I/O, HTTP et Web-Services

Programmation avancée – Java

F5 – ISIMA 2020/2021

Olivier Goutet
o.goutet@openium.fr

20 octobre 2020

Retours, TP3 – Un fichier par cas

```
public void testEmptyFile() {
        List<MetroStop> metroStopList = Parser.csvParser("testFiles/empty_file.csv");
        assertEquals(0, metroStopList.size());
}

        /**
         * Line reading test (1 column missing, 1 column adding, bad separator)
         */
public void testLineReading() {
        List<MetroStop> metroStopList = Parser.csvParser("testFiles/line_reading.csv");
        assertEquals(1, metroStopList.size());
}
```

Retours, TP3 – Tester ce qui a peu de chance de merder

```
public class MetroStopTest extends TestCase {
    MetroStop metro1 = new MetroStop(1649,2.36697471484243, 48.7870

    public void testGetId() {
        Integer x = 1649;
        assertEquals(x, metro1.getId());
    }

    public void testGetLongitude() {
        Double x = 2.36697471484243;
        assertEquals(x, metro1.getLongitude());
    }     ...
```

Retours, TP3 – Tester ce qui ne peut pas planter

```
public void testParse() {
                // Instanciation
        String[] chaine = {"1975","2.33871281165883", "48.884417645
        String[] parsed = mParser.parse("1975#2.33871281165883#48.8

        // Test du parser correct
        assertEquals(chaine[0], parsed[0] );
        assertEquals(chaine[1], parsed[1] );
        assertEquals(chaine[2], parsed[2] );
        assertEquals(chaine[3], parsed[3] );
        assertEquals(chaine[4], parsed[4] );
        assertEquals(chaine[5], parsed[5] );
}
```

Retours, TP3 – Chemin en dur

```
public void testGetBufferedReader1() {
  BufferedReader b;
  try {
    b = Parser.getBufferedReader("C:\\...\\ISI_3\\src\\ratp_arret.c
    } catch(Exception e) {
    b = null;
    System.out.println("Exception␣:␣" + e);
    }
  assertTrue(b instanceof BufferedReader);
}
```

Retours, TP3 – Test qui doit envoyer une exception

```
@Test
void testWrongLine(){
    String line = "19752.338712811658834 8.844176451841AbbessesPARIS
    try{
        Parser.createMetroStop(line);
        fail("La fonction n'a pas retourne de NumberFormatException
    } catch (NumberFormatException e){
        assert(true);
    }
}
```

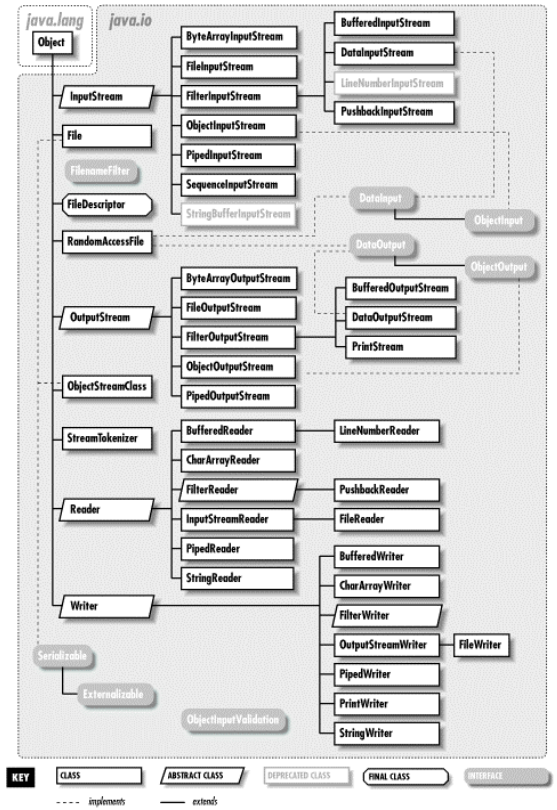
Plan

Plan

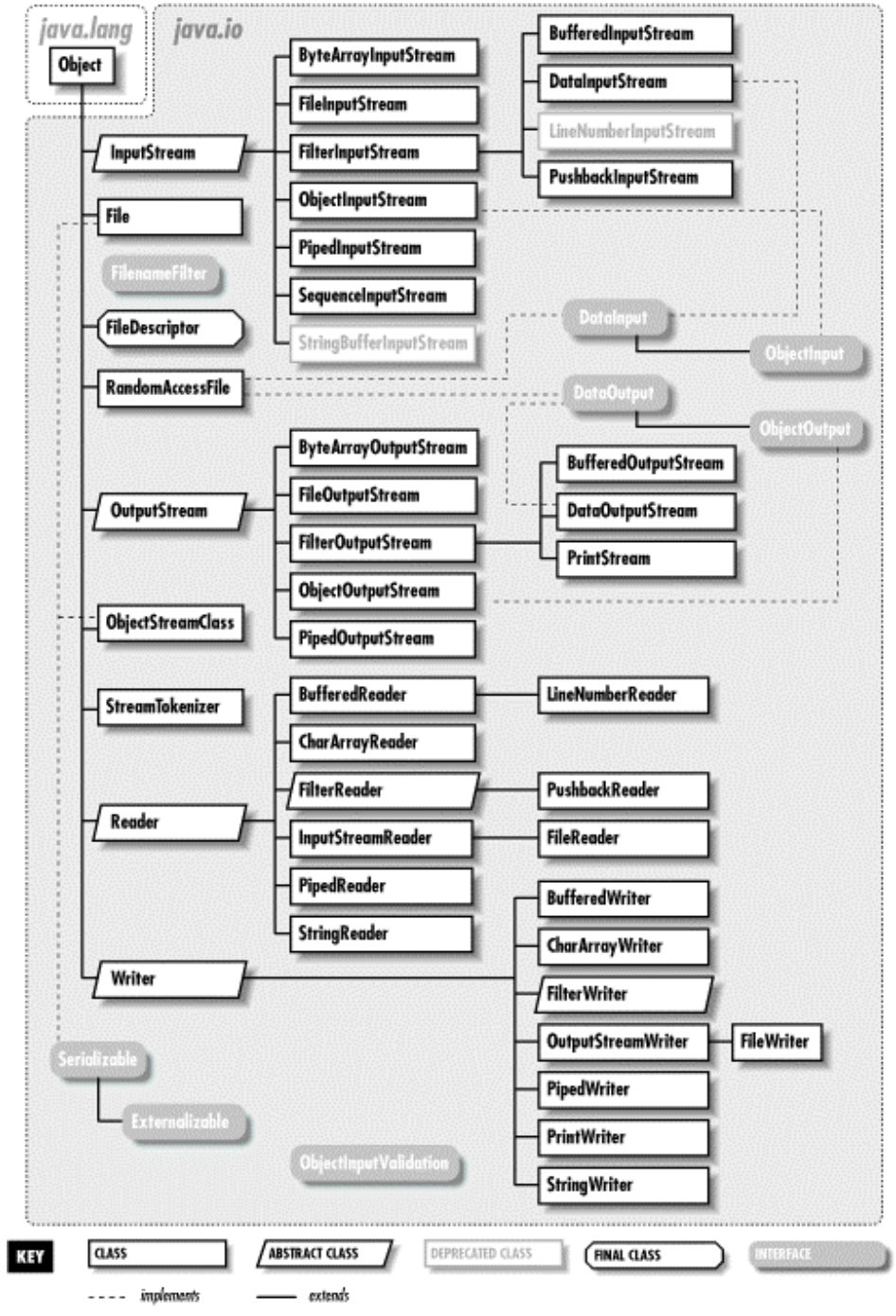
I/O

- Entrées et sorties
  - Fichier
  - Réseau
- Java fournit tout ce qu'il faut nativement
- Stream
  - Flot de données
  - Uni ou bi-directionnel
  - InputStream, OutputStream, Reader, ...

Package java.io

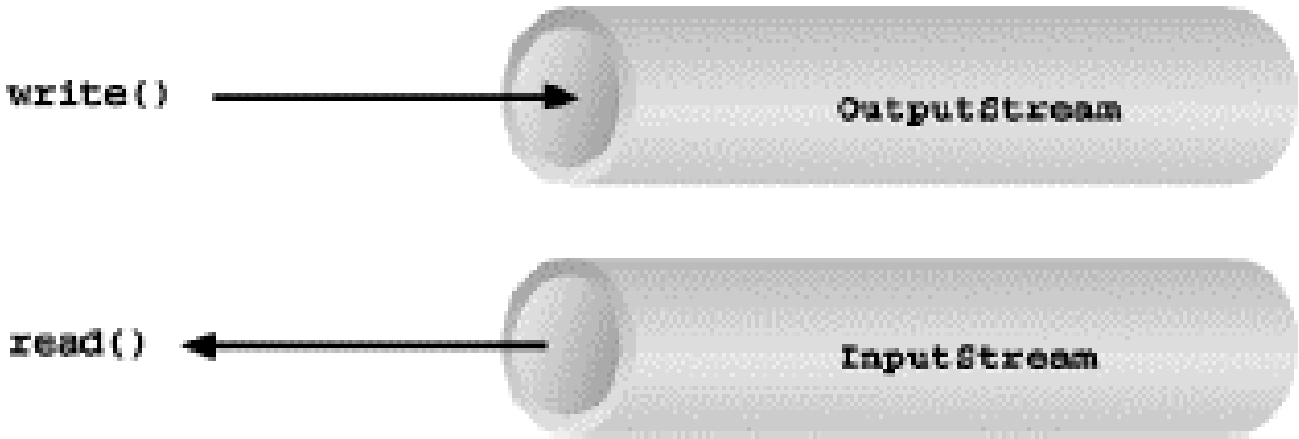


Learning Java 2nd edition – O'Reilly – Pat Neiemeyer & Jonathan Knudsen


java.lang
java.io
Object
InputStream
ByteArrayInputStream
FileInputStream
FilterInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream
StringBufferInputStream
BufferedInputStream
DataInputStream
LineNumberInputStream
PushbackInputStream
File
FilenameFilter
FileDescriptor
RandomAccessFile
DataInput
ObjectInput
DataOutput
ObjectOutput
OutputStream
ByteArrayOutputStream
FileOutputStream
FilterOutputStream
ObjectOutputStream
PipedOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
ObjectStreamClass
StreamTokenizer
Reader
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader
Writer
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter
Serializable
Externalizable
ObjectInputValidation
KEY
CLASS
ABSTRACT CLASS
DEPRECATED CLASS
FINAL CLASS
INTERFACE
implements
extends

Streams

- `InputStream` / `OutputStream`
  - Classes abstraites
  - Définissent les fonctions de base
  - Base de tous les streams



Learning Java 2nd edition – O'Reilly – Pat Neiemeyer & Jonathan Knudsen

Streams

- `Reader` / `Writer`
  - Classes abstraites
  - Dédiées aux séquences de caractères (support de unicode)
  - Base des streams de caractères
- `InputStreamReader` / `InputStreamWriter`
  - Pont entre les streams et les readers

Streams

- `DataInputStream` / `DataOutputStream`
  - Types de base (String, int...)
- `ObjectInputStream` / `ObjectOutputStream`
  - Objets sérialisés
- Buffered...
  - Streams et Readers
  - Ajout d'un système de tampon
  - Amélioration des performances
- `FileInputStream` / `FileOutputStream`
  `FileReader` / `FileWriter`
  - Lecture et écriture dans le système de fichier
- ...

Streams standards

- in / out / err

```
InputStream stdin = System.in;
OutputStream stdout = System.out;
OutputStream stderr = System.err;

int attente = System.in.available();
if ( attente > 0 ){
    byte [] donnees = new byte [ attente ];
    System.in.read ( donnees );
    ...
}
```

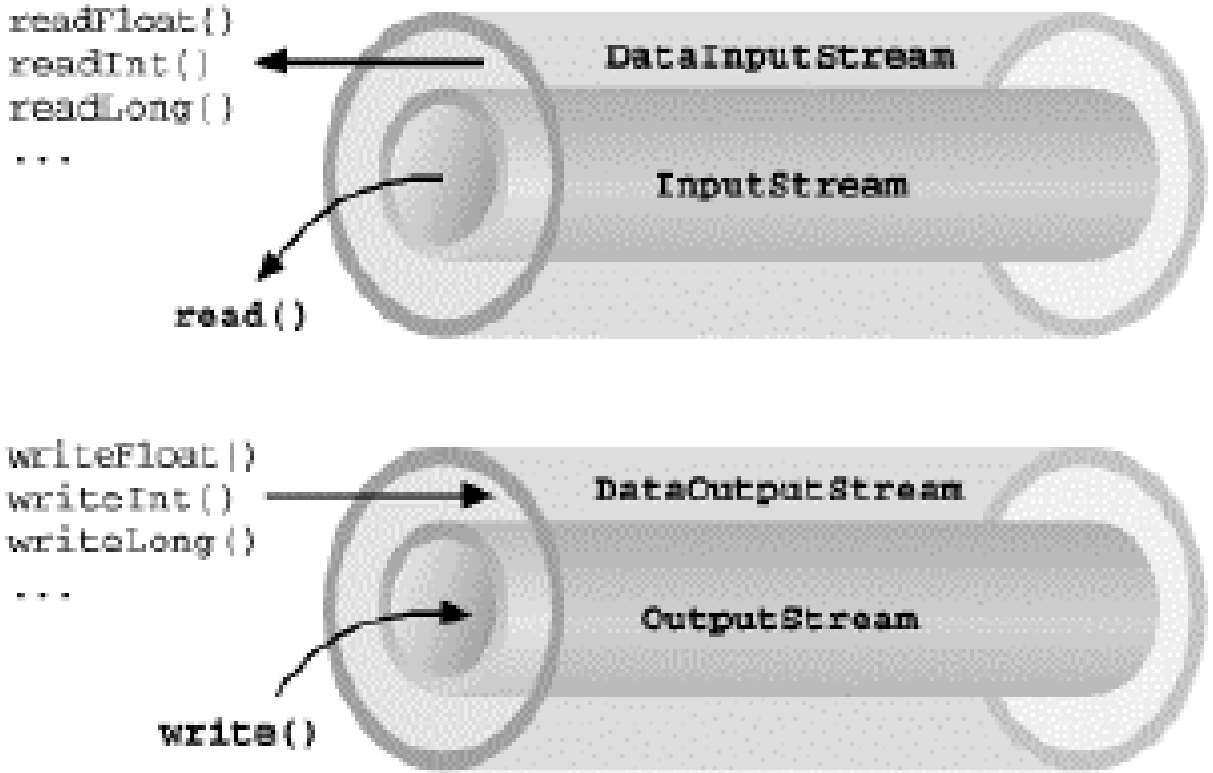
Readers

- `InputStream` / `OutputStream`
  - Byte = 1 octet
  - Unicode ?
- Readers
  - Gèrent l'unicode
  - `InputStreamReader` et `InputStreamWriter` pour faire les conversions avec les Stream
  - Fonctionne quelle que soit la langue

Readers – Exemple

```
try{
   InputStreamReader convertisseur=new InputStreamReader(System.in);
   BufferedReader in = new BufferedReader(convertisseur);
   String texte = in.readLine();
} catch( IOException e){
   System.out.println("IOException␣message:"+ e.getMessage())
   e.printStackTrace();
}
```

Enveloppes




```
DataInputStream dis = new DataInputStream(System.in);
double d = dis.readDouble();
```

Enveloppes

- Permet de spécialiser le flux selon ses besoins
- On peux en enchaîner plusieurs
  - `InputStream`
  - `BufferedInputStream`
  - `DataInputStream`

I/O bufferisés

- Buffered...
- Ajout d'un tampon entre les données et l'application
- Peut améliorer les performances en diminuant le nombre d'accès réels
- Utilisation des enveloppes

I/O bufferisés

```
FileInputStream fis = new FileInputStream("donnees.dat");
BufferedInputStream bis = new BufferedInputStream(fis,4096);
// On peut le re-encapsuler si besoin
bis.read();
```

- Au premier appel à read, chargement de 4096 octets (si possible)
- “Reremplissage” automatique si besoin
- Écriture sur le même principe
  - `flush()` pour vider le buffer
  - Ne pas l’oublier !

Pour aller plus loin

- PrintWriter
- StringReader
- ...
- Compression de données
  - `GZIPInputStream` / `GZIPOutputStream`
- NIO
  - E/S asynchrones
  - Performances améliorées
  - Fichiers mappés en mémoire
  - ...

Plan

HTTP

- HyperText Transfer Protocol
- Base d'Internet
- Port 80
- Non sécurisé
- Version sécurisée
  - HTTPS
  - Port 443
- State-less

Codes de retour HTTP

- 1XX information
  - 100 Continue
- 2XX Succès
  - 200 : ok, 204 : pas de contenu
- 3XX Redirection
  - 302 : déplacé temporairement
- 4XX Erreur
  - 400 : mauvaise requête, 404 : ressource non trouvée
- 5XX Erreur du serveur
  - 500 : Erreur interne, 503 : Service non disponible

Méthodes HTTP

- `GET`
- `POST`
- `PUT`
- `DELETE`
- `HEAD`
- `PATCH`
- `TRACE`
- `CONNECT`

Méthodes HTTP

- `GET` Récupération de contenu
- `POST` Envoi de contenu
- `PUT` Mise à jour de contenu
- `DELETE` Suppression de contenu
- `HEAD` Récupération du header sans le contenu
- `PATCH` Modification partielle du contenu
- `TRACE` Diagnostique
- `CONNECT` Utiliser un proxy comme tunnel de communication

Exemple GET HTTP

```
wget http://etudiants.openium.fr/javaf5/message.txt
```

```
GET /javaf5/message.txt HTTP/1.1
User-Agent: Wget/1.15 (darwin13.0.0)
Accept: */*
Host: etudiants.openium.fr
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 20 Oct 2020 08:00:52 GMT
Server: Apache/2.2.22 (Debian)
Last-Modified: Mon, 19 Oct 2019 20:15:01 GMT
ETag: "a8a8d5-7-505538dae7b40"
Accept-Ranges: bytes
Content-Length: 7
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/plain

coucou
```

Exemple GET HTTP Erreur

`wget http://etudiants.openium.fr/javaf5/toto.txt`

```
GET /javaf5/toto.txt HTTP/1.1
User-Agent: Wget/1.15 (darwin13.0.0)
Accept: */*
Host: etudiants.openium.fr
Connection: Keep-Alive

HTTP/1.1 404 Not Found
Date: Tue, 20 Oct 2020 08:00:52 GMT
Server: Apache/2.2.22 (Debian)
Vary: Accept-Encoding
Content-Length: 299
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<html><head><title>404 Not Found</title></head>
<body>
<h1>Not Found</h1>
<p>The requested URL /javaf5/toto.txt was not found on this server.
</body></html>
```

Exemple HEAD HTTP

```
curl -I http://etudiants.openium.fr/javaf5/message.txt
```

```
HEAD /javaf5/message.txt HTTP/1.1
User-Agent: curl/7.34.0
Host: etudiants.openium.fr
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 20 Oct 2020 08:00:52 GMT
Server: Apache/2.2.22 (Debian)
Last-Modified: Mon, 19 Oct 2020 20:15:01 GMT
ETag: "a8a8d5-7-505538dae7b40"
Accept-Ranges: bytes
Content-Length: 7
Vary: Accept-Encoding
Content-Type: text/plain
```

Exemple POST HTTP

```
wget --post-data="{"id":45,"nom":"olivier"} http://etudiants.
```

```
  POST /javaf5/message.txt HTTP/1.1
User-Agent: Wget/1.15 (darwin13.0.0)
Accept: */*
Host: etudiants.openium.fr
Connection: Keep-Alive
Content-Type: application/json
Content-Length: 11

{"id":45,"nom":"olivier"}

HTTP/1.1 201 CREATED
Date: Mon, 10 Oct 2020 20:25:04 GMT
Server: Apache/2.2.22 (Debian)
...
Connection: Keep-Alive
Content-Type: application/json
{"id":45}
```

Plan

Web-Services

- Méthodes de communication entres machines
- RESTfull WS
  - Utilisation de HTTP
  - Utilisation des Uri
  - Utilisation de langages de description : JSON, XML...

Web-Services - Exemple

- `http://exemple.com/etudiants`
  - `GET` : Récupération des étudiants
  - `PUT` : Modification de la liste d'étudiants
  - `POST` : Envoi d'une nouvelle liste d'étudiants
  - `DELETE` : Suppression de tous les étudiants
- `http://exemple.com/etudiants/17`
  - `GET` : Récupération de l'étudiant
  - `PUT` : Modification de l'étudiant et création si il n'existe pas
  - `POST` : A ne pas utiliser
  - `DELETE` : Suppression de l'étudiant
- Variante : `http://example.com/etudiants?id=17`
- Variante : `http://example.com/etudiants?id=17&nom=toto`

Appel HTTP en Java

- Plus complexe qu'une simple socket
- Utilisation de classes dédiées
  - Java HttpURLConnection
    - Simple d'utilisation
    - Intégré au langage
  - Apache HttpClient
    - Plus simple d'interface
    - Gère des pool de connexion

Exemple HttpURLConnection

```
HttpURLConnection urlConnection = null;
try {
    URL url = new URL("http://www.android.com/");
    urlConnection = (HttpURLConnection) url.openConnection();

    InputStream in = new BufferedInputStream(
                                urlConnection.getInputStream());
    readStream(in);
} finally {
    if (urlConnection != null){
        urlConnection.disconnect();
    }
}
```

HTTP GET sur www.android.com

Traitement des données

- On récupère un InputStream sur des données
- Comment les traiter ?
- Choix en fonction du format
  - Texte brut : `Reader`
  - Image : `javax.imageio.ImageIO`
  - XML : Bibliothèque de traitement XML
  - JSON : Bibliothèque de traitement JSON

Traitement des données - Json

- Utilisation d'une bibliothèque
  - Gson
  - Jackson
  - javax.json
- Simplifie le "parse" des flux JSON

Traitement des données avec Gson

```
{
    "id":45,
    "nom":"Goutet",
    "prenom":"Olivier"
}
```

```
public class Etudiant {
    private int id;
    private String nom;
    private String prenom;
}
```

```
Gson gson = new Gson();
Etudiant e = gson.fromJson(br, Etudiant.class);
```

Traitement des données avec Gson – Binding

```
@SerializedName("etudiant")
public class Etudiant {
    @SerializedName("id")
    private int mId;
    @SerializedName("nom")
    private String mNom;
    @SerializedName("prenom")
    private String mPrenom;
}
```

Java Pro Tips

- Item 65 : Dont' ignore exceptions

```
// Empty catch block ignores exception -- Highly suspect!
try {
   ...
} catch (Exception e){
}
```

- Item 2 : Consider a builder when faced with many constructors parameters

```
NutritionFacts cocaCola =
                new NutritionFacts(240, 8, 100, 0, 35, 27);
```

Java Pro Tips Item 2 (1)

```
// Telescoping constructor pattern - does not scale well!
public class NutritionFacts {
    private final int servingSize;  // (mL)             required
    private final int servings;     // (per container) required
    private final int calories;     //                 optional
    private final int fat;          // (g)             optional
    private final int sodium;       // (mg)            optional
    private final int carbohydrate; // (g)             optional

    public NutritionFacts(int servingSize, int servings) {
        this(servingSize, servings, 0);
    }

    public NutritionFacts(int servingSize, int servings,
            int calories) {
        this(servingSize, servings, calories, 0);
    }

    public NutritionFacts(int servingSize, int servings,
            int calories, int fat) {
        this(servingSize, servings, calories, fat, 0);
    }
```

Java Pro Tips Item 2 (2)

```
    public NutritionFacts(int servingSize, int servings,
            int calories, int fat, int sodium) {
        this(servingSize, servings, calories, fat, sodium, 0);
    }

    public NutritionFacts(int servingSize, int servings,
           int calories, int fat, int sodium, int carbohydrate) {
        this.servingSize  = servingSize;
        this.servings     = servings;
        this.calories     = calories;
        this.fat          = fat;
        this.sodium       = sodium;
        this.carbohydrate = carbohydrate;
    }
}
```

- Difficile à lire
- Long à écrire
- et si on a 15 paramètres ?

Java Pro Tips Item 2 – Bean version

```
// JavaBeans Pattern - allows inconsistency, mandates mutability
public class NutritionFacts {
    // Parameters initialized to default values (if any)
    private int servingSize  = -1; // Required; no default value
    private int servings     = -1;  //     "     "       "       "
    private int calories     = 0;
    private int fat          = 0;
    private int sodium       = 0;
    private int carbohydrate = 0;

    public NutritionFacts() { }
    // Setters
    public void setServingSize(int val)  { servingSize = val; }
    public void setServings(int val)     { servings = val; }
    public void setCalories(int val)     { calories = val; }
    public void setFat(int val)          { fat = val; }
    public void setSodium(int val)       { sodium = val; }
    public void setCarbohydrate(int val) { carbohydrate = val; }
}

NutritionFacts cocaCola = new NutritionFacts();
cocaCola.setServingSize(240);
cocaCola.setServings(8);
cocaCola.setCalories(100);
cocaCola.setSodium(35);
cocaCola.setCarbohydrate(27);
```

Java Pro Tips Item 2 – Builder Pattern (1)

```
public static class Builder {
  // Required parameters
  private final int servingSize;
  private final int servings;

  // Optional parameters - initialized to default values
  private int calories      = 0;
  private int fat           = 0;
  private int carbohydrate  = 0;
  private int sodium        = 0;

  public Builder(int servingSize, int servings) {
    this.servingSize = servingSize;
    this.servings    = servings;
  }

  public Builder calories(int val)
   { calories = val;      return this; }
  public Builder fat(int val)
   { fat = val;           return this; }
  public Builder carbohydrate(int val)
   { carbohydrate = val;  return this; }
  public Builder sodium(int val)
   { sodium = val;        return this; }

  public NutritionFacts build() {
    return new NutritionFacts(this);
  }
}
```

Java Pro Tips Item 2 – Builder Pattern (2)

```
// Builder Pattern
public class NutritionFacts {
    private final int servingSize;
    private final int servings;
    private final int calories;
    private final int fat;
    private final int sodium;
    private final int carbohydrate;

    public static class Builder {
       // ...
    }

    private NutritionFacts(Builder builder) {
        servingSize  = builder.servingSize;
        servings     = builder.servings;
        calories     = builder.calories;
        fat          = builder.fat;
        sodium       = builder.sodium;
        carbohydrate = builder.carbohydrate;
    }
}
```

Java Pro Tips Item 2 – Builder Pattern utilisation

```
NutritionFacts cocaCola = new NutritionFacts.Builder(240, 8).
  calories(100).sodium(35).carbohydrate(27).build();

// ou

NutritionFacts.Builder builder = new NutritionFacts.Builder(240,8);
builder.calories(100);
builder.sodium(35);
builder.calories(27);
NutritionFacts nutritionFacts = builder.build();
```