Bases de la POO / Java

Les évolutions de Java

Java 5 (Tiger) J2SE 5.0 - 2004

- ➔Autoboxing / AutoUnboxing
- ➔Itération : Nouvelle boucle for
- ➔Enumérations (Enum)
- ➔Nombre d'arguments variable
- ➔Instruction printf (C)
- ➔Import statique
- ➔Classe Scanner
- ➔Programmation générique (Generic)
- ➔Annotations (Meta données / @)

➔http://lroux.developpez.com/article/java/tiger/

Autoboxing/AutoUnboxing

➔ Transformation automatique et implicite d'un type primitif en un objet wrapper à la compilation

➔Exemple

```
ArrayList<Integer> list = new ArrayList<Integer>();
int a = 100;
list.add(a);
int b = list.get(0);
```

Itération : nouvelle boucle for

➔ for (variable : collection)

- collection =
  - Soit un tableau classique
  - Soit un objet qui implémente l'interface «Iterable»

➔Exemple

```
String[] noms= {"Marie", "Jean", "Paul"};
for (String s : noms)
      System.out.println(s);
```

Enumération

Solution avant Java 5

```
public class Couleurs {
public static final int jaune=0;
public static final int vert=1;
public static final int rouge=2;
- - -
}
```

Accès à un élément par la syntaxe : `Couleurs.jaune`

Inconvénients de la syntaxe précédente

1) Pas "type safe"
   - `int val = Couleurs.bleu;`
2) Pas de correspondance valeur et description
3) Pas d'adaptation pour l'instruction switch
4) Paradigme objets pas respecté (une entité sans comportements !)

Enumération de type safe

Description d'une liste de constantes par une instruction spécifique (**enum**) qui supprime les inconvénients précédents

Pas de conversion possible avec le type int.

Exemple :

```
public enum Couleurs{jaune, vert, rouge};
```

Enumération plus complexe


```
public enum Numbers
{
ONE("première occurrence ", 1),
TWO("deuxième occurrence ", 2),
THREE("troisième occurrence ", 3);

Numbers(String d, int v){description = d;value = v;}
private String description;
private int value;
public String getDescription() { return description; }
public int getValue() { return value; }
}
```

Utilisation : **`Numbers num = Numbers.ONE;`**

Signature formelle extensible

La signature formelle d'une méthode peut spécifier un nombre variable de paramètres.

Usage de la meta description **...** (**ellipse**)

Exemple :

public float moyenne (int coef, int ... val);

Exemple


```
public class Exemple {
  public static float moyenne (int ... notes) {
         float resultat= 0.0f;
         for (int i=0; i < notes.length ; i++)
               resultat += notes[i];
         return resultat / notes.length;
  }

  public static void main(String[] args) {
    System.out.println ("M1= " + moyenne (12, 8, 13) );
    System.out.println ("M2= " + moyenne (12, 8, 13, 10, 17));
  }
}
```

Instruction **printf**

Nouvelle méthode de la classe **java.io.PrintStream**

Permet au programmeur de maîtriser la mise en forme du flux de sortie

Exemple :

```
double x= 1./3;
System.out.printf(" %5.2f ", x);
```

Voir aussi méthode de classe ***String.format*** et la classe **java.util.Formatter**.

Importation statique

Possibilité d'éviter de préfixer systématiquement les appels des méthodes de classe par l'identificateur de la classe support.

Exemple :

```
import static java.lang.Math.PI ;
import static java.lang.Math.sin();
// ou
import static java.lang.Math.* ;
PI ;   // fait référence à Math.PI
sin(x) ; // fait référence à Math.sin()
cos(x) ; // fait référence à Math.cos()
```

Classe **java.util.Scanner**

➔A simple text scanner which can parse primitive types and strings using regular expressions

- Analyseur lexicographique (tokens parser)

➔Voir aussi classe **java.util.StringTokenizer**

➔Délimiteur par défaut : \p{javaWhiteSpace}+

- http://docs.oracle.com/javase/1.5.0/docs/api/java/util/regex/Pattern.html

➔Un des constructeurs disponibles exploite toute instance valide de la classe **java.io.File**

Exemple


```
String input = "1 test 2 test red test bluetest ";
Scanner s = new
Scanner(input).useDelimiter("\\s*test\\s*");
System.out.println(s.nextInt());
System.out.println(s.nextInt());
System.out.println(s.next());
System.out.println(s.next());
s.close();
```

```
1
2
red
blue
Process completed.
```

Java 6 (Mustang) J2SE 6.0 - 2006

➔XML et Web services

➔AWT – Swing

➔Nouvelles collections (Deque, NavigableSet)

➔E/S et réseaux

➔http://adiguba.developpez.com/tutoriels/java/6/

Java 7 (Dolphin) J2SE 7.0 - 2011

➔Notation binaire, caractère _ sur grands nombres, switch sur String, notation <>

➔**E/S**

- **java.io.File** remplacé par **java.nio.file.Path**
- classe utilitaire **Files**
- **java.nio.channels** pour les E/S asynchrones

➔…

➔http://adiguba.developpez.com/tutoriels/java/7/#Lo/