Cours JAVA : Le bases du langage Java.

Version 3.02

Julien Sopena[1]

[1]julien.sopena@lip6.fr
Équipe REGAL - INRIA Rocquencourt
LIP6 - Université Pierre et Marie Curie

Licence professionnelle DANT - 2015/2016

Grandes lignes du cours

`Java` en quelques mots

Comparatif `Java` et `C++`

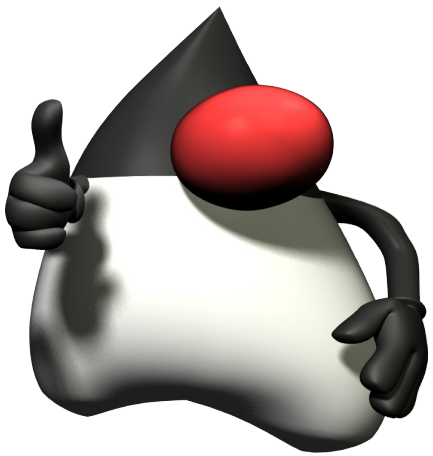
Programmation orientée objets.

- Conception par traitements.
- Conception par objets.

Outline

Java c'est quoi ?

- Un langage : Orienté objet fortement typé avec classes
- Un environnement d'exécution (JRE) : Une machine virtuelle et un ensemble de bibliothèques
- Un environnement de développement (JDK) : Un compilateur et un ensemble d'outils
- Une mascotte : Duke



Java c'est qui ?

La plate-forme et le langage Java sont issus d'un projet de *Sun Microsystems* datant de 1990.
Généralement, on attribut sa paternité a trois de ses ingénieurs :

- James Gosling
- Patrick Naughton
- Mike Sheridan

Figure – 1990 Barbecue chez James Gosling

Java pourquoi ?

Java est devenu aujourd'hui l'un des langages de programmation les plus utilisés.

Il est incontournable dans plusieurs domaines :

- **Systèmes dynamiques :** Chargement dynamique de classes
- **Internet :** Les *Applets* java
- **Systèmes communicants :** *RMI*, *Corba*, *EJB*, etc.

Java pour qui ?

Pour tous : Le 13 novembre 2006, Sun annonce le passage de Java, c'est-à-dire le JDK (JRE et outils de développement) sous **licence GPL**.

Pour vous : Cette UE sur Java servira de base à l'ensemble des UE techniques du deuxième semestre.

L'environnement actuel Java 2 Standard Edition

J2SE

L'outil de base : le JDK (Java Development Kit) de SUN :

- `http://java.sun.com`.
- gratuit.
- Dernière version : 1.6.
- comprend de nombreux outils :
  - le compilateur.
  - le compilateur à la volé "JIT".
  - le débogueur.
  - le générateur de documentation.

Des environnements de développements gratuits

- NetBeans : `http://www.netbeans.org/`
- Eclipse : `http://www.eclipse.org/`

Java évolue tout le temps

Java n'est pas un langage normalisé et il continu d'évoluer. Cette évolution se fait en ajoutant de **nouvelle API**, mais aussi en **modifiant la machine virtuelle**.

L'ensemble de ces modifications est géré par le **JCP (Java Community Process ; http ://www.jcp.org/)** dans lequel Sun continue de tenir une place prépondérante.

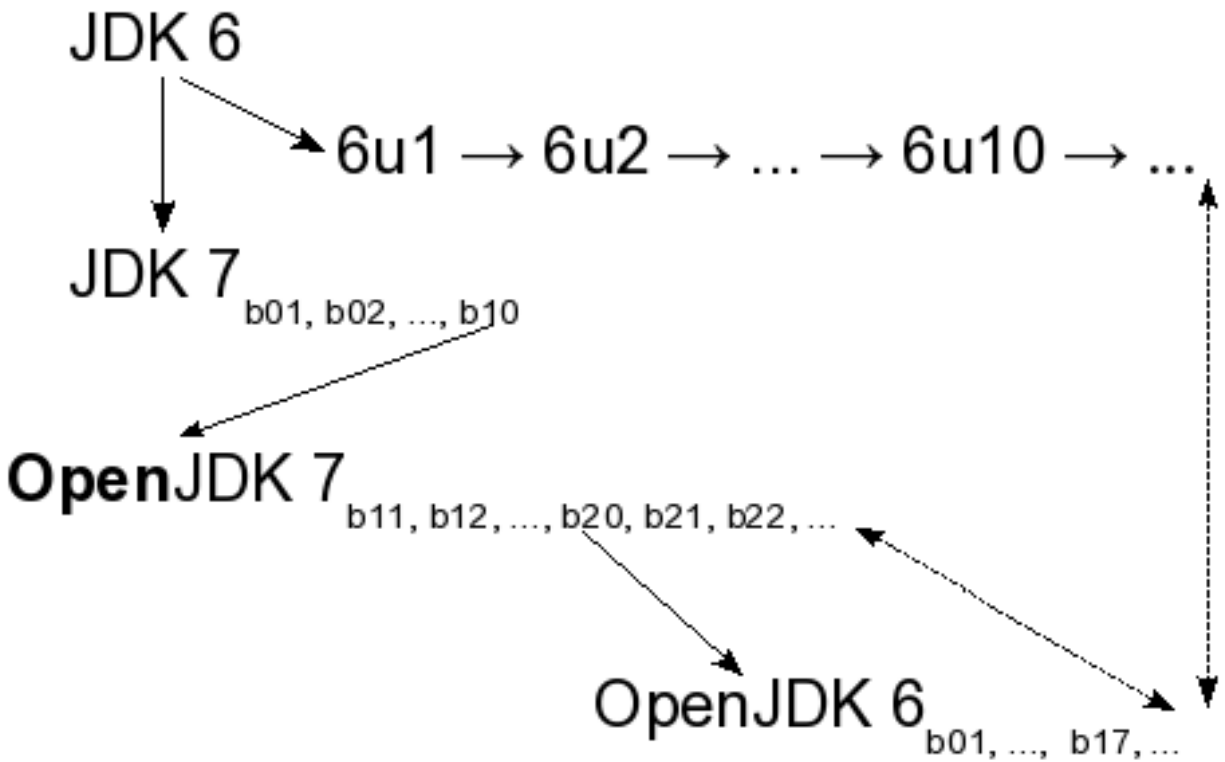
Il peut alors être nécessaire d'identifier une version précise du compilateur et/ou de la machine virtuelle : Ça n'est pas simple.

La numérotation des versions :

$$1.0 \longrightarrow 1.1 \longrightarrow \underbrace{1.2 \longrightarrow 1.3 \longrightarrow 1.4 \longrightarrow 5.0}_{\text{Toutes ces versions : } \textbf{Java 2}} \longrightarrow 6.0$$

Tout se complique

Attention, avec l'arrivée de la GPL tout se complique :



Les versions de Java

JDK 1.0 (1996 - 211 classes et interfaces)
Version initiale.

JDK 1.1 (1997 - 477 classes et interfaces)
Ajoute : classes internes, JavaBeans, JDBC, Java Remote Invocation (RMI).

J2SE 1.2 (1998 - 1 524 classes et interfaces) – Playground
Ajoute : réflection, SWING, compilateur JIT (Just in Time), Java IDL pour Corba.

J2SE 1.3 (2000 - 1 840 classes et interfaces) – Kestrel
Ajoute : HotSpot JVM, service de nomage (JNDI) et JavaSound.

Les versions de Java (suite)

J2SE 1.4 (2002 - 2 723 classes et interfaces) – Merlin
Ajoute : mot-clé assert, expressions rationnelles, chaînage d–exception, parser XML et du moteur XSLT (JAXP), extensions de sécurité JCE (Java Cryptography Extension) et Java Web Start.

J2SE 5.0 (2004 - 3 270 classes et interfaces) – Tiger
Ajoute : syntaxe à la foreach, enumerations (*enum*), classe Integer, autoboxing/unboxing

Java SE 6 (2006 - 3 777 classes et interfaces) – Mustang
Ajoute : covariance (redéfinition avec modification du type de retour), @overhiding.

Java SE 7 – Nom de code Dolphin
Ajoutera : des closures (en cours de spécifications).
Ce sera la première Version 100% open source.

Outline

Java *versus* C++

- Filiation historique :
  - **1983** (AT&T Bell) : C++
  - **1991** (Sun Microsystems) : Java
- Java est très proche du langage C++ (et donc du langage C).
- Toutefois Java est plus simple que le langage C++, car les points "critiques" du langage C++ (ceux qui sont à l'origine des principales erreurs) ont été supprimés.
- Cela comprend :
  - Les pointeurs
  - La surcharge d'opérateurs
  - L'héritage multiple

Java *versus* C++ : concepts (2)

De plus,

- ✍ Tout est dynamique : les instances d'une classe sont instanciées dynamiquement.
- ✍ La libération de mémoire est transparente pour l'utilisateur. Il n'est pas nécessaire de spécifier de mécanisme de destruction. La libération de l'espace mémoire est prise en charge un gestionnaire appelé garbage collector ⇒ chargé de détecter les objets à détruire.

Notes

- gain de fiabilité (pas de désallocation erronée).
- a un coût (perte en rapidité par rapport au C++).

Java *versus* C++ : production du binaire

Une fois achevée la production du logiciel, un choix doit être fait entre fournir le source ou le binaire pour la machine du client.

Généralement, une entreprise souhaite protéger le code source et distribuer le code binaire.

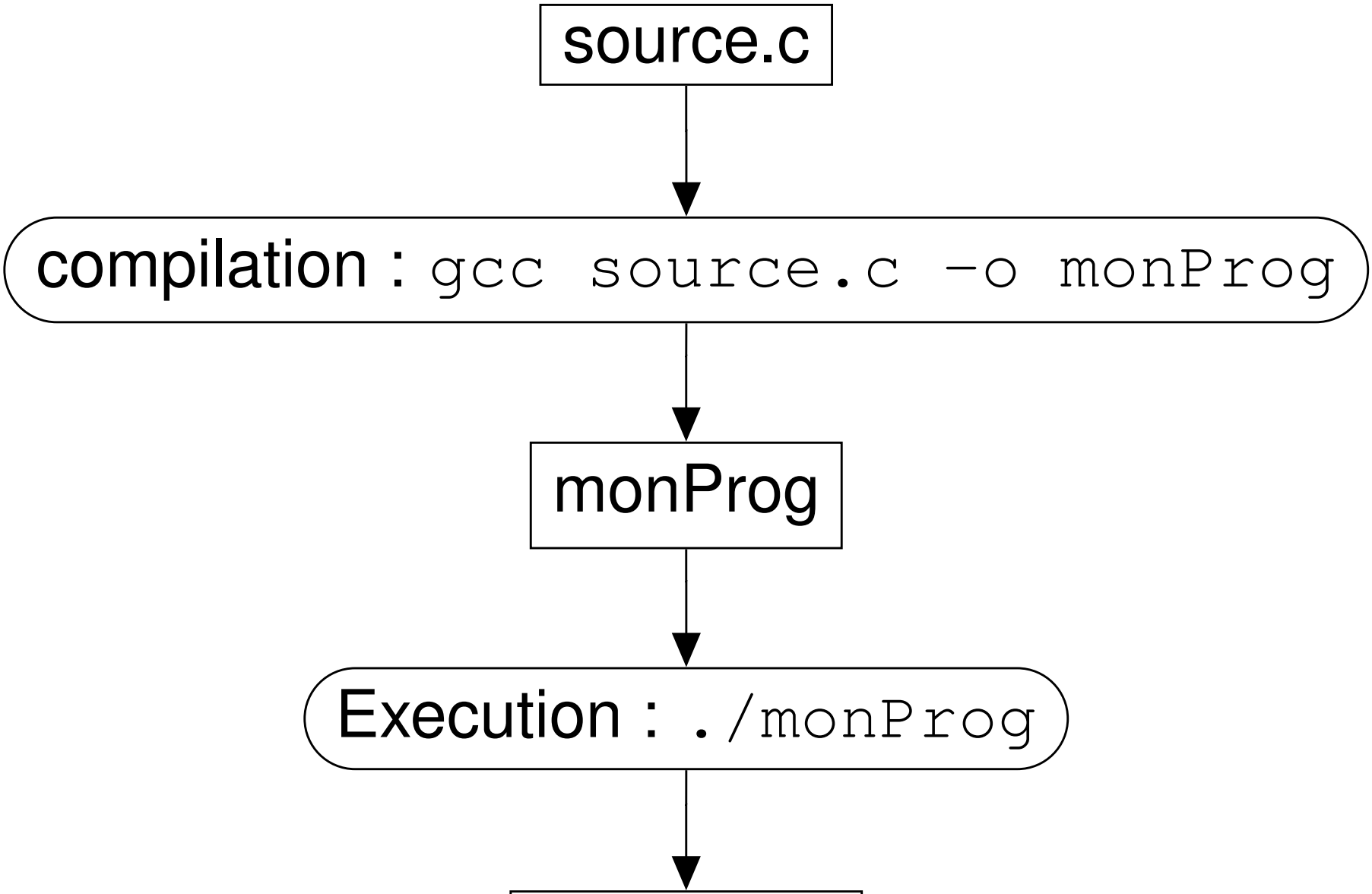
Le code binaire doit donc être portable sur des architectures différentes (processeur, système d'exploitation, etc.).

Java *versus* C++ : compilateur C++

À l'instar du compilateur C, le compilateur C++ produit du code natif, *i.e.*, qu'il produit un exécutable propre à l'environnement de travail ou le code source est compilé.

On doit donc créer les exécutables pour chaque type d'architecture potentielle des clients.

Java *versus* C++ : chaîne de production du C



Java *versus* C++ : compilateur Java

En Java, le code source n'est pas traduit directement dans le langage de l'ordinateur.

Il est d'abord traduit dans un langage appelé "bytecode", langage d'une machine virtuelle (JVM – Java Virtual Machine) définie par Sun.

Portabilité

Le *bytecode* généré par le compilateur ne dépend pas de l'architecture de la machine où a été compilé le code source, c'est-à-dire que les bytecodes produits sur une machine pourront s'exécuter (au travers d'une machine virtuelle) sur des architectures différentes.

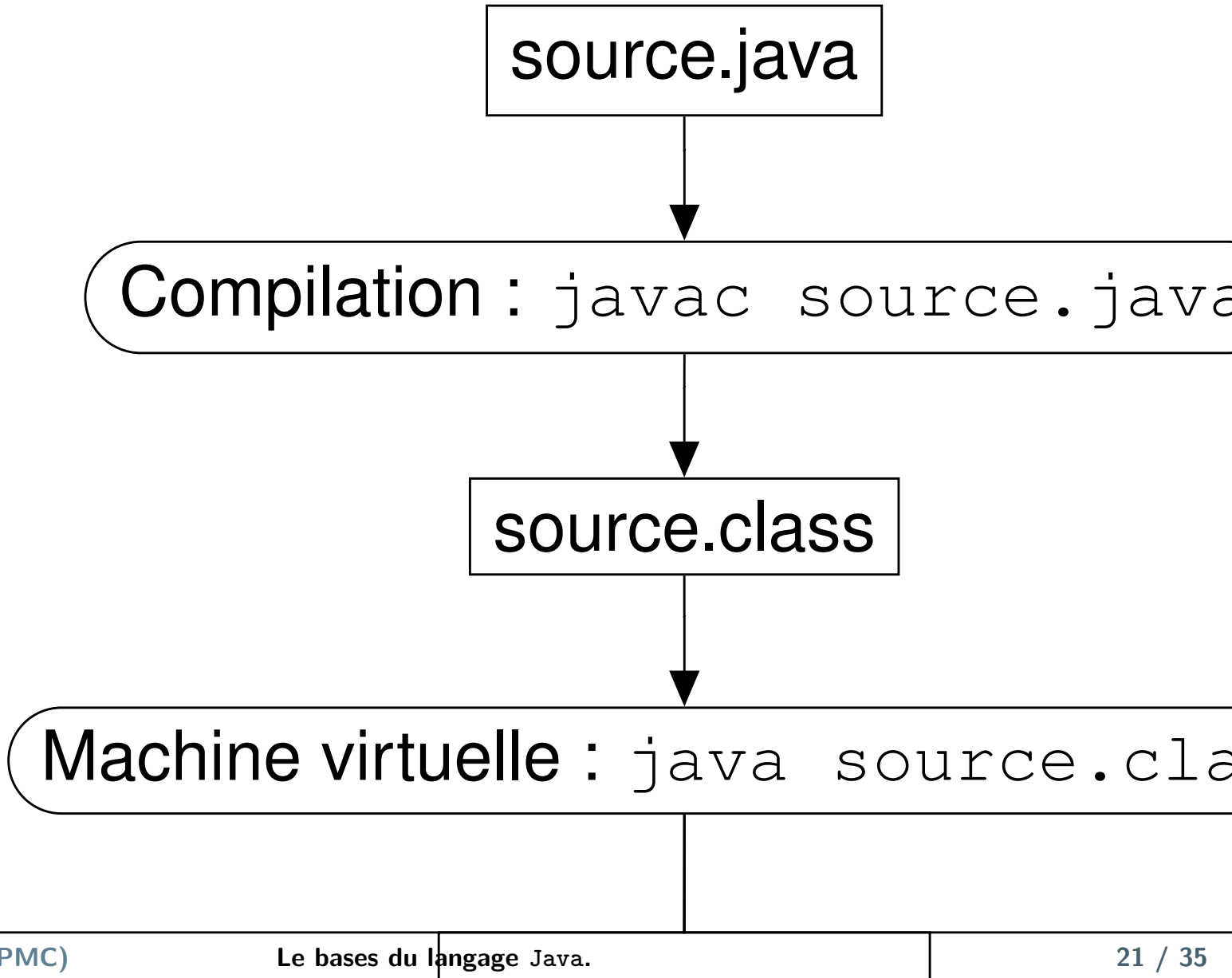
Exécution du bytecode

Le bytecode doit être exécuté par une Machine Virtuelle Java.

Cette JVM n'existe pas. Elle est simulée par un programme qui :

1. lit les instructions (en bytecode) du programme .class
2. fait une passe de vérification (type opérande, taille de pile, flot données, variable bien initialisé,...) pour s'assurer qu'il n'y a aucune action dangereuse.
3. fait plusieurs passes d'optimisation du code
4. les traduit dans le langage natif du processeur de l'ordinateur
5. lance leur exécution

Java *versus* C++ : chaîne de production du Java



Coût de la JVM sur les performances.

Les vérifications effectuées sur le bytecode et la compilation du bytecode vers le langage natif du processeur, ralentissent l'exécution des classes Java.

Mais les techniques de compilation à la volée "Just In Time (JIT)" ou "Hotspot" réduisent ce problème : elles permettent de ne traduire qu'une seule fois en code natif les instructions qui sont (souvent pour Hotspot) exécutées.

Java *versus* C++ : en résumé...

Le langage Java est :

- « C-like » : Syntaxe familière aux programmeurs de C
- Orienté objet : Tout est objet, sauf les types primitifs (entiers, flottants, booléens, ...)
- Robuste : Typage fort, pas de pointeurs, etc.
- Code intermédiaire : Le compilateur ne produit que du bytecode indépendant de l'architecture de la machine où a été compilé le code source

Note

Java perd (un peu) en efficacité par rapport à C++// mais gagne (beaucoup) en portabilité.

Outline

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

$$\boxed{\text{Système informatique} = \text{Structures de données} + \text{Traitements}}$$

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

$$\boxed{\text{Système informatique} = \text{Structures de données} + \text{Traitements}}$$

Le cycle de vie d'un système peut être décomposé en deux grandes phases :

- Une phase de **production** qui consiste à réaliser le logiciel.
- Une phase de **maintenance** qui consiste à corriger et à faire évoluer le logiciel.

Problématique de la programmation

Le schéma simplifié d'un système informatique peut se résumer par la formule :

$$\boxed{\text{Système informatique} = \text{Structures de données} + \text{Traitements}}$$

Le cycle de vie d'un système peut être décomposé en deux grandes phases :

- Une phase de **production** qui consiste à réaliser le logiciel.
- Une phase de **maintenance** qui consiste à corriger et à faire évoluer le logiciel.

Lors de la production du système (au sens industriel du terme), le concepteur a deux grandes options :

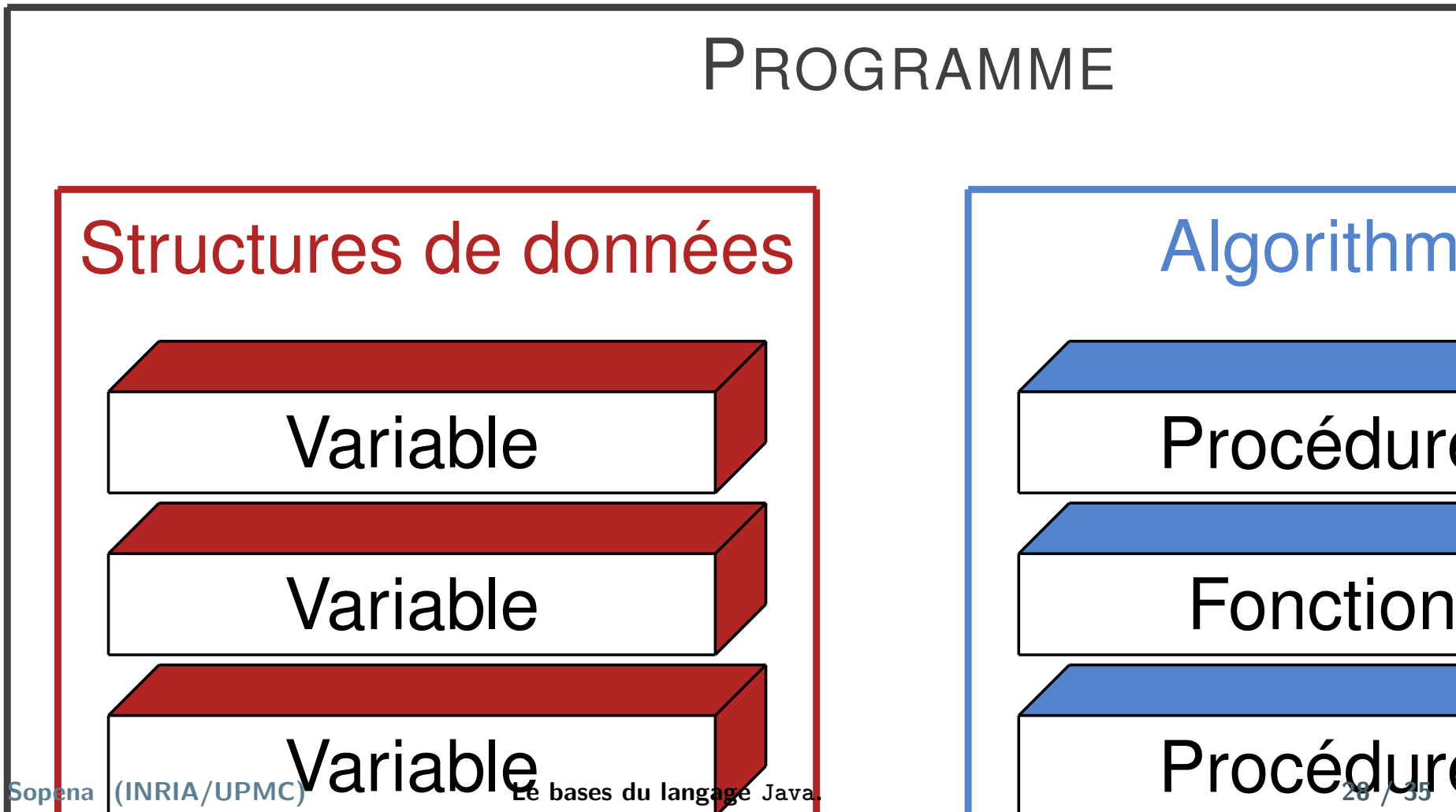
- ☞ soit orienter sa conception sur **les traitements**.
- ☞ soit orienter sa conception sur **les données**.

Outline

Conception par traitements.

Conception par traitements : principe

Principe : On sépare les données des moyens de traitement de ces données.



Conception par traitements : +/-

- Les premiers concepteurs de système informatique ont adopté cette approche : systèmes d'exp., gestionnaires de fenêtres, logiciels de gestion, logiciels de bureautique, logiciels de calcul scientifique, etc.
- De nombreux systèmes informatiques sont encore développés selon cette approche.

☞ Systèmes *ad-hoc*, *i.e.*, adaptés au problème de départ, mais dont la maintenance est difficile.

☞ Les traitements sont généralement beaucoup moins stables que les données : changement de spécification, ajout de nouvelles fonctionnalités, etc.

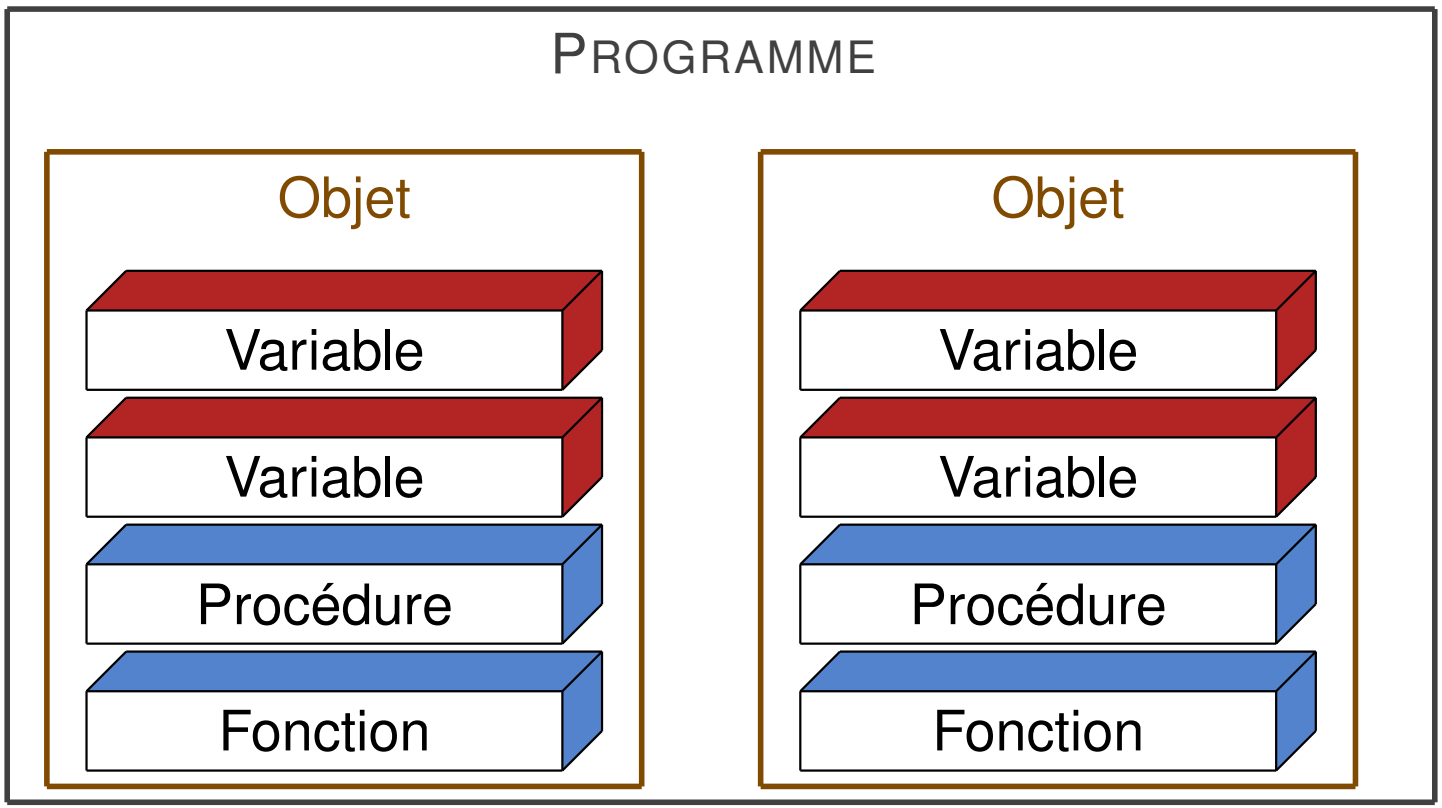
☞ Les structures de données sous-jacentes sont choisies en relation étroite avec les traitements à effectuer.

Outline

Conception par objets.

Conception par objets : principe

Principe : afin d'établir de façon stable et robuste l'architecture d'un système, il semble raisonnable de s'organiser autour des données manipulées.



Conception par objets : points clés

- La construction d'un système va s'axer principalement sur la détermination des données dans un premier temps et la réalisation les traitements (de haut-niveau) agissant sur ces données dans un second temps.
- Cette approche permet de bâtir des systèmes plus simples à maintenir et à faire évoluer.
- On regroupe dans une même entité informatique, appelé objet, les structures de données et les moyens de traitement de ces données.

Le concept d'objet : définition

Définition

*Un **objet** est une entité autonome, qui regroupe un ensemble de propriétés (données) cohérentes et de traitements associés.*

À retenir

Ne commencez pas par vous demander ce que fait l'application mais ce qu'elle manipule.

Le concept d'objet : vocabulaire

- Les structures de données définies dans l'objet sont appelés ses attributs (propriétés).
- Les procédures et fonctions définies dans l'objet sont appelés ses méthodes (opérations).
- Les attributs et méthodes d'un objet sont appelés ses membres.
- L'ensemble des valeurs des attributs d'un objet à un instant donné est appelé état interne.