

Cours de Java

Ahmed Zidna, Bureau : B37.
Département Informatique de l'IUT
Univerisité de Lorraine, Ile du Saulcy, F-57045 METZ
ahmed.zidna@univ-lorraine.fr

Objectifs

- ▶ Acquérir les concepts de la programmation objet (UML)
- ▶ Acquérir les bases de la programmation objet Java
- ▶ Développer des applications java.
- ▶ Réaliser un projet Java avec une interface graphique.

- POO : 24h TD+ 36h TP (Z. Habbas + R.Othman + A.Zidna)

- POO : 24h TD+ 36h TP (Z. Habbas + R.Othman + A.Zidna)
- COO : 24h TD (S. Minich + R.Othman + A.Zidna)

- POO : 24h TD+ 36h TP (Z. Habbas + R.Othman + A.Zidna)
- COO : 24h TD (S. Minich + R.Othman + A.Zidna)

Organisation du cours

- POO : 24h TD+ 36h TP (Z. Habbas + R.Othman + A.Zidna)
- COO : 24h TD (S. Minich + R.Othman + A.Zidna)
- Java : 2 TP notés + un projet COO-POO JAVA par binôme

Organisation du cours

- POO : 24h TD+ 36h TP (Z. Habbas + R.Othman + A.Zidna)
- COO : 24h TD (S. Minich + R.Othman + A.Zidna)
- Java : 2 TP notés + un projet COO-POO JAVA par binôme
- COO : Devoir + projet COO-POO JAVA par binôme

① Introduction

① Introduction

① Introduction

② Généralités

Sommaire

- ① Introduction
- ② Généralités
- ③ Classe

- ① **Introduction**
- ② Généralités
- ③ Classe
- ④ Héritage

- ① **Introduction**
- ② Généralités
- ③ Classe
- ④ Héritage
- ⑤ Collection

Sommaire

- 1 Introduction
- 2 Généralités
- 3 Classe
- 4 Héritage
- 5 Collection
- 6 Exception

- ① **Introduction**
- ② Généralités
- ③ Classe
- ④ Héritage
- ⑤ Collection
- ⑥ Exception
- ⑦ Interface graphique

- ① **Introduction**
- ② Généralités
- ③ Classe
- ④ Héritage
- ⑤ Collection
- ⑥ Exception
- ⑦ Interface graphique

- 1 **JAVA La maîtrise**
Jérôme Bougeault
Le guide de formation Tsoft, Eyrolles

- ① **JAVA La maîtrise**
Jérôme Bougeault
Le guide de formation Tsoft, Eyrolles
- ② **Références complètes Java**
H.Schildt, First Interactive

- 1 **JAVA La maîtrise**
Jérôme Bougeault
Le guide de formation Tsoft, Eyrolles
- 2 **Références complètes Java**
H.Schildt, First Interactive
- 3 **Programmer en Java**
Claude Delannoy, Eyrolles

- ① **JAVA La maîtrise**
Jérôme Bougeault
Le guide de formation Tsoft, Eyrolles
- ② **Références complètes Java**
H.Schildt, First Interactive
- ③ **Programmer en Java**
Claude Delannoy, Eyrolles
- ④ **consulter des cours sur le web**

Java en quelques mots

Java n'est pas vraiment novateur :

- ▶ les fonctionnalités dont il dispose sont utilisées depuis des années par d'autres langages

Java n'est pas vraiment novateur :

- ▶ les fonctionnalités dont il dispose sont utilisées depuis des années par d'autres langages
- ▶ Java en a fait une très bonne synthèse

Java en quelques mots

Java n'est pas vraiment novateur :

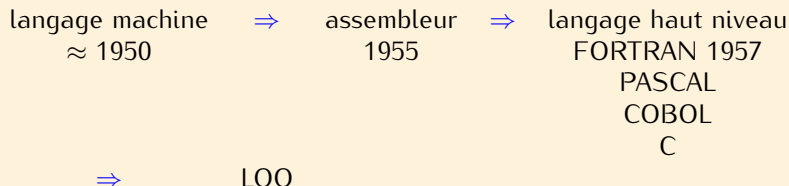
- ▶ les fonctionnalités dont il dispose sont utilisées depuis des années par d'autres langages
- ▶ Java en a fait une très bonne synthèse
- ▶ Java ne doit que sa syntaxe à C/C++

Java en quelques mots

Java n'est pas vraiment novateur :

- ▶ les fonctionnalités dont il dispose sont utilisées depuis des années par d'autres langages
- ▶ Java en a fait une très bonne synthèse
- ▶ Java ne doit que sa syntaxe à C/C++
- ▶ Java est orienté objet, on ne peut l'utiliser efficacement sans connaître ce type de programmation.

Evolution des langages I

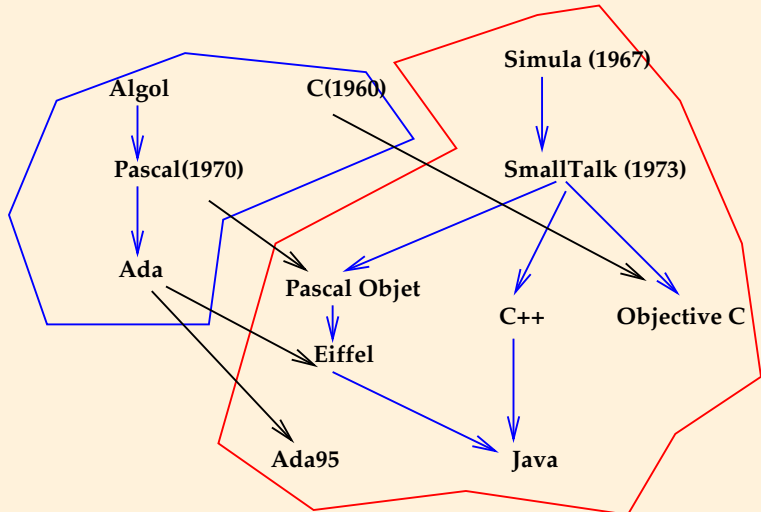


- ▶ Au cours de cette évolution, les langages s'éloignent des contraintes du matériel.
- ▶ Les LOO sont adaptés à la réalisation de logiciel de grande taille.
- ▶ Les LOO sont utilisés dans des domaines : (CAO, IA, Bdd, GL...).

Evolution du Langage

Structurés

Objets



- ▶ **Simula (1967)** : un des ancêtre de Java, le premier à introduire les concepts de la POO.
- ▶ **Smalltalk (1972)** : inspiré par Simula pour la POO et Lisp pour le côté langage interprété.
- ▶ **C++ (1983)** : 1983 : Bjarne Soustrup a fait évoluer le langage C en langage C++ qui offre les concepts de programmation orientée objet, le polymorphisme, ..etc.
- ▶ **Java (1995)** : langage plus simple, inspiré de Smalltalk et C++ (a éliminé par exemple : héritage multiple, surcharge des opérateurs..).

Avantage de la POO I

- ▶ Ils apportent une plus grande souplesse de programmation :
- ▶ la possibilité de programmer de façon rigoureuse,
- ▶ la possibilité de contrôler très efficacement le code produit,
- ▶ un meilleur formalisme algébrique,
- ▶ la réutilisabilité du code.

Java en quelques mots I

Pour **SUN (Oracle)**, Java est un langage :

- ▶ simple, orienté objet, distribué, interprété,
- ▶ robuste, sûr, indépendant de l'architecture, portable,
- ▶ efficace, multi-thread, dynamique,
- ▶ mais qui ne fait pas encore le café !

Simple mais pas simpliste !

Java est conçu pour être assimilé rapidement :

- ▶ faible nombre de constructions
- ▶ syntaxe familière au plus grand nombre (C, C++)
- ▶ évite les fonctionnalités rarement utilisées :
 - structures, unions,
 - héritage multiple, surcharge d'opérateur,
 - l'arithmétique sur les pointeurs, préprocesseur
- ▶ mémoire est gérée automatiquement

① Orienté objet

- Java **ne supporte que le style programmation objet** contrairement à C++.
- que **des objets** (des instances de classes) sauf les types de base tels que :int,double...
- contrairement à C ou C++, le **codage des types de base est indépendant** de l'architecture

② Interprété

- le code source **n'est pas traduit directement** dans le langage de l'ordinateur
- le compilateur génère **des bytes codes** indépendants de toute architecture.

① Robuste

- fortement typé
- gestion mémoire automatique (ramasse-miette)
- gestion des exceptions beaucoup plus stricte qu'en C++

② Efficace

- Java est efficace par rapport aux langages interprétés tels que : Basic, perl, Tcl
- actuellement, le byte code interprété est réputé plus lent que C/C++

① Distribué

- conçu pour supporter des applications réparties sur le réseau
- il gère les protocoles TCP-IP. L'accès à une ressource grâce à une URL est proche de l'accès à un fichier.

② Sûr

- En Java, la sécurité est assurée par la robustesse du langage et par le runtime :
- les bytes code sont typés, et on peut vérifier au chargement qu'il n'y a pas "tentative de fraude"

Indépendance de l'architecture et Portabilité I

- ▶ les applications Java peuvent tourner sur n'importe quelle machine disposant de la machine virtuelle (JVM) sans recompilation
- ▶ les accès aux ressources du système se font de façon homogène à travers une API;
- ▶ l'utilisation de l'AWT permet une manipulation homogène de la GUI locale.
- ▶ le runtime est écrit en C ANSI, le compilateur en Java, ce qui rend les différentes architectures relativement faciles à xsupporter.

① **Multi-threading** : Java prend en charge la programmation multiprocessus qui permet décrire plusieurs tâches simultanément.

- le multi threading est supporté au niveau du langage (primitives de synchronisation, exclusion mutuelle)

② **Dynamique**

- les classes sont chargées en cours d'exécution et uniquement lorsque c'est nécessaire

La doc API : Application Programming Interface I

La doc [API](#) : docs.oracle.com/javase/7/docs/api/ évolue tout le temps :

- ▶ [java.lang](#) : contient les classes les plus centrales du langage. Il contient la classe [Object](#) qui est la super-classe ultime de toutes les classes ([String](#), [Thread](#) ...).
- ▶ [java.util](#) : complète [java.lang](#) ([Date](#), [Vector](#),...).
- ▶ [java.io](#) : contient les classes nécessaires aux entrées-sorties.
- ▶ [java.awt](#) : awt pour Abstract Window Toolkit contient des classes pour fabriquer des interfaces graphiques.

La doc API : Application Programming Interface I

- ▶ [java.applet](#) : utile pour faire des applets, applications utilisables à travers le Web.
- ▶ [java.sql](#) : pour l'accès aux base de données (JDBC : Java DataBase Connector).
- ▶ [java.rmi](#) : pour invoquer les méthodes d'objets distants (RMI : Remote Method Invocation).
- ▶ [java.net](#) : fournit une infrastructure pour la prog réseau.

Les outils Java I

- ▶ Les outils fournis sont (entre autres)
- ▶ **javac** : le compilateur
- ▶ **java** : la machine virtuelle
- ▶ **javadoc** : le générateur de documentation
- ▶ **appletviewer** : la machine d'exécution d'applets
- ▶ **jar** : l'archiveur
- ▶ **jdb** : le débogueur.

Compiler et exécuter en Java ? I

- Tous les fichiers sources doivent avoir l'extension **.java**.

```
1 public class Somme{  
    public static void main(String[] arg){  
3     int i, somme = 0;  
    for (i = 1; i <= 100; i++) somme += i;  
5     System.out.println("la_somme_des_100"+"premiers_entiers:"  
        +somme);  
    }  
7 }
```

- Pour compiler : **javac Somme.java --> Somme.class**
- Pour exécuter : **java Somme**

Compiler et exécuter en Java ? I

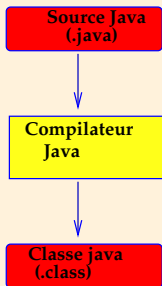
- Un programme Java est structuré en modules appelés classes

```
1  public class Exemple{  
   public static void main (String[ ] args)  
3  {  
       int i=3;  
5       afficher(i);  
   }  
7  public static void afficher(int n)  
   {  
9       for(int i=0; i<n;i++)  
         System.out.println("Je_fais_du_Java");  
11  }  
   }
```

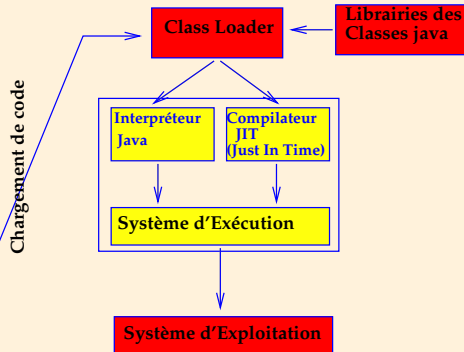
- Une classe est une suite de déclarations et de méthode

Compilation Exécution

Environnement de compilation

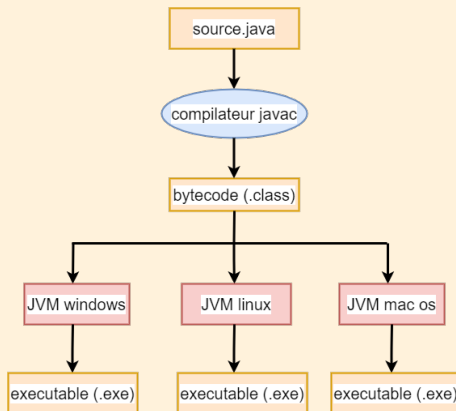


Environnement d'exécution



Compilation Exécution

- Les applications Java sont appelées **WORA (Write Once Run Anywhere)**



JDK pour développer des applications Java

- ▶ Le kit de développement Java JDK fournit
 - ⇒ l'environnement d'exécution JRE
 - ⇒ les outils de développement (javac,JavaDoc,Debugger,.).
- ▶ L'environnement d'exécution Java (JRE) est un progiciel :
 - ⇒ la machine virtuelle JVM
 - ⇒ des bibliothèques de classes Java,
 - ⇒ des composants pour exécuter des applications Java.
- ▶ La machine virtuelle Java (JVM) :
 - ⇒ convertit le Bytecode en code machine natif : ensemble d'instructions que le processeur d'un ordinateur exécute directement.

JVM,JRE et JDK

