

IFT 6800

Atelier en Technologies d'information

Cours 7 : Le langage de programmation Java
chapitre 2 : Structure du langage Java



1

Premier programme: "Factorielle.java"

```
/* Mon premier programme Java */  
public class Factorielle {  
    public static double factorielle(int x) {  
        if (x < 0)  
            return 0.0;  
        double fact = 1.0;  
        while (x > 1) {  
            fact = fact * x;  
            x = x - 1;  
        }  
        return fact;  
    }  
    public static void main(String[] args) {  
        int entree = Integer.parseInt(args[0]);  
        double resultat = factorielle(entree);  
        System.out.println(resultat);  
    }  
}
```

2

Un premier programme (2)

- Pour compiler le programme précédent, il faut taper:
`phobos% javac Factorielle.java`
- La compilation d'un fichier ".java" produit un fichier ".class" dans le même répertoire.
- Pour exécuter le programme précédent, il faut taper:
`phobos% java Factorielle 4`
24.0

3

Structure d'un fichier Java

- Chaque fichier Java doit contenir la définition d'une classe, portant le même nom que celui du fichier (attention aux minuscules/majuscules).
- C'est pourquoi on ne distinguera pas la classe du fichier qui contient sa définition.
- Pour pouvoir être exécutée de cette façon, une classe doit contenir une méthode statique appelée "main".

4

Type de données

- Java offre les types élémentaires suivants:
 - `boolean` true/false
 - `char` Caractère Unicode (sur 16 bits)
 - `byte` Octet signé
 - `short` Entier 16-bits signé
 - `int` Entier 32-bits signé
 - `long` Entier 64-bits signé
 - `float` Nombre à point flottant (sur 32-bits)
 - `double` Nombre à point flottant (sur 64 bits)
- De plus, Java offre le type `String` pour manipuler les chaînes de caractères.
 - Donc, ne pas utiliser un tableau de caractères...

5

Type de données (2)

- Tableaux d'éléments de même type

```
byte[] tableauByte;
int[] fibonacci = { 1, 1, 2, 3, 5, 8, 13, 21 };
String[] listeAdresses = new String[4];
```
- La taille d'un tableau se définit soit par le nombre d'éléments qui l'initialisent, soit à l'allocation.
- L'accès aux éléments d'un tableau se fait comment en C.

```
x = x + fibonacci[3];
listeAdresses[i] = "1, place du Café, JavaTown"
```
- Le nombre d'éléments d'un tableau s'obtient par l'attribut `length` (en lecture seulement).

```
for (i = 0; i < tableauByte.length; i++)
```

6

Opérateurs

- La plupart ressemblent à ceux qu'on retrouve en C:
 - Arithmétiques: * / % + - ++ --
 - Logiques: ! || &&
 - Bits: ~ | & ^ << >>
 - Relationnels: < <= == != >= >
 - Conditionnels: ?:
 - Assignment: = *= /= %= += -= <<= >>= &= ^= |=
- Quelques nouveaux venus:
 - Concaténation de chaînes: +
 - Décalage de bits vers la droite, sans extension de signe: >>>
 - ...et son correspondant avec assignation: >>>=
 - Les opérateurs & et | peuvent aussi s'appliquer sur des booléens. Dans ce cas, les 2 opérandes seront toujours évaluées.

7

Énoncés

- Les énoncés Java ressemblent aussi beaucoup à ceux qu'on retrouve en C:
 - Expression
 - Étiquette
 - Déclaration de variables
 - Conditions: if, select/case
 - Boucles: while, do, for
 - Altérations de l'exécution: break, continue, return
- Quelques particularités
 - Les déclarations peuvent se faire n'importe où dans une méthode.
 - Les énoncés break et continue peuvent accepter une étiquette comme paramètre, précisant la boucle qu'ils affectent.

8

1. Structure du langage

9

Structure du langage Les types primitifs

- **boolean(true/false), byte (1 octet), char (2 octets), short (2 octets), int (4 octets), long (8 octets), float (4 octets), double (8 octets).**
- **Les variables peuvent être déclarées n'importe où dans un bloc.**
- **Les affectations non implicites doivent être *castées* (sinon erreur à la compilation).**

```
int i = 258;
long l = i; // ok
byte b = i; // error: Explicit cast needed to convert
            int to byte
byte b = 258; // error: Explicit cast needed to convert
            int to byte
byte b = (byte)i; // ok mais b = 2
```

10

Structure du langage

Les structures de contrôle et expressions

- **Essentiellement les mêmes qu'en C**

- `if, switch, for, while, do while`

- `++, +=, &&, &, <<, ?:`

- **Plus les blocs labélisés**

```
UN: while(...) {
  DEUX: for(...) {
    TROIS: while(...) {
      f (...) continue UN; // Reprend sur la première boucle while
      if (...) break DEUX; // Quitte la boucle for
      continue; // Reprend sur la deuxième boucle while
    }
  }
}
```

11

Structure du langage

Les tableaux

- **Déclaration**

```
int[] array_of_int; // équivalent à : int array_of_int[];
Color rgb_cube[][][];
```

- **Création et initialisation**

```
array_of_int = new int[42];
rgb_cube = new Color[256][256][256];
int[] primes = {1, 2, 3, 5, 7, 7+4};
array_of_int[0] = 3
```

- **Utilisation**

```
int l = array_of_int.length; // l = 42
int e = array_of_int[50]; // Lève une ArrayIndexOutOfBoundsException
```

12

Exceptions(1)

- Lorsqu'une méthode Java rencontre une situation qu'elle ne peut traiter, elle *lance* une exception grâce à l'énoncé `throw`.

```
if (index < 0) throw new  
    ArrayOutOfBoundsException();
```

- Une exception est un objet, une instance d'une sous-classe de `java.lang.Exception`.
- Une méthode en amont peut *attraper* l'exception, permettant une gestion des erreurs plus souple qu'en C, grâce aux énoncés `try` et `catch`.

13

Exceptions(2)

- **Exemple:**

```
try {  
    ...code susceptible de générer une exception  
}  
catch (ArrayOutOfBoundsException e)  
{  
    ...actions à faire pour cette exception  
}  
catch (IOException e)  
{  
    ...regroupe toutes les exceptions d'entrée/sortie  
}  
catch (Exception e)  
{  
    ...toutes les autres exceptions  
}
```

14

Structure du langage

Les exceptions (3)

- Elles permettent de séparer un bloc d'instructions de la gestion des erreurs pouvant survenir dans ce bloc.

```
try {
    // Code pouvant lever des IOException ou des
    // SecurityException
}
catch (IOException e) {
    // Gestion des IOException et des sous-classes de
    // IOException
}
catch (Exception e){
    // Gestion de toutes les autres exceptions
```

15

Structure du langage

Les exceptions (4)

- Ce sont des instances de classes dérivant de `java.lang.Exception`
- La levée d'une exception provoque une remontée dans l'appel des méthodes jusqu'à ce qu'un bloc `catch` acceptant cette exception soit trouvé. Si aucun bloc `catch` n'est trouvé, l'exception est capturée par l'interpréteur et le programme s'arrête.
- L'appel à une méthode pouvant lever une exception doit :
 - soit être contenu dans un bloc `try/catch`
 - soit être situé dans une méthode propageant (`throws`) cette classe d'exception
- Un bloc (optionnel) `finally` peut-être posé à la suite des `catch`. Son contenu est exécuté après un `catch` ou après un `break`, un `continue` ou un `return` dans le bloc `try`

16

Structure du langage

Les unités de compilation

- Le code source d'une classe est appelé *unité de compilation*.
- Il est recommandé (mais pas imposé) de ne mettre qu'une classe par unité de compilation.
- L'unité de compilation (le fichier) doit avoir le même nom que la classe qu'elle contient.

17

Structure du langage

Les packages (1)

- Un package regroupe un ensemble de classes sous un même espace de 'nomage'.
- Les noms des packages suivent le schéma : `name.subname`
...
- Une classe `Watch` appartenant au package `time.clock` doit se trouver dans le fichier `time/clock/Watch.class`
- Les packages permettent au compilateur et à la JVM de localiser les fichiers contenant les classes à charger.
- L'instruction `package` indique à quel package appartient la ou les classe(s) de l'unité de compilation (le fichier).

18

Structure du langage Les packages (2)

- Les répertoires contenant les packages doivent être présents dans la variable d'environnement `CLASSPATH`
- En dehors du package, les noms des classes sont : `packageName.className`
- L'instruction `import packageName` permet d'utiliser des classes sans les préfixer par leur nom de package.
- Les API sont organisées en package (`java.lang`, `java.io`, ...)

19

Structure du langage Les packages (3)

```
CLASSPATH = $JAVA_HOME/lib/classes.zip;$HOME/classes
```

```
~dedieu/classes/graph/2D/Circle.java  
package graph.2D;  
public class Circle()  
{ ... }
```

```
~dedieu/classes/graph/3D/Sphere.java  
package graph.3D;  
public class Sphere()  
{ ... }
```

```
~dedieu/classes/paintShop/MainClass.java  
package paintShop;  
import graph.2D.*;  
public class MainClass()  
{  
    public static void main(String[] args) {  
        graph.2D.Circle c1 = new graph.2D.Circle(50)  
        Circle c2 = new Circle(70);  
        graph.3D.Sphere s1 = new graph.3D.Sphere(100);  
        Sphere s2 = new Sphere(40); // error: class paintShop.Sphere not found  
    }  
}
```

API standard

- Les classes de l'API standard se regroupent en "packages". Parmi les plus utilisés, on retrouve:
 - `java.applet` Création et communication avec une applet
 - `java.awt.*` Interface graphique
 - `java.io` Entrée/sortie
 - `java.lang.*` Éléments du langage
 - `java.math` Arithmétique sur grands entiers
 - `java.net` Télécommunications via socket
 - `java.security.*` Gestion de clés et de certificat
 - `java.text` Formats de message
 - `java.util.*` Utilitaires
 - `javax.swing.*` Nouvelle interface graphique

21

API standard (2)

- Pour utiliser une classe de l'API à l'intérieur d'une de nos classes, on a 3 possibilités:
 - On peut utiliser le nom complet de la classe.

```
java.io.FileInputStream fis;
```
 - On peut importer la classe. Tous les symboles exportés par la classe deviennent accessibles sans avoir à les préfixer par le nom du "package".

```
import java.net.Socket;
```
 - On peut importer d'un seul coup toutes les classes d'un "package".

```
import java.math.*;
```
- Le "package" `java.lang` est importé par défaut.
- Si 2 classes de 2 "packages" différents portent le même nom, il faudra quand même utiliser leur nom complet pour les différencier.

22

Références

- Thinking in Java. Eckel.
<http://www.EckelObject.com/Eckel>
- <http://www.javasoft.com> : *Site officiel Java (JDK et doc.)*
- <http://www.javasoft.com> : *Site officiel Java (JDK et doc.)*
- <http://www.javaworld.com> : *Info sur Java*
- <http://www.gamelan.com> : *applications, applets, packages, ...*
- <http://www.jars.com> : *idem*
- <http://www.blackdown.com> : *Java pour linux*