Chapitre 1: Systèmes ordinés

Présentation pour

Java Software Solutions

Foundations of Program Design

Deuxième Edition

par John Lewis et William Loftus

Java Software Solutions est publié par Addison-Wesley

Focus du cours

- **Développement orienté-objet de logiciel**
  - **Résolution de problèmes**
  - **Implantation et design de programmes**
  - **Concepts orientés-objet**
    - **objets**
    - **classes**
    - **interfaces**
    - **héritage**
    - **polymorphisme**
  - **Le langage de programmation Java**

Hardware et Software

- **Hardware**
  - **Les parties physiques d'un ordinateur**
  - **clavier, moniteur, cables, chips**
- **Software**
  - **programmes et données**
  - **un *programme* est une série d'instructions**
- **Un ordinateur a besoin des deux: hardware et software**
- **Chacune est à proprement parler inutile sans l'autre**

Catégories de software

- **Système d'exploitation**
  - **Contrôle toutes les activités de l'ordinateur**
  - **Fournit l'interface entre l'usager et l'ordinateur**
  - **Gère les ressources telles le CPU et la mémoire**
  - **Ex: Windows 98, Windows NT, Unix, Linux, Mac OS**
- **Programme d'application**
  - **Nom générique pour tout autre type de software**
  - **Ex: traitement de texte, compilateur/debugger, jeux**
- **La plupart des systèmes d'exploitation et des programmes d'application ont un interface usager graphique (GUI)**

Résolution de problème

- **Écrire un programme a pour but de résoudre un problème**

- **Les étapes générales pour résoudre un problème sont :**

  - **Comprendre le problème**
  - **Décomposer le problème en morceaux traitables**
  - **Faire le design d'une solution**
  - **Considérer les alternatives à la solution et la raffiner**
  - **Implanter la solution**
  - **Tester la solution et régler tout problème qui survient**

Résolution de problème

- **Plusieurs projets de software échouent parce que le développeur n'a pas réellement compris le problème à résoudre**
- **On doit éviter les hypothèses et clarifier les ambiguïtés**

- **Lorsque les problèmes et leurs solutions grossissent, on doit organiser le développement en morceaux traitables**
- **Cette approche est fondamentale au développement de software**
- **Dans une approche *orientée-objet*, on sépare les solutions en morceaux appelés classes et objets**

Le langage de programmation Java

- **Un *langage de programmation* spécifie les mots et symboles qu'on utilise pour écrire un programme**

- **Un langage de programmation utilise un ensemble de règles qui spécifie comment les mots et les symboles peuvent être mis ensemble pour former des *énoncés de programme* valides**

- **Java a été créé par Sun Microsystems, Inc.**
- **Il a été introduit en 1995 et est devenu très populaire**
- **Java est un langage orienté-objet**

Structure d'un programme Java

- **Dans le langage de programmation Java :**
  - **Un programme est composé d'une ou de plusieurs *classes***
  - **Une classe contient une ou plusieurs *méthodes***
  - **Une méthode contient des *énoncés* (instructions) de programme**
- **Ces termes seront explorés en détail dans le cours**
- **Une application Java contient toujours une méthode appelée `main`**
- **Voir `Lincoln.java` (page 26)**

Structure d'un programme Java

```
//  commentaires sur la classe
public class MyProgram
{



}
```

Entête de la classe

Corps de la classe

Des commentaires peuvent être ajoutés presque n'importe où

Structure d'un programme Java

```
//  commentaires sur la classe
public class MyProgram
{
    //  commentaires sur la méthode
    public static void main (String[] args)
    {
    }
}
```

Corps de la méthode

Entête de la méthode

Commentaires

- **Les commentaires dans un programme sont aussi appelés *inline documentation***
- **Ils devraient être inclus pour expliquer le but du programme et décrire les étapes du traitement**
- **Ils ne modifient pas comment un programme s'exécute**
- **Les commentaires en Java peuvent prendre deux formes:**

```
// ce commentaire se rend jusqu'à la fin de la ligne

/*  ce commentaire continue jusqu'au symbole de
    terminaison, même sur plusieurs lignes      */
```

Identificateurs

- Les *identificateurs* sont des mots qu'un programmeur utilise dans un programme
- Un identificateur peut être constitué de lettres, de chiffres, le caractère souligné (_), et le signe de dollar ($)
- Un identificateur ne peut pas commencer par un chiffre
- Java distingue entre les lettres majuscules et les minuscules, et ainsi `Total` et `total` sont des identificateurs différents

Identificateurs

- On choisit parfois nous-mêmes des identificateurs en écrivant un programme (tel `Lincoln`)
- On utilise parfois le code d'autres programmeurs, et alors on utilise les identificateurs qu'ils ont choisis (tel `println`)
- On utilise souvent des identificateurs spéciaux appelés des mots réservés qui ont une signification prédéfinie dans le langage
- Un mot réservé ne peut pas être utilisé pour signifier autre chose

Mots réservés

- **Les mots réservés en Java sont :**

abstract	default	goto	operator	synchronized
boolean	do	if	outer	this
break	double	implements	package	throw
byte	else	import	private	throws
byvalue	extends	inner	protected	transient
case	false	instanceof	public	true
cast	final	int	rest	try
catch	finally	interface	return	var
char	float	long	short	void
class	for	native	static	volatile
const	future	new	super	while
continue	generic	null	switch	

Espace blanc

- Les espaces, les lignes blanches et les tabulations sont appelés les *espaces blancs*
- Un espace blanc est utilisé pour séparer les mots et les symboles dans un programme
- Tout espace blanc en extra est ignoré
- Un programme Java valide peut être formatté de plusieurs façons différentes
- Les programmes devraient être formattés pour améliorer la lecture, en utilisant une indentation uniforme
- Voir `Lincoln2.java` et `Lincoln3.java`

Niveau d'un langage de programmation

- **Il y a quatre niveaux pour un langage de programmation :**
  - **Langage machine**
  - **Langage assembleur**
  - **Langage de haut niveau**
  - **Langage de quatrième génération**
- **Chaque type de CPU a son *langage machine* spécifique**
- **Les autres niveaux ont été créés pour faciliter à un humain l'écriture de programmes**

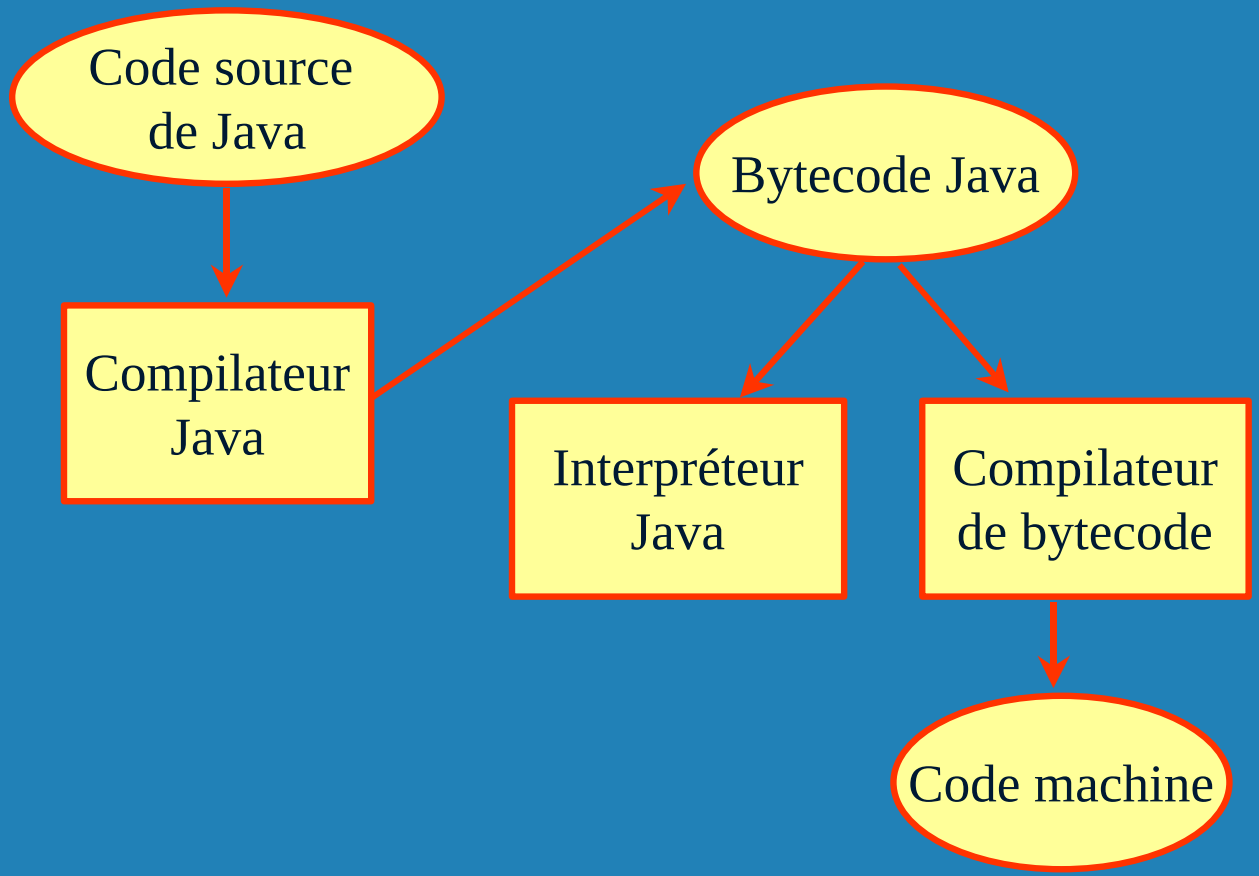
Langages de programmation

- **Un programme (code source) doit être traduit en langage machine avant d'être exécuté sur un type particulier de CPU**
- **Un *compilateur* est un outil software qui traduit le *code source* dans un langage cible spécifique**
- **Ce langage cible est souvent le langage machine pour un type particulier de CPU**
- **L'approche Java est un peu différente**

Traduction et exécution de Java

- **Le compilateur Java traduit le code source Java en une représentation spéciale appelée *bytecode***
- **Le bytecode Java n'est pas le langage machine pour un CPU traditionnel**
- **Un autre outil software, appelé un *interpréteur*, traduit le bytecode en langage machine et l'exécute**
- **Ainsi le compilateur Java n'est pas lié à une machine en particulier**
- **Java est considéré comme *architecture-neutral***

Traduction et exécution de Java



Environnements de développement

- **Il y a plusieurs environnements de développement pour écrire du software en Java :**
  - **Sun Java Software Development Kit (SDK)**
  - **Borland JBuilder**
  - **MetroWork CodeWarrior**
  - **Microsoft Visual J++**
  - **Symantec Café**
- **Quoique les détails de ces environnements diffèrent, le processus de base de compilation et d'exécution est essentiellement le même**

Syntaxe et sémantique

- **Les *règles de syntaxe* d'un langage définissent comment on peut mettre ensemble des symboles, des mots réservés et des identificateurs pour rendre un programme valide**
- **La *sémantique* d'un énoncé d'un programme définit ce que cet énoncé signifie (son but ou son rôle dans un programme)**
- **Un programme qui est syntaxiquement correct n'est pas nécessairement logiquement (sémantiquement) correct**
- **Un programme fera toujours ce qu'on lui dit de faire, pas nécessairement ce qu'on voudrait lui dire de faire**

Erreurs

- **Un programme peut avoir trois types d'erreurs**
- ***Erreurs de compilation* : le compilateur trouve les problèmes avec la syntaxe et les autres problèmes de base**
  - **Ex: variable non déclarée, assignation de types différents**
  - **S'il y a des erreurs de compilation, aucune version exécutable du programme n'est créée**
- ***Erreurs d'exécution* : un problème peut se produire lors de l'exécution du programme, comme en essayant de diviser par zéro, ce qui force le programme à se terminer anormalement**
- ***Erreurs logiques* : un programme peut s'exécuter, mais produire des résultats incorrects**