $URL | hdfs dfs -put - arbres.csv
```

Attention, il y a deux tirets isolés. Et, si vous avez oublié de supprimer le fichier de HDFS, il vous signale `file exists`.

Voici une dernière commande, juste pour la curiosité :

- `hdfs fsck /user/votre_login -files -blocks` : affiche la liste des blocs utilisés par vos fichiers. Les blocs font 64 Mo chacun (256 ou 512 Mo sur un cluster)...

Il est demandé de ne pas essayer d'autres commandes afin de ne pas casser HDFS. Le niveau de protection du cluster est assez bas et vous pourriez le casser avec certaines commandes. Sachez que toutes vos commandes sont enregistrées.

3.3. Téléchargement de nombreux fichiers

Les opérations de cette section ne sont pas possibles cette année, à cause du *shutdown* de l'administration américaine. Le site de téléchargement de la NOAA a été arrêté. Pour cette raison, les fichiers météo de l'an dernier ont été conservés. S'il n'y avait pas eu ce problème, vous auriez pu faire ce qui suit. Passez directement à la section 3.4 suivante.

Vous allez chacun(e) commencer à rajouter des données volumineuses dans HDFS afin de préparer les TPs suivants. Ce sont les archives météorologiques de stations placées dans le monde entier. Ils sont collectés par le National Climatic Data Center (NCDC) qui dépend de la National Oceanic and Atmospheric Administration (NOAA).

Le site principal est <http://www1.ncdc.noaa.gov/pub/data/noaa/>. Ouvrez-le dans le navigateur. Cette page contient un dossier par année. Allez dans l'une des années. Dans ces dossiers d'année, il y a de nombreux fichiers, un par station. Les stations sont représentées par leur identifiant (code USAF), par exemple celui de l'aéroport de Lannion est 071180. Ne téléchargez pas cette station, car elle l'est déjà et par contre, on va vous indiquer laquelle télécharger, lire la suite. Chacun(e) d'entre vous aura tous les relevés d'une station à télécharger².

Toutes les stations n'ont pas des relevés depuis 1901, il y en avait très peu gérées à cette époque. Lannion n'a commencé qu'en 2001. Alors justement, la station que vous aurez à télécharger ont au moins 75 ans d'activité. Voici ce qu'il faut faire.

- Téléchargez le script [GetNOAA.sh](#), en principe dans Téléchargements. Surtout ne le copiez pas sur HDFS, car c'est un script bash, donc à exécuter uniquement sur la machine hadoop, dans la fenêtre ssh.
- Rendez-le exécutable : `cd ~/Téléchargements ; chmod u+x GetNOAA.sh`.

Ce script permet de télécharger tous les relevés météo de votre station. Voici son cœur : 

```
station=$1
hdfs dfs -mkdir -p /share/noaa/data/$station
hdfs dfs -chmod go+rx /share/noaa/data/$station
# récupération des données de la station
for annee in 1901 1902 1903 ... 2019 ; do
    nom=$station-99999-$annee.gz
    fichier=/share/noaa/data/$station/$nom
    url=http://www1.ncdc.noaa.gov/pub/data/noaa/$annee/$nom
    wget -nv -O - $url | hdfs dfs -put - $fichier
    hdfs dfs -chmod go+r $fichier
done
```

Le vrai script est nettement plus complexe car en réalité, les fichiers ne sont pas tous nommés station-99999-annee.gz et il faut vérifier la taille après transfert. Et pour finir, les fichiers sont décompressés lors du téléchargement.

- Dans le shell sur hadoop, tapez la commande `mail` sans paramètre. Ça va afficher la liste de vos messages internes. Les messages ont pour sujet « TP1 partie noaa » et ont été envoyés

²On espère que l'IUT de Lannion ne va pas être blacklisté par tous ces transferts simultanés.

par `root`.

- Tapez le numéro du message, par exemple 1 ou *entrée* pour afficher le message. Il vous dit quoi faire : lancer le script `GetNOAA.sh` avec le numéro de votre station.
- Quittez mail en tapant `x`. Si vous voulez revoir le message, tapez `mail -f`.
- Lancez le script comme indiqué : `./GetNOAA.sh votrenumérodestination`

Continuez la suite du TP dans un autre shell pendant que les téléchargements ont lieu.

3.4. État du cluster

Les services Hadoop génèrent des pages web automatiquement pour permettre de suivre le fonctionnement. Cliquez sur ce lien : <http://hadoop> pour vous y connecter.

La page que vous voyez propose plusieurs liens vers différents services Hadoop. On va s'intéresser à HDFS (1^{er} lien) cette semaine. Cliquez sur son lien ; ça amène sur la page *Overview*. Dans le tableau *Summary* vous avez l'espace total, l'espace utilisé, l'espace libre, avec des valeurs en %, des liens vers les DataNodes vivants, morts ou en train de se désactiver (*decommissioning nodes*).

Tout en haut, il y a une barre verte avec différents liens. Cliquez sur **Datanodes**. Vous voyez la capacité et la charge de chaque DataNode. La colonne *Last Contact* indique le nombre de secondes depuis le précédent « battement de cœur » (*heartbeat*) envoyé par le Datanode au Namenode, c'est le terme employé pour un signal périodique de bon fonctionnement. Il y a un contact toutes les 3 secondes. Un Datanode est considéré comme mort lorsqu'il n'a pas donné signe de vie depuis 10 minutes. Vous pouvez rafraîchir pour voir l'évolution.

Toujours en haut, il y a un bouton **Utilities** avec un item **Browse the file system**. Vous pouvez parcourir l'arbre des fichiers HDFS, par contre, vous n'êtes pas authentifié(e) (*Dr Who*), donc vous ne pourrez pas voir les fichiers protégés contre la lecture par tout le monde. Par exemple, descendez dans `/user/vous`, il y a un message d'erreur tout en haut. Descendez dans `/share`. Vous allez voir plusieurs dossiers et fichiers. En cliquant sur le lien proposé, vous aurez des informations sur les blocs et les machines contenant ce fichier. Ne cliquez sur **Download** que pour un petit fichier.

Dans le menu **Utilities**, il y a un item **Logs**. Vous pouvez aller voir le fichier `hadoop-hdfs-namenode-master.out` et prendre la mesure de la complexité du bestiau rien qu'avec le `classpath` visible au début. Le logo Hadoop 🐘 n'a pas été choisi par hasard.

Notez que vous ne pourrez pas revenir à la page d'accueil avec le bouton *back*, il faut re-saisir l'URL ou dérouler l'historique et revenir 2 crans en arrière.

3.4.1. Changement de proxy (tout à fait optionnel)

Avec l'URL <http://hadoop> vous n'aurez pas accès aux machines internes du cluster (hadoop1, hadoop2...) car elles ne sont pas référencées dans le DNS du réseau. Pour les voir, il faut d'abord configurer le navigateur comme ceci :

1. Ouvrez les préférences de Iceweasel (avec le bouton à trois lignes complètement à droite dans la barre d'adresse),
2. Allez sur l'onglet **Avancé** et le sous-onglet **Réseau**,
3. Appuyez sur le bouton **Paramètres...** en face de « Configurer la façon dont Iceweasel se connecte à Internet »,

4. Cocher le bouton **Configuration manuelle** du proxy, puis saisissez 192.168.100.99 dans « Proxy HTTP » et 80 dans « Port », cochez **Utiliser ce serveur proxy pour tous les protocoles** et aussi **DNS distant**.

Vous pouvez maintenant ouvrir l'URL <http://master.cluster.iutlan>. Attention, cet URL n'a aucune validité en dehors de la salle I106 et sans la configuration qu'on vient de faire. Ajoutez-le en favoris.

Pour récupérer un accès normal à internet dans Iceweasel, vous serez obligé(e) de défaire partiellement cette configuration. Ou alors, utilisez Chromium pour accéder à internet et gardez Iceweasel pour le cluster.

1. Ouvrez les préférences de Iceweasel,
2. Allez sur l'onglet **Avancé** et le sous-onglet **Réseau**,
3. Appuyez sur le bouton **Paramètres...**,
4. Cocher le bouton **Utiliser les paramètres proxy du système**, puis décochez **DNS distant**.


Vous pourrez revenir assez rapidement à la configuration permettant de voir le cluster car le n° du proxy est mémorisé.

Si vous ne voulez voir que la machine `master.cluster.iutlan`, alors il suffit d'ouvrir l'URL <http://hadoop> et il n'y a pas besoin de configurer le proxy.

4. Programmation avec l'API HDFS

Dans cette partie, nous allons programmer en langage Java en utilisant Eclipse. C'est pour préparer le TP sur MapReduce. Dans les sections suivantes, vous allez créer différents petits projets. Ils vont tous être pareils : un source java et l'importation de bibliothèques Hadoop pour fabriquer un `fichier.jar`.

Le principe est le suivant : vous allez éditer le source et compiler sur votre poste de travail de la salle de TP avec Eclipse, et vous exécuterez le programme sur le cluster Hadoop.

Les bibliothèques nécessaires pour compiler n'ont pas été installées sur votre poste. Elles sont sur la machine hadoop mais doivent être copiées vers votre poste de travail afin de pouvoir compiler. Voici les étapes à faire une seule fois, dans la fenêtre `ssh hadoop` : 

```
mkdir -p ~/lib/hadoop
cp /usr/lib/hadoop-hdfs/hadoop-hdfs-2.8.4.jar ~/lib/hadoop
cp /usr/lib/hadoop/client/hadoop-common-2.8.4.jar ~/lib/hadoop
cp /usr/lib/hadoop/client/hadoop-mapreduce-client-core-2.8.4.jar ~/lib/hadoop
```

Votre dossier `~/lib/hadoop` contient maintenant les archives permettant de compiler un projet HDFS dans Eclipse.

4.1. Affichage d'un fichier simple

Pour un premier projet HDFS, nous allons écrire un programme qui compte les lignes du fichier `/share/paris/arbres.csv`.

- Créez un dossier appelé TP1 dans votre compte. Ça sera la *workspace* pour ce TP.

- Lancez Eclipse. Il va vous demander dans quel workspace travailler : indiquez le dossier TP1.
- Créez un nouveau projet type Java Project, appelé CompterLignesArbres.
- Modifiez les propriétés du projet : clic droit sur le projet, item Properties, onglet Java Build Path, sous-onglet Libraries, cliquez sur le bouton Add External Jars..., allez dans le dossier lib/hadoop de votre compte, sélectionnez tous les éléments et validez. Fermez le dialogue. Pour info, ces chemins sont enregistrés dans un fichier caché appelé .classpath dans le projet.
- Avec l'assistant, créez une classe appelée comme le projet.

Copiez-collez le source suivant dans la classe CompterLignesArbres :



```
import java.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;


public class CompterLignesArbres
{
    public static void main(String[] args) throws IOException
    {
        // nom complet du fichier
        Path nomcomplet = new Path("/share/paris/arbres.csv");

        // ouvrir le fichier sur HDFS
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        FSDataInputStream inStream = fs.open(nomcomplet);
        try {
            // préparer un lecteur séquentiel
            InputStreamReader isr = new InputStreamReader(inStream);
            BufferedReader br = new BufferedReader(isr);

            // parcourir les lignes une par une
            String ligne = br.readLine();
            while (ligne != null) {

                // traiter la ligne courante
                System.out.println(ligne);

                // passer à la ligne suivante (retourne null si fin fichier)
                ligne = br.readLine();
            }
        } finally {
            // fermer le fichier quoi qu'il arrive
            inStream.close();
            fs.close();
        }
    }
}
```


Téléchargez le fichier suivant <https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/fichiers/Makefile>, il est à mettre dans le dossier du projet (à côté des sous-dossiers bin et src), donc le mieux est de faire la commande suivante en étant dans le dossier du projet : 

```
wget https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/fichiers/Makefile
```

```
# Makefile pour compiler et lancer les projets HDFS de Eclipse

run:    projet.jar
        hadoop jar $<

SOURCES=$(shell find src -name *.java)

projet.jar:    $(shell fgrep -l 'void main' $(SOURCES)) $(SOURCES)
               @mkdir -p bin
               javac -cp $(subst $(eval) ,.,$(wildcard ~/lib/hadoop/*.jar)) $^ -d bin
               jar cfe $@ $(subst /,.,$(basename $(<:src/%=))) -C bin $(dir $(<:src/%=))


clean:
        rm -fr projet.jar bin
```

Ce Makefile est assez complexe car il est capable de s'adapter à n'importe quelle configuration de projet : package ou pas. Par contre, il faut impérativement que les `.jar` soient mis dans `~/lib/hadoop`, c'est à dire directement dans votre compte et pas dans un sous-dossier.

Allez maintenant dans la fenêtre shell connectée à `hadoop`, descendez dans le dossier du projet, là où il y a le dossier `src` et le fichier `Makefile` et tapez `make`. Le programme se compile puis se lance. Il affiche toutes les lignes du fichier `arbres.csv`.


Le travail qui vous est demandé n'est pas compliqué : au lieu d'afficher toutes les lignes, il faut seulement les compter et afficher leur nombre à la fin.

4.2. Affichage de champs d'un fichier CSV

Le but de cet exercice est d'afficher seulement l'année et la hauteur de chaque arbre du fichier `arbres.csv`. Il décrit les arbres remarquables de Paris. Il provient de <http://opendata.paris.fr>. Chaque ligne décrit un arbre : position GPS, arrondissement, genre, espèce, famille, année de plantation, hauteur, circonférence, etc. Le séparateur est `' ; '`. Il faut juste signaler une erreur ligne 38 dans le champ famille du fichier original. Voici ses premières lignes. 

```
Geo point;ARRONDISSEMENT;GENRE;ESPECE;FAMILLE;ANNEE PLANTATION;HAUTEUR;CIRCONF...
(48.857140829, 2.295334553);7;Maclura;pomifera;Moraceae;1935;13.0;;Quai Branly...
(48.8685686134, 2.31331809304);8;Calocedrus;decurrens;Cupressaceae;1854;20.0;...
```

- Dans Eclipse, créez un nouveau projet appelé `AnneeHauteurArbres` en copiant-collant le projet précédent. Il n'est pas nécessaire de refaire la création, un simple `CTRL-C CTRL-V` sur le premier pour créer le second. Il aura les mêmes options et le même contenu. Utilisez « Refactor/rename » (raccourci `F2` ou `MAJ-ALT-R`) pour renommer la classe principale. Ignorez les problèmes rencontrés – ils viennent du fait que la classe pouvant être appelée de l'extérieur, ça pourrait planter si on oublie de renommer l'appel aussi.

- Ajoutez une classe publique appelée `Arbre`. Cette classe va représenter l'une des lignes du fichier et elle aura des accesseurs pour les champs. Voici cette classe³ : 

```
public abstract class Arbre
{
    static String[] champs;

    public static void fromLine(String ligne)
    {
        champs = ligne.split(";");
    }

    public static double getAnnee() throws NumberFormatException
    {
        return Double.parseDouble(champs[5]);
    }
}
```

Toutes ses méthodes sont statiques, en fait c'est une classe sans instance, indiqué par le mot clé `abstract`. En patrons de conception, c'est une [fabrique](#) ayant des [méthodes de fabrique statique](#). Cette technique permet d'éviter les allocations mémoire. Elle ne fonctionne que parce qu'il n'y a qu'un seul arbre considéré à un moment donné. Il faudrait créer des instances, donc tout revoir, si elle devait représenter différents arbres simultanément. Cependant, ce ne serait pas une bonne chose pour l'efficacité, il faut éviter cela.

- La méthode `fromLine` sépare une ligne du fichier en mots, chacun est mis dans `champs`. Retenez bien la méthode `split()` qui sépare une chaîne en mots selon un séparateur.
- La méthode `getAnnee` convertit le 6^e champ en double, si c'est possible, sinon elle émet une exception. Vous devrez penser à intercepter les exceptions dans le programme principal, par exemple en ignorant toute la ligne de données. Au passage, est-ce que c'est correct que l'année soit un double ? Il faudrait aller voir dans le fichier ce qu'il en est et sans doute changer pour des entiers.

Votre travail consiste à étendre et employer cette classe pour afficher ligne par ligne l'année de plantation et la hauteur des arbres du fichier. L'affichage doit se faire dans la classe principale. Celle-ci fait une boucle sur toutes les lignes, elle les fournit à `Arbre.fromLine()` puis affiche les champs voulus.

4.3. Affichage d'un fichier compact (optionnel)

On se tourne maintenant vers l'un des fichiers météo de la NOAA. On va prendre pour exemple `/share/noaa/isd-history.txt`. Il vient de [cet URL](#). C'est un fichier « compact » : il y a des champs comme dans un CSV, mais sans séparateur – les champs sont de taille fixe. Tapez ceci pour voir ses dernières lignes :

```
hdfs dfs -tail /share/noaa/isd-history.txt
```


³Désolé pour le mélange anglais-français. Vous pouvez tout mettre en anglais si ça vous pique les yeux.

Voici une description des champs :

offset	taille	exemple	signification
0	6	225730	USAF = Air Force Datsav3 station number
7	5	99999	WBAN = NCDC WBAN number
13	29	LESUKONSKOE	Station name
43	2	US	FIPS country ID (pays)
48	2	KS	ST = State for US stations
51	4	LFRD	CALL = ICAO call sign (code aéroport)
57	7	+64.900	LAT = Latitude in decimal degrees
65	8	+045.767	LON = Longitude in decimal degrees
74	7	+0071.0	ELEV = Elevation in meters (altitude)
82	8	19590101	BEGIN = Beginning Period Of Record (YYYYMMDD)
91	8	20140206	END = Ending Period Of Record (YYYYMMDD)

Par exemple le nom de la station commence au caractère 13 (premier = n°0) et fait 29 caractères de long.

Devinez quoi : créez un projet qui va afficher le code USAF, le nom, le pays (FIPS country ID) et l'altitude de chaque station. Voici les étapes :

- Copiez-collez le projet `AnneeHauteurArbres`. Renommez-le ainsi que la classe principale en `PaysNomAltStations` (tout avec le refactor/rename, SHIFT ALT R). Changez le nom du fichier à lire.
- Créez une classe `Station` : 

```
public abstract class Station
{
    static String ligne;

    public static void fromLine(String ligne)
    {
        Station.ligne = ligne;
    }

    public static String getNom() throws IndexOutOfBoundsException
    {
        return ligne.substring(13, 13+29);
    }

    public static Double getAltitude()
        throws IndexOutOfBoundsException, NumberFormatException
    {
        return Double.parseDouble(ligne.substring(74, 74+7));
    }
}
```

Dans cette classe, le découpage en champs et la conversion en valeurs se fait au moment où on en

a besoin. Vous rajouterez les méthodes manquantes.

Dans la boucle de parcours du fichier, il faut ignorer les 22 premières lignes du fichier parce que ce ne sont pas des données.

5. Travail à rendre

Ouvrez un shell dans le dossier TP1 (le workspace du TP). Tapez ces commandes shell :



```
for d in */Makefile ; do pushd $(dirname $d) ; make clean ; popd ; done
tar cfz tp1.tar.gz $(for d in */Makefile ; do dirname $d ; done)
tar tfvz tp1.tar.gz
```

Vérifiez que l'archive contient bien les projets du TP.

Déposez le fichier `tp1.tar.gz` sur la page Moodle dédiée.