FICHIERS TEXTES

Lire et écrire

- Les fichiers
  - Un fichier permet de conserver de façon pérenne des données sur un support qui peut conserver l'information de façon permanente, type disque dur, clé USB…
  - Il est ainsi possible de conserver des données entre deux sessions successives d'utilisation d'un logiciel, comme un traitement de texte
  - On distingue en général deux grands types de fichiers
    - Fichiers binaire
    - Fichiers texte

- Les fichiers
  - Dans un fichier binaire les données sont enregistrées en codage binaire
  - L'organisation de l'ensemble des données en binaire enregistrées dans le fichier est propre au logiciel qui l'a enregistré et qui « connaît » la structure
    - Fichier exécutable d'un logiciel (par exemple les fichiers .class produit par le compilateur Scala sont en bytecode)
    - Fichier « document » de certains traitement de texte, images, vidéo
    - …

- Les fichiers
  - Dans un fichier texte, les données sont bien sûr enregistrées en binaire, mais elles sont toutes de type caractère
    - Par exemple la valeur numérique 12.5 sera enregistrée dans le fichier en enregistrant successivement les 4 caractères: ‘1’, ‘2’, ‘.’ et ‘5’
    - Il y a une similitude avec le fonctionnement de la console sur laquelle tous les affichages se font au format caractère
  - Les caractères sont organisés les uns à la suite des autres.
  - Les données peuvent être structurées de façon textuelle sous forme de langages (HTML, XML, SVG …)
    - Fichiers au format texte plat (brut, sans mise en forme, CSV, fichier source Java…)
    - Fichiers « documents » avec mise en forme (HTML)
    - …

- Les fichiers
  - Dans un fichier, les données sont enregistrée séquentiellement, c'est à dire au fur et à mesure où elles sont « écrites » dans le fichier
    - Cela implique qu'un fichier se lit aussi séquentiellement du début du fichier à la fin. Il est nécessaire de lire les données depuis les premières pour pouvoir atteindre les dernières
  - Il existe d'autres types de support pour conserver des données comme par exemple les bases de données qui sont optimisées pour organiser et exploiter les données qui y sont enregistrées
  - Nous ne verrons ici que l'utilisation des fichiers textes
    - Les deux opérations principales sont d'écrire et de lire un fichier texte
    - Il est important de comprendre que les données ne peuvent pas être enregistrées en vrac, mais qu'il faut les organiser de façon systématique pour pouvoir ensuite être en mesure de les relire correctement en leur donnant leur signification pour pouvoir les réutiliser.

- Les fichiers
  - Un format de structuration des données dans un fichier texte est le CSV ou Comma Separated Values
    - Il consiste à écrire toutes les données correspondant à une entité (un rendez-vous, un livre, une personne…) sur une seule ligne et à écrire chaque données dans un ordre précis en les séparant par une virgule (ou un point-virgule ou une tabulation ou tout autre caractère qui peut être utilisé comme caractère de séparation)
    - Si on enregistre une liste de personne représentées par leur genre, le nom, leur prénom et leur date de naissance, le format CSV correspondant pourra être

      M,Dupont,Gérard,1934

      F,Porte,Nadine,1950

      …

- Les fichiers
  - Un format de structuration des données dans un fichier texte est le CSV ou Comma Separated Values
    - Cette organisation permet de pouvoir relire le fichier texte en étant en mesure d'identifier les données et de leur associer leur rôle
    - Le fichier texte sera lu ligne par ligne, en sachant que chaque ligne représente une personne
    - On va ensuite « spliter » la ligne en sous-partie en utilisant le caractère virgule pour délimiter les données (en connaissant l'ordre dans lequel les données sont enregistrées)
    - Si on n'utilise pas cette organisation l'enregistrement des données donnera quelque chose comme

      MDupontGérard1934FPorteNadine1950

      Et il sera impossible de délimiter et d'identifier les données à la relecture (ce qui rend l'enregistrement dans le fichier inutile)

- Fichiers
  - Ecrire/enregistrer des données dans un fichier
    - Import d'un package Java

      import java.io.PrintWriter
    - Instanciation d'un objet PrintWriter

      val fw = new PrintWriter("fichier.txt")
      - fichier.txt est le nom du fichier physique qui sera créé sur le disque
    - Ecriture d'une ligne
      - *fw.println(…)*
      - Les paramètres de la méthode println sont les mêmes que ceux de la méthode d'affichage à la console
      - Cette méthode est à utiliser autant de fois que nécessaire, par exemple dans une boucle
    - Fermeture du fichier
      - *fw.close*

- Fichiers
  - Ecrire/enregistrer des données dans un fichier
    - Par défaut, la méthode d'ouverture d'un fichier en écriture va soit créer un nouveau fichier initialement vide si il n'existe pas encore, soit supprimer le fichier et le recrée vide si il existe déjà
    - Il est souvent utile de pouvoir rajouter des données à la suite de celles qui sont déjà enregistrées dans le fichier
    - Pour obtenir ce résultat, il faut ouvrir le fichier de façon différente:

      import java.io.{FileWriter,PrintWriter}

      val pw = new PrintWriter(new FileWriter("fichier.txt",true))

      Si le fichier fichier.txt existe déjà, les données qui seront écrites dans le fichier ensuite seront ajoutées à la suite de celles qui y sont déjà enregistrées

- Fichiers
  - Lire les données depuis un fichier

https://alvinalexander.com/scala/how-to-open-read-text-files-in-scala-cookbook-examples

    - Import d'un package Scala
      - *import scala.io.Source*
    - Création d'un source d'entrée (fichier) pour la lecture
      - *val fr = Source.fromFile("fichier.txt")*
    - Déclaration d'un itérateur de lecture du fichier ligne par ligne
      - *val lignefr = fr.reset.getLines*
    - Test de fin de fichier
      - *lignefr.isEmpty*
      - La lecture peut se faire ligne par ligne jusqu'à la fin du fichier. On sait qu'on a atteint la fin du fichier quand *isEmpty* retourne la valeur *true*
    - Lecture d'une ligne
      - *var ligne = lignefr.next*
      - *ligne* est un String et son contenu correspond à celui de la ligne courante du fichier
      - L'itérateur passe à la ligne suivante
      - On va lire le fichier ligne par ligne jusqu'à atteindre la fin du fichier (isEmpty vaut true)
    - Fermeture du fichier

      fr.close

- Fichiers textes
  - Exemple: écriture et lecture d'un fichier: dans cet exemple, on déclare une classe Personne et on instancie 4 objets de cette classe qu'on enregistre dans un fichier texte en utilisant le format CSV
    - Deux méthodes sont définies dans la classe Personne: une méthode qui construit un String au format d'une ligne CSV à partir des attributs de la classe
    - Une méthode qui affiche les attributs à la console
  - Puis on lit le fichier ligne par ligne. Chaque ligne est récupérée sous forme de String. Ce String est splité en plusieurs sous-Strings pour récupérer les valeurs des attributs de chaque personne.
  - Chaque personne lue dans le fichier est instanciée et stockée dans un tableau.

- Fichiers textes
  - Exemple: écriture et lecture d'un fichier

PremierProgramme.scala

```scala
import io.StdIn;
import java.io.PrintWriter;
import scala.io.Source;

class Personne(val genre: Char, val prenom: String, val nom: String, val anneeNaissance: Int) {
  def saveCSV: String = {
    return genre + "," + prenom + "," + nom + "," + anneeNaissance
  };

  def affiche: Unit = {
    println("Genre: " + genre + ", Prénom: " + prenom +
      ",Nom : " + nom + ", Année de naissance : " + anneeNaissance)
  }
};
```

- Fichiers textes
  - Exemple: écriture et lecture d'un fichier

```
object PremierProgramme {
  def main(args: Array[String]): Unit = {
    // ## instanciation de 4 objets Personne ##
      val p1 = new Personne('M', "Jean-Jacques", "Rousseau", 1712);
      val p2 = new Personne('F', "Marie", "Curie", 1867);
      val p3 = new Personne('M', "Bob", "Marley", 1945);
      val p4 = new Personne('F', "Serena", "Williams", 1981);
    // ## enregistrement dans le fichier ##
      val fw = new PrintWriter("fichier.txt");
      fw.println(p1.saveCSV);
      fw.println(p2.saveCSV);
      fw.println(p3.saveCSV);
      fw.println(p4.saveCSV);
      fw.close;
```

- Fichiers textes
  - Exemple: écriture et lecture d'un fichier

```
    // ## lecture du fichier ##
      val toutlemonde = Array.fill[Personne](4)(null);
      val fr = Source.fromFile("fichier.txt") ;
      val lignefr = fr.reset.getLines;
      var i = 0;
    // ## boucle de lecture ligne par ligne (tant qu'on n'a pas atteint la fin du fichier)
      while (! lignefr.isEmpty) {
      // lecture d'une ligne (on récupère un String qui contient toute la ligne)
        var ligne = lignefr.next;
      // La ligne est splitée en utilisant la , comme caractère de séparation
        var unePersonne = ligne.split(",");
        toutlemonde(i) = new Personne(unePersonne(0).charAt(0),
                                      unePersonne(1),
                                      unePersonne(2),
                                      unePersonne(3).toInt);
        i += 1
      };
      fr.close;
    // ## affichage des personnes lues dans le fichier ##
      for (x <- toutlemonde) x.affiche
  }
};
```

- Fichiers textes
  - Exemple: écriture et lecture d'un fichier

Problems | Tasks | Console | Debug | Scala Interpreter (PremierProjet)

```
<terminated> PremierProgramme$ [Scala Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_66.jdk/Contents/Home/bin/java (10 août 2017 à 18
Genre: M, Prénom: Jean-Jacques,Nom : Rousseau, Année de naissance : 1712
Genre: F, Prénom: Marie,Nom : Curie, Année de naissance : 1867
Genre: M, Prénom: Bob,Nom : Marley, Année de naissance : 1945
Genre: F, Prénom: Serena,Nom : Williams, Année de naissance : 1981
```