

Introduction à Java

Matthieu Herrb
CNRS-LAAS

<http://homepages.laas.fr/matthieu/cours/java/java.pdf>

Mars 2012

Histoire et motivations

Langage développé par James Gosling chez Sun Microsystems à partir de 1990

Pour des environnements embarqués :

- appareils spécifiques ("Appliances")
- applications Web

Contraintes :

- indépendant du matériel
- sécurité

La machine virtuelle Java

Le langage Java utilise une *machine virtuelle* :

- le compilateur produit un **bytecode**
- ce code est indépendant de l'architecture matérielle sur laquelle il est exécuté
- la **machine virtuelle** interprète ce bytecode et le transforme en code machine natif à la volée (*Just in time compilation*)

4/26

Machines virtuelles Java

HotSpot (Oracle JRE/JDK)

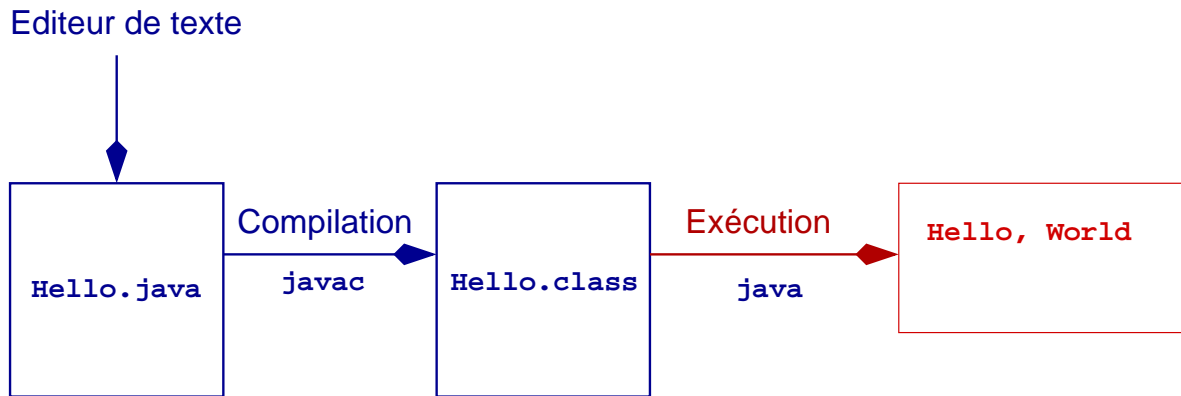
Dalvik (Google/Android)

Jazelle extension du jeu d'instruction de certains processeurs ARMv5 pour exécution directe de byte-code java.

ThumbEE généralisation de Jazelle sur processeurs ARMv7. Pas spécifique de Java (supporte aussi C#, Perl, Python)

5/26

Cycle de développement



6/26

Premier programme java

```
/**  
 ** Premier programme java  
 **/  
public class Hello {  
    public static void main(String argv[]) {  
        System.out.println("Hello, World!");  
    }  
}
```

7/26

Classes et objets

- **Classe** définit un type de données, et des fonctions (*méthodes*) qui le manipulent.
- **Objet** une *instance* de classe : une variable du type définit par la classe et ses méthodes.
- **Héritage** permet de créer des classes qui reprennent les données de la classe parente.
- **Interface** définition des prototypes des méthodes d'une classe sans préciser leur implémentation.

8/26

Gestion de la mémoire

Java simplifie la gestion de la mémoire

- pas de pointeurs : les références aux objets sont implicites, copies automatiques lorsque nécessaire.
- pas besoin de détruire ou libérer explicitement les objets : ceux qui ne sont plus utilisés sont recyclés par le processus **ramasse-miettes**

9/26

Variables

Quatre catégories de variables :

variables d'instance : données d'un objet

variables de classe : variable globale d'une classe,
existe en un seul exemplaire pour tous les objets de la
classe.

déclarée dans une classe avec `static`.

`final` déclare une variable de classe constante.

variables locales : déclarées dans le corps d'une méthode,
locales à la méthode qui les déclare.

paramètres : pseudo-variables utilisées pour décrire les paramètres d'une
méthode

11/26

Types de données de base

`byte` 1 octet ($-128.. + 127$)

`short` entiers sur 16 bits ($-32768.. + 32767$)

`int` entiers sur 32 bits

`long` entiers sur 64 bits

`float` nombres réels sur 32 bits

`double` nombre réels sur 64 bits

`boolean` variable binaire (`true` ou `false`)

`char` caractère Unicode

Il existe aussi des classes pour les types de base (**Number** et **String**).

12/26

Tableaux

Collections d'objets dont la taille est fixée à la création.

```
int[] anArray;  
anArray = new int[10];  
int[] anotherArray = {100, 200, 300, 400};  
anArray[0] = anotherArray[0];
```

13/26

Opérateurs

opérateur	précédence
postfix	<i>expr++ expr-</i>
unaire	<i>++expr -expr +expr -expr ~ !</i>
multiplicatif	<i>* / %</i>
additif	<i>+ -</i>
décalages	<i><< >> >>></i>
relations	<i>< > <= >= instanceof</i>
égalités	<i>== !=</i>
et bit à bit	<i>&</i>
ou exclusif bit à bit	<i>^</i>
ou inclusif bit à bit	<i> </i>
et logique	<i>&&</i>
ou logique	<i> </i>
ternaire	<i>? :</i>
affectation	<i>= += -= *= /= %= &= ^= = <<= >>= >>>=</i>

14/26

Instructions

- `if (expr) { ... }`
- `if (expr) { ... } else { ... }`
- `switch case`
- `while (expr) { ... }`
- `do { ... } while (expr)`
- `for (initialisation; terminaison; incrément) { ... }`
- `break`

15/26

Déclaration de classes

```
class MaClasse {  
    // variables, constructeurs, méthodes...  
}
```

Classe dérivée :

```
class MaClasse extends MaSuperClasse {  
    // variables, constructeurs, méthodes...  
}
```

Implémentation d'une interface :

```
class MaClasse implements MonInterface {  
    // variables, constructeurs, méthodes...  
}
```

17/26

Constructeurs

Le constructeur porte le nom de la classe, ne retourne rien.

Appelé par l'opérateur `new`.

Exemple :

```
public class Cube {
    float dimension;
    String couleur;

    public Cube(float d, String c) {
        dimension = d;
        couleur = c;
    }
}
..
Cube c = new Cube(1.0, "rouge");
```

18/26

Objets

- Créer les objets avec l'opérateur `new`
- Appel des méthodes de l'objet avec `.`

```
public class Cube {
    float dimension;
    ...

    public float volume() {
        return this.d*this.d*this.d;
    }
}

Cube c = new Cube(0.75, "vert");
System.out.println("le volume de c est "
    + c.volume + "m3");
```

19/26

Exceptions

Méthode de traitement des erreurs.

```
try {  
    code  
} catch (ExceptionType1 nom1) {  
    .... traitement exception 1  
} catch (ExceptionType2 nom2) {  
    .... traitement exception 2  
}
```

En Java il est **interdit** d'ignorer une exception !

20/26

Génération d'une exception

- Opérateur `throw`
- Classe `Exception`

Exemple :

```
try {  
    ... code  
    if (erreur)  
        throw new MyException("bug de logique");  
    ... code  
} catch (MyNewException e) {  
    writeln("exception: " + e);  
}
```

21/26

Synchronisation

Moniteurs

- mot clé `synchronized` dans la déclaration d'une méthode.
- condition anonyme : `wait()`
- signal sur une condition : `notify()` ou `notifyAll()`.

```
public synchronized void P(Semaphore s) {
    while (! s.valeur == 0) {
        try {
            wait();
        } catch (InterruptedException e) {}
    }
    s.valeur--;
}
```

22/26

Exemple: philosophes et spaghettis

```
public synchronized void arriver(int i) {
    philo[i].etat = DEMANDEE;
    while (! accessible(i)) {
        try {
            wait();
        } catch (InterruptedException e) {}
    }
    philo[i].etat = OCCUPEE;
}
public synchronized void quitter(int i) {
    philo[i].etat = LIBRE;
    notifyAll();
}
```

23/26

Packages

Mécanisme pour grouper un ensemble de classes et les rendre accessibles aux applications.

Un environnement Java (JRE) fourni un ensemble de packages standard.

Exemples :

<code>System</code>	fonctions système, en particulier entrées/sorties
<code>java.lang.Math</code>	fonctions mathématiques
<code>java.util.Random</code>	nombre aléatoires
<code>javax.swing</code>	boite à outils pour applications graphiques

Utilisation : `import java.util.Random;`

Questions ?