NFA031 : Programmation avec Java : notions de base

CM1-1 : Java

Mickaël Martin Nevot

V1.2.1

Programmation avec Java : notions de base

Java

- Sun Microsystem (1995)
- Langage :
  - **Orienté objet** et **fortement typé**
  - **Héritage simple**, interface, polymorphisme
- JRE :
  - JVM : **machine virtuelle** qui interprète le code
  - API : bibliothèques standards
- JDK :
  - Compilateur
  - JVM : débogueur

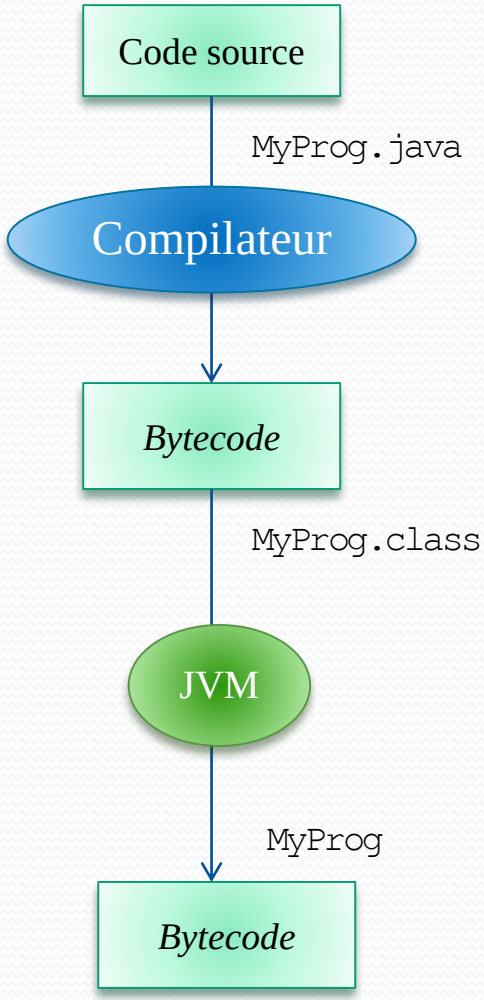
Philosophie

- **Simple** et familier
- **Robuste** et sûr
- **Indépendant** de la machine employée pour l'exécution
- Très **performant**
- Interprété, multitâches et dynamique
- Pourquoi apprendre Java ?
  - Plus de 4,5 milliards de périphériques
  - Programmes Web et services Web
  - Programme sur téléphone portable

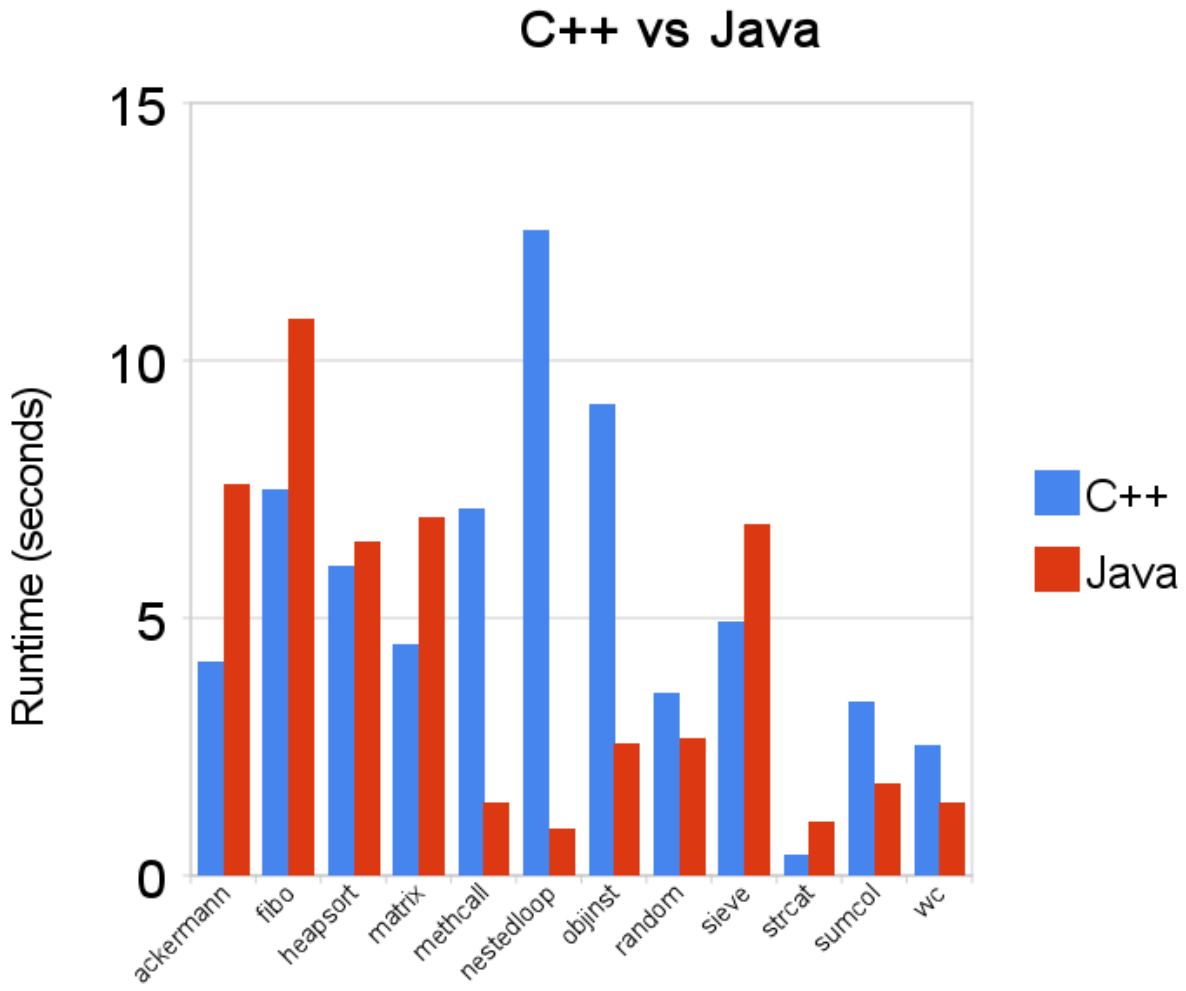
Duke, la mascotte de Java

Fonctionnement

- Création du code source
- **Compilation** en *bytecode* :
  - À partir du code source
  - Code exécutable sur toute JVM
- Exécution :
  - Interprète le *bytecode*
  - *Bytecode* indépendant de la plateforme



Différences par rapport à C++



Structure du code source

- Un fichier source Java contient une **classe**
- Une classe contient des **attributs** et des **méthodes**
- Une méthode contient des **instructions**

```
// Déclaration de classe.
class MyClass {
    // Déclaration d'attribut.
    int att1;

    //Déclaration de méthode.
    void meth1(int i) {
        instruction1
        instruction2
        …
    }
}
```

Structure du code source

- Méthode :

```
typeDeRetour myMeth(paramètre(s))
```

Signature

L'ordre des paramètres est déterminant

Délimitation de la classe

```
// Déclaration de classe.
class MyClass {
    int att1; // Déclaration d'attribut.

    float meth1(int i) { //Déclaration de méthode.
        instruction1
        instruction2
        …
    }
}
```

Paramètre (typé)

Délimitation de la méthode

Valeur de retour (`void` : aucune)

Utilisation

- Fichier source : extension `.java`
- Fichier binaire : extension `.class`
- **Un** fichier source contient **une** classe
- Le nom du fichier est identique au nom de la classe
- On lance l'exécution par la classe principale :
  - Contient un point de commencement d'exécution du code
- Sensible à la casse : `MyClass` est différent de `Myclass`
- Respecter les mots-clefs réservés

Instruction

- Se termine par `;`
- Type d'instruction :
  - Déclaration : `String att1;`
  - Affectation : `a = 10;`
  - Appel de fonction/méthode : `myMeth();`
  - Instruction conditionnelle : `if (a == b) { … }`
  - Instruction vide : `;`
  - Bloc (d'instructions) :

```
{
    instruction1
    …
}
```

Mise en forme / commentaires

- Mise en forme :
  - Indentation à chaque niveau de bloc
  - Convention de nommage :
    - `mypackage`
    - `MyClass`
    - `myMethod`
    - `myVar`
    - `MY_CONST`
- Commentaires de type C/C++ :

```
// Commentaire (une seule ligne)
/* Autre commentaire (une ligne). */
/*
   Autre commentaire (sur plusieurs lignes).
*/
```

Structure de contrôle

- Conditionnelle :
  - `if (condition) { … } else { … }`
- Branchement conditionnel :
  - `switch (ident) { case val0 : … case val1 : … default: … }`
- Boucles :
  - `for (initialisation ; condition ; modification) { … }`
  - `for (Type var : Collection) { … }`
  - `while (condition) { … }`
  - `do { … } while (condition)`
- Mots clefs `break`/`continue` :
  - `break` : permet de sortir du bloc (boucle, branchement conditionnel, etc.)
  - `continue` : « saute » à l'itération suivante d'une boucle

Portée et variable locale

- **Portée** (d'une variable) :
  - Début : à partir de sa déclaration
  - Fin : la fin du bloc d'instructions dans lequel elle se trouve (ou celui du corps de la méthode pour un paramètre)
- **Variable locale** :
  - Déclarée dans une méthode ou un bloc d'une méthode
  - Durée de vie : sa portée
  - **Visible qu'à l'intérieur du bloc**
  - Pas de valeur par défaut

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

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