

CCI - Initiation à Java

E.Coquery

`emmanuel.coquery@liris.cnrs.fr`

Outline

- 1 Présentation du langage
- 2 Types de données primitifs
- 3 Objets

Un langage orienté objet

- La notion d'objets et de classes centrale en Java
 - Un fichier Java correspond normalement à une classe.
 - Une application Java est constituée d'un ensemble de classes.
 - Lancer une application Java consiste à appeler une méthode particulière de la classe correspondant au programme à exécuter.

Une syntaxe proche du C

- Déclarations de variables similaire au C :

```
type nom;      ou      type nom = val;
```

- Structures de contrôle du C :

- if (*condition*) { ... } else { ... }

- switch (*expr*) {
 case *cas1* : ... ; break ;
 case *cas2* : ... ; break ;

- ...

- default : ...

- }

- while (*condition*) do { ... }

- for (*init*; *test*; *modif*) { ... }

- Commentaires C/C++ :

- // *commentaire jusqu'à la fin de la ligne*

- /* *commentaire pouvant tenir sur plusieurs lignes* */

Des application portables

“Write Once, Run Everywhere”

- Compilation d'une application Java :
 - Un fichier `.class` par classe de l'application.
 - Contient du *bytecode* indépendant de la plateforme.
- Exécution d'une application :
 - Java Virtual Machine (JVM)
 - ⇒ Exécution sur toute plateforme possédant une JVM.
 - Performance : recompilation à la volée du bytecode vers du code natif (Hotspot).

Gestion de la mémoire automatisée

- Pas de pointeurs explicites en Java
- La notion de référence reste présente
- Les JVM intègrent un ramasse-miettes (*Garbage Collector*)

Libère automatiquement la mémoire occupée par les structures de données qui ne sont plus utilisées.

Un langage répandu

- JVM sur Windows, Mac OS, Linux et d'autres
- Bibliothèque standard Java très fournie :
 - Structures de données
 - Entrées sorties
 - Réseau
 - Bases de données, XML
 - Interfaces graphiques
- Nombreuses autres bibliothèques disponibles
- IDEs évoluées : Netbeans, Eclipse, JBuilder, ...
- Pages Web dynamiques et serveurs d'applications en Java

Outline

- 1 Présentation du langage
- 2 Types de données primitifs
- 3 Objets

Types de base

En plus des objets, Java inclus des types de base :

- Types numériques (signés) :
 - `byte` (8 bits), `short` (16 bits), `int` (32 bits), `long` (64 bits)
 - `float` (32 bits), `double` (64 bits)
- Caractères : type `char`
 - codé sur 16 bits en utilisant le standard unicode.
 - Peuvent être manipulés comme des entiers non signés
- Booléens : type `boolean` (`true` et `false`)
 - Les booléens *ne sont pas* des entiers en Java

L'implémentation des types est standard

- ne dépend pas de la JVM (ni de la plateforme)
⇒ meilleure portabilité

Tableaux

- Déclaration : `type [] nom;`
 - `int [] monTableau;`
- Création : `new type [taille]`
 - `monTableau = new int [10];`
- Création avec valeurs : `{val1, val2, ..., valn}`
ou `new type [] {val1, val2, ..., valn}`
 - `int [] autreTableau = {3,5,1,3,8};`
- Accès : `nom[index]`
 - `int unNombre = monTableau[3] + 1;`
 - `monTableau[7] = unNombre * 5;`

Une erreur survient si on accède hors des cases du tableau

Tableaux (2)

- Taille : `nom.length`
 - `int tailleDuTableau = monTableau.length;`
- Tableaux multidimensionnels (tableaux de tableaux) :
 - `int [][] tableauDeuxDimensions;`
 - `tableauDeuxDimensions = new int [10][20];`
 - `tableauDeuxDimensions = new int [10] [];`
 - `tableauDeuxDimensions[0] = new int[5];`

Tableaux et références

Une variable tableau est une référence vers le tableau :

- ```
int [] tab1 = {0,1,2,3};
int [] tab2 = new int [tab1.length];
for (int i = 0; i < tab1.length; i++)
 { tab2[i] = tab1[i]; }
tab1[2] = 9;
```

tab2[2] vaut toujours 2.
- ```
int [] tab1 = {0,1,2,3};  
int [] tab2 = tab1;  
tab1[2] = 9;
```

tab2[2] vaut 9.

Outline

- 1 Présentation du langage
- 2 Types de données primitifs
- 3 Objets**

Qu'est qu'une classe Java ?

- Pas un objet, plutôt un type :
on peut le voir comme un *struct C* amélioré.
- Mais surtout un schéma pour créer des objets.
- Dans une classe, on défini :
 - Des variables d'instance (ou champs, ou attributs)
`type nom ;`
éventuellement avec une valeur par défaut :
`type nom = val ;`
ex : `double x = 3.0 ;`
 - Des méthodes : fonctions qui sont liées à un objet.
 - Des méthodes particulières servant à l'initialisation des objets créés : les *constructeurs*

Exemple de classe

```
public class Point {  
  
    double x; // attribut  
    double y; // attribut  
  
    public Point() { // constructeur  
        x = 0.0; // initialisation de x  
        y = 0.0; // initialisation de y  
    }  
  
    public Point(double abscisse, double ordonnee) { //constructeur  
        x = abscisse;  
        y = ordonnee;  
    }  
  
    public double getDistanceOrigine() { // methode  
        double sommeCarres = x*x+y*y;  
        return Math.sqrt(sommeCarres);  
    }  
  
    public void afficheDistance() { // methode  
        System.out.println(getDistanceOrigine());  
    }  
  
}
```

Qu'est qu'une instance d'une classe ?

- C'est un objet créé en utilisant cette classe comme schéma de construction.
- Tout objet est une instance d'une certaine classe.
- Cet objet possède tous les champs et méthodes définis dans cette classe.

Utilisation des objets

- Variable contenant un objet : `Classe nom ;`
`Point monPoint ;`
Comme pour les tableaux, la variable contient une *référence* vers l'objet.
- Création d'un objet : `new Classe(arg1, ..., argn)`
`monPoint = new Point(2.0, 3.0) ;`
Appel au constructeur Point à deux arguments
- Accès à un champ : `objet.champ`
`monPoint.x = monPoint.y * 2 ;`
- Utilisation d'une méthode : `objet.methode(arg1, ..., argn)`
`monPoint.afficheDistance() ;`

Le mot clef `static`

- Normalement les méthodes et les champs sont liés à des objets.
- L'utilisation de `static` permet d'attacher un champ ou une méthode à une `classe`.
- L'utilisation est similaire au champ et méthodes non statiques :
on remplace simplement l'objet par la classe : `Classe.champ`
`System.out`
`Classe.methode(arg1, ..., argn)`
`Math.sqrt(4.0) ;`
- Attention : une méthode statique ne peut pas utiliser directement une méthode non statique.

Exemple

```
public class Bonjour {  
  
    int nombreFois;  
  
    public Bonjour(int nbFois) {  
        nombreFois = nbFois;  
    }  
  
    public void ditBonjour() {  
        for(int i = 0; i < nombreFois; i++) {  
            System.out.println("Bonjour");  
        }  
    }  
  
    public static void main(String [] args) {  
        Bonjour bj = new Bonjour(3);  
        bj.ditBonjour();  
    }  
  
}
```

Les *packages*

- Groupes de classes
- Basée sur une organisation de répertoires
 - un paquet correspond à un sous-répertoire
 - dans le nom du paquet, les / sont remplacés par des .
 - `java.lang`, `java.io`, `javax.swing`, `javax.xml`,
`org.xml.sax`, etc
- Lorsqu'une classe est dans un paquet, son nom complet est de la forme :
nom.du.paquet.LaClasse
ex : `java.lang.Math`

Les *packages* (2)

- Lorsqu'une classe est dans un paquet, le fichier dans lequel elle est définie doit débiter par :

```
package nom.du.paquet ;
```

- Il n'est pas nécessaire d'utiliser le nom complet des classes qui sont dans le même paquet que la classe que l'on est en train de définir.
- Pour éviter de répéter le nom du paquet d'une classe, on peut utiliser les déclarations :

- pour une classe :

```
import nom.du.paquet.LaClasse ;
```

- pour toutes les classes du paquet d'un coup :

```
import nom.du.paquet.* ;
```

Autres modificateurs de déclaration

Il existe des modificateurs pour la visibilité

- *public* : visible partout
- *protected* : visible dans le paquet courant et dans les classes qui héritent de la classe courante
- quand on ne dit rien : visible dans le paquet courant
- *private* : visible uniquement dans la classe courante

modificateur *final* :

- interdit les changements pour les variables et les champs (combiné avec `static`, on obtient des constantes)
- interdit la surcharge pour les méthodes

Une classe particulière : String

- Chaînes de caractères en Java
 - Ce ne sont pas des tableaux de caractères
 - Non modifiables
 - Valeurs entre " ".
- ```
String coucou = "bonjour" ;
```
- Concaténation : opérateur +
  - Nombreuses méthodes :
    - *char charAt(int)*
    - *boolean equalsTo(Object)*
    - *boolean startsWith(String)*
    - *boolean endsWith(String)*
    - *String substring(int, int)*

et bien d'autres ...

# Les StringBufferers

- Chaînes de caractères modifiables
  - y compris au niveau de leur taille
- Utilisées pour générer des chaînes de caractères
- Exemple :

```
StringBuffer sb = new StringBuffer();
for (int i = 0; i < 10; i++) {
 sb.append("Bonjour\n");
}
String dixFoisBonjour = sb.toString();
```