Fiche

Utilisation de `org.json`

Cette fiche présente brièvement comment utiliser la bibliothèque `org.json` pour générer un document JSON, l'afficher à l'écran, le sauvegarder dans un fichier et le relire depuis ce fichier.

1 Classe `Personne`

Cette classe est utilisée pour illustrer la sérialisation en JSON. Nous avons besoin d'une classe possédant des *getters*.

```
public class Personne {

    private String nom;
    private String prenom;
    private int age;

    public Personne(String nom, String prenom, int age) {
        this.nom = nom;
        this.prenom = prenom;
        this.age = age;
    }
    public String getNom() {
        return nom;
    }
    public String getPrenom() {
        return prenom;
    }
    public int getAge() {
        return age;
    }
    public String toString() {
        return nom + "␣" + prenom + "␣(" + age + "an(s))";
    }
}
```

2 Génération d'un fichier JSON

Ce programme permet de créer un document JSON avec un tableau de personnes. Ce tableau est placé dans un objet, avec la propriété `contacts`. Pour le sauvegarder dans un fichier, on sauvegarde simplement la chaîne de caractères récupérée depuis l'objet `JSONObject`.

Dans un premier temps, nous créons notre tableau d'objets `Personne`.

```
Personne p[] = {
        new Personne("John", "Smith", 30),
        new Personne("Cyril", "Rabat", 25)
    };
```

Nous créons l'objet JSON et nous y ajoutons le tableau de `Personne` associé à la clé *contacts*. Le tableau sera automatiquement sérialisé par l'API de JSON.org en utilisant les *getters*.

```
JSONObject objet = new JSONObject();
try {
    objet.put("contacts", new JSONArray(p));
} catch(JSONException e) {
    System.err.println("Erreur lors de l'insertion du tableau.");
    System.err.println(e);
    System.exit(-1);
}
```

Nous ouvrons un flux vers le fichier de sortie (passé en argument).

```
FileWriter fs = null;
try {
  fs = new FileWriter(args[0]);
} catch(IOException e) {
  System.err.println("Erreur lors de l'ouverture du fichier '" + args[0]
      + "'.");
  System.err.println(e);
  System.exit(0);
}
```

La méthode `write` de l'objet JSON prend le flux vers le ficher, le nombre d'espaces pour les tabulations (ici 3) et le décalage (ici 0). Le JSON est sauvegardé dans le fichier.

```
try {
  objet.write(fs, 3, 0);
  fs.flush();
} catch(IOException e) {
  System.err.println("Erreur lors de l'écriture dans le fichier.");
  System.err.println(e);
  System.exit(0);
}
```

3 Lecture d'un fichier contenant du JSON

Le programme suivant permet d'ouvrir le fichier dont le nom est passé en paramètre. On lit le contenu du fichier, que l'on place dans une chaîne de caractères. On peut ensuite créer un objet `JSONObject` à partir de cette chaîne de caractères. Le programme parcourt ensuite les éléments et les affiche à l'écran.

Nous commençons par récupérer l'ensemble des caractères du fichier :

```
String json = "";
try {
  byte[] contenu = Files.readAllBytes(Paths.get(args[0]));
  json = new String(contenu);
} catch(IOException e) {
  System.err.println("Erreur␣lors␣de␣la␣lecture␣du␣fichier␣'" + args[0]
     + "'");
  System.exit(0);
}
```

L'objet JSON est créé directement à partir de la chaîne lue :

```
JSONObject objet = new JSONObject(json);
```

Pour afficher les informations sur les contacts, nous pouvons récupérer le tableau à l'aide de la méthode `getJSONArray`, puis le parcourir :

```
JSONArray tableau = objet.getJSONArray("contacts");
System.out.println("Liste␣des␣personnes␣:");
for(int i = 0; i < tableau.length(); i++) {
  JSONObject element = tableau.getJSONObject(i);
  System.out.print("nom=" + element.getString("nom"));
  System.out.print(",␣prenom=" + element.getString("prenom"));
  System.out.println(",␣age=" + element.getInt("age"));
}
```

4 Compilation et exécution

`org.json` est téléchargeable au format JAR (une archive *Java* spécifique). Si vous utilisez un éditeur (*Eclipse* ou *NetBeans*), il faut intégrer cette archive au projet. Le reste est automatique. Sinon, il faut spécifier le fichier JAR dans la ligne de commandes. Par exemple :

```
javac -cp "json.jar;." GenerateurJSON.java
java -cp "json.jar;." GenerateurJSON
```

Pour la compilation sous le terminal Linux, il faut utiliser : (deux points) au lieu de ; (point virgule) comme séparateur.

Les fichiers sources fournis contiennent des accents. Ajoutez l'option `-encoding "utf-8"` pour éviter les problèmes d'affichage dans la console.

Une autre solution consiste à renseigner la variable d'environnement `CLASSPATH` qui indique tous les répertoires contenant les bibliothèques *Java* utilisées.