Qualité de développement

CM3-5 : Java « avancé »

Mickaël Martin Nevot

V2.0.1

Qualité de développement

Pile/tas

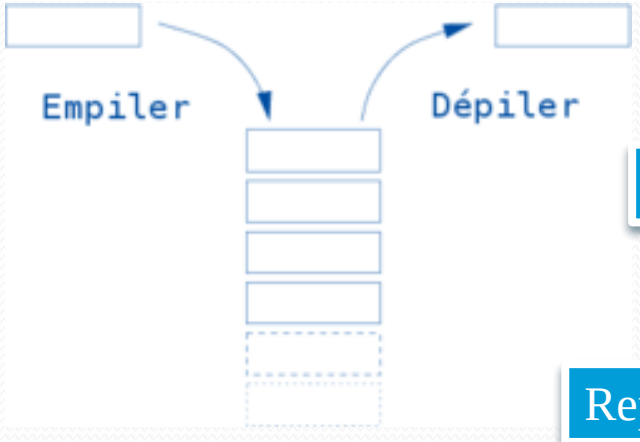
Fonctionnement dépendant de l'implémentation de la JVM : seul le principe général est expliqué ici !

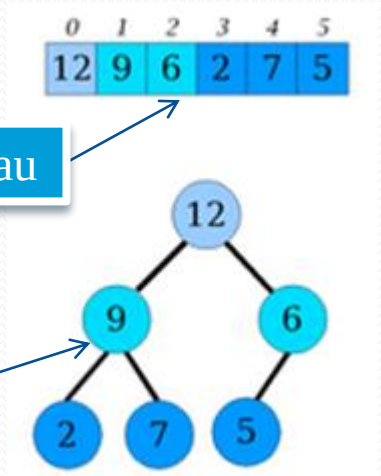
Pile (ou registre)

- Principe d'une pile :
  - Sommet : dernier élément
- Appels de méthodes
- Variables locales

Tas

- Objet :
  - Place allouée par `new`
  - Contient attributs/méthodes
  - $taille_{objet} = \sum taille_{attributs}$
  - *Garbage collector*





Représentation en tableau

Représentation en arbre

Destruction et garbage collector

- Destruction implicite en Java
- *Garbage collector* (ou ramasse-miettes):
  - Appel automatique :
    - Si plus aucune variable ne référence l'objet
    - Si le bloc dans lequel l'objet est défini se termine
    - Si l'objet a été affecté à `null`
  - Appel manuel :
    - Instruction : `System.gc();`
- Usage de **pseudo-destructeur** :
  - Classe utilisateur : `protected void finalize()`
  - Appelée juste avant le *garbage collector* ? Pas certain !

Pour être sûr que `finalize()` soit appelée, il faut appeler manuellement le *garbage collector*

Flux

- Flux de données (comme un film en « streaming » !)
- Paquetage `java.io` :
  - Flux binaires (lecture/écriture d'octets) :
    - `InputStream`, etc.
    - `OutputStream`, etc.
  - Flux de caractères (lecture/écriture de caractères) :
    - `Reader` (`BufferedReader`, `FileReader`, etc.)
    - `Writer` (`BufferedWriter`, `FileWriter`, etc.)

```
// Lit l'entrée standard.
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
// Lit la ligne jusqu'au prochain retour chariot.
String inputLine = br.readLine();
```

Sérialisation

- Interface `Serialisable`
- Mot clef `transient` (pas de sérialisation) :

```
public class MyClass implements Serialisable {
    …
    protected MyClass2 myObj1 = new MyClass2();
    // myObj2 ne sera pas sérialisé.
    transient MyClass3  myObj2 = new MyClass3();
}
```

- « Désérialisation » :

```
FileInputStream fis = new FileInputStream ("myFile.ser");
ObjectInputStream ois = new ObjectInputStream (fis);
Object first = ois.readObject ();
MyClass myObj = (MyClass) first;
ois.close();
…
```

L'extension du fichier n'est pas obligatoire et n'a aucune importance

Collections (Java 2)

- Désavantages d'un tableau :
  - Taille statique
  - Recherche lente (exhaustive)
  - Pas de *pattern* de déplacement dans les éléments
- API Java :
  - Collection (interface `Collection`) :
    - Généricité et références (n'importe quelles références objets)
    - Opérations optimisées et communes
    - Itérateurs (parcourent les éléments un à un sans problème de type)
  - Tableau dynamique : `ArrayList`
  - Liste : `LinkedList`
  - Ensemble : `HashSet`, `TreeSet`

Itérateurs

- Monodirectionnel : interface `Iterator`
  - Toutes les collections en ont un :

```
// C'est une collection : on récupère son itérateur.
Iterator iter = c.iterator();
while (iter.hasNext()) {
    MyClass o = iter.next();
}
```

- Bidirectionnel : interface `ListIterator` (dérive de `Iterator`)
  - Listes et tableaux dynamiques uniquement
  - Deux sens

```
ListIterator iter = c.listIterator();
while (iter.hasPrevious()) {
    MyClass o = iter.previous();
}
```

Exemples de collections

- `LinkedList` (liste doublement chaînée)
- `ArrayList` (à la place de `Vector` qui est déprécié) :
  - Encapsulation du tableau avec une taille dynamique

```
ArrayList arrList <String>= new ArrayList<String>();
arrList.add("toto"); // Valide.
arrList.add(new String ("tata")); // Valide.
arrList.add(10); // Non valide ...
```

- `HashSet` :
  - Permet des éléments identiques
  - Prévoit la redéfinition des méthodes :
    - `hashCode()` : ordonnancer les éléments
    - `equals(...)`

Ellipse : varargs (Java 5)

- Nombre indéfini de valeurs de même type en paramètre
- Traitée comme un tableau
- Deux manières :
  - Avec un tableau (éventuellement vide)
  - Avec un ensemble de paramètres
- Placée en dernier dans la liste des paramètres
- En cas de surcharge de méthode, la méthode contenant l'ellipse a la **priorité la plus faible**

```
public meth1(Type... tab) { … }
…
int[] t = {1, 2, 3, 4, 5};
meth1(t); // Envoyé comme un tableau.
meth1(1, 2, 3, 4, 5); // Envoyé comme un ensemble de paramètres.
```

Utilisation de l'ellipse : …

JAR/WAR

- Formats de fichier
- JAR (extension `.jar`) :
  - Outil d'archivage du *bytecode* et des **métadonnées**
    - Fichier *manifest* : `MANIFEST.MF`

```
Manifest-Version: 1.0
Created-By: 1.4.1_01 (Sun Microsystems Inc.)
Main-class: HelloWorld
```

Fichier `MANIFEST.MF`

Classe principale à exécuter

  - On peut lire/utiliser le contenu d'un JAR
- WAR (extension `.war`) :
  - Assemblage de JAR pour une application Web
  - Utilisé pour un déploiement sur un serveur d'application

À savoir

- Interface `Cloneable`, permet de disposer de la méthode :

```
protected Object clone() { … }
```

- `enum` :

```
public enum Animal {KANGAROO, TIGER, DOG, SNAKE, CAT, … };
```

- `instanceOf` : ← A utiliser avec parcimonie

```
if (myObj instanceOf MyClass) {
    myObj2 = (MyClass) myObj; // Downcasting.
}
```

Bonnes pratiques

- Traitez toutes les exceptions susceptibles d'être lancées
- Faites attention à ne pas créer de *deadlock*
- Attention à l'héritage d'un générique

Aller plus loin

- Mot clef `volatile`
- Métaprogrammation par annotation
- Synchronisation de haut niveau (API de concurrence)
- API de management
- Gestion de flux standards : classe `Scanner`
- Site Web avec JHipster

Liens

- Documents électroniques :
  - http://hdd34.developpez.com/cours/artpoo/
  - http://fabrique-jeu-video.blogspot.fr/2013/08/poo.html
  - http://nicolas.baudru.perso.luminy.univ-amu.fr/#PO
- Document classique :
  - Livres :
    - Claude Delannoy. *Programmer en Java 2ème édition.*
  - Cours :
    - Hugues Fauconnier. *Programmation orientée objet en Java.*
    - Guillaume Revy. *Introduction à la Programmation Orientée Objet... et son application au C++.*
    - Craig Larman. *UML2 et les design patterns.*
    - Mathieu Nebra. *La programmation orientée objet.*

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com