Chapitre 1 : introduction

objectifs du cours

introduction à Java

Objectifs généraux

Spécifier, concevoir, réaliser des composants logiciels, des applications informatiques en Java.

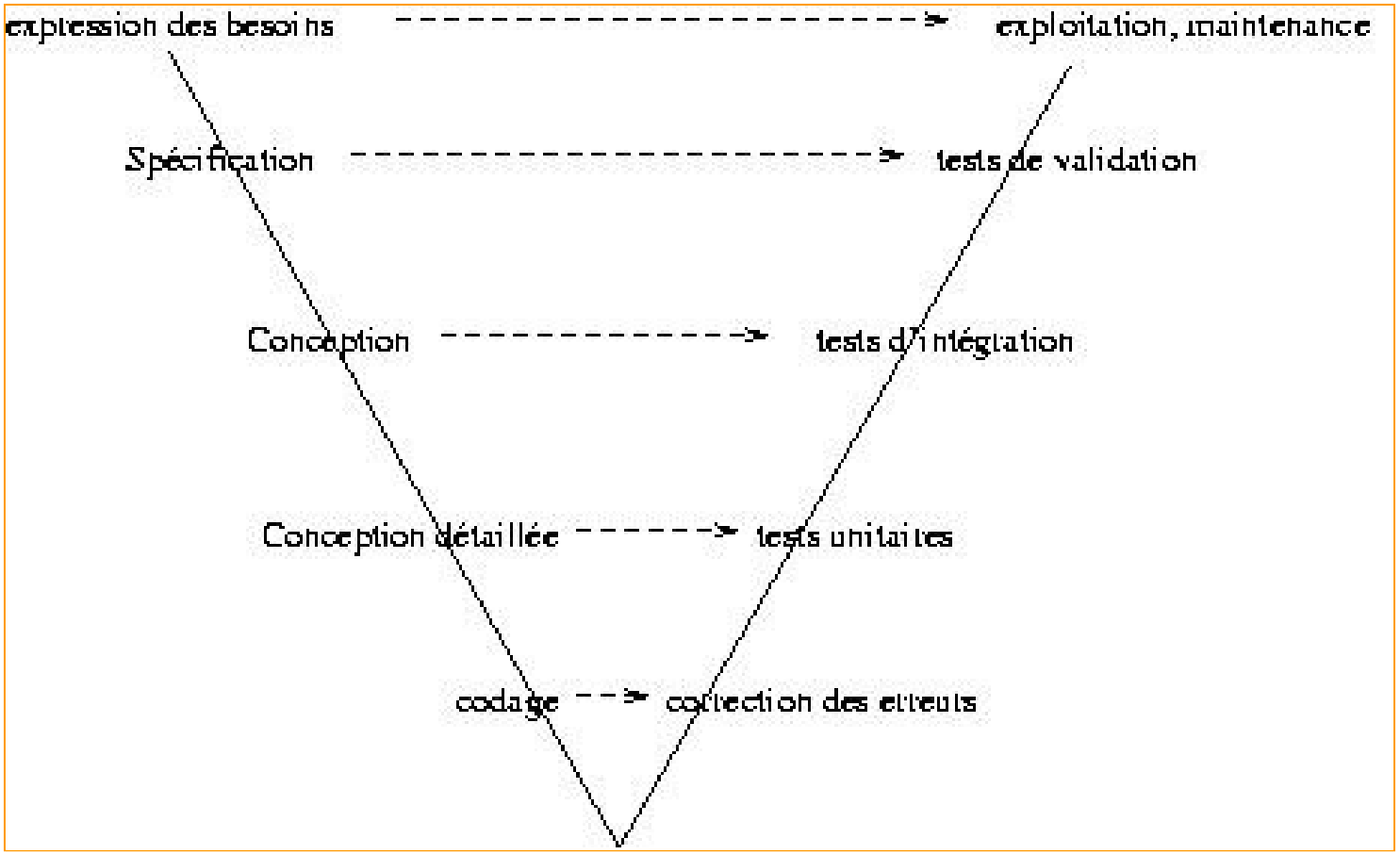
Avec les qualités requises pour un produit industriel :

efficacité
fiabilité
lisibilité
réutilisabilité
extensibilité

Donner les bases durables du métier de concepteur, développeur d'applications informatiques

Permettre une adaptation rapide aux différents langages de programmation

Cycle de vie du logiciel



Environnement de programmation

applications informatiques	**génie logiciel**
développement d'une application	cycle de vie
étapes de développement	analyse, conception, codage, tests, maintenance
environnement de programmation	outils, méthodes, langages
méthodes	UML, SADT, SART, Merise
outils	interface utilisateur, gestionnaire de données, compilateur, chargeur, éditeur de liens
langages	Ada, C++, Eiffel, Java

Les étapes du cycle de vie
- Analyse -

Dans cette étape, on s'attache à répondre à trois grandes questions :

- comprendre le problème, identifier les données et les résultats attendus
- dégager les grandes fonctionnalités du système (spécification fonctionnelle)
- identifier les ressources nécessaires (matérielles et humaines)

Cette phase est indépendante de tout langage d'implantation.

Les étapes du cycle de vie

- Conception -

On décompose le problème en sous-problèmes plus simples :

- Ceci donne naissance à un ensemble d'unités informatiques (composants) qui constituent le logiciel d'application.
- C'est dans cette phase que l'architecture du logiciel est élaborée. Les composants (modules ou unités) et leurs relations sont spécifiés.

Il s'agit d'un processus itératif qui peut conduire à un retour vers l'analyse.

Java peut être utilisé comme langage de conception.

Les étapes du cycle de vie

- Codage -

Il s'agit, dans cette étape, d'implanter (coder) la conception, c'est à dire l'ensemble des composants de l'application.

Plus précisément, il s'agit de traduire les traitements en terme de structures de contrôle et les données en termes des structures de données du LP

On utilise un langage de programmation (Ada, C, Java, C++, Eiffel, Cobol, Fortran, Pascal, ...) pour exprimer le code.

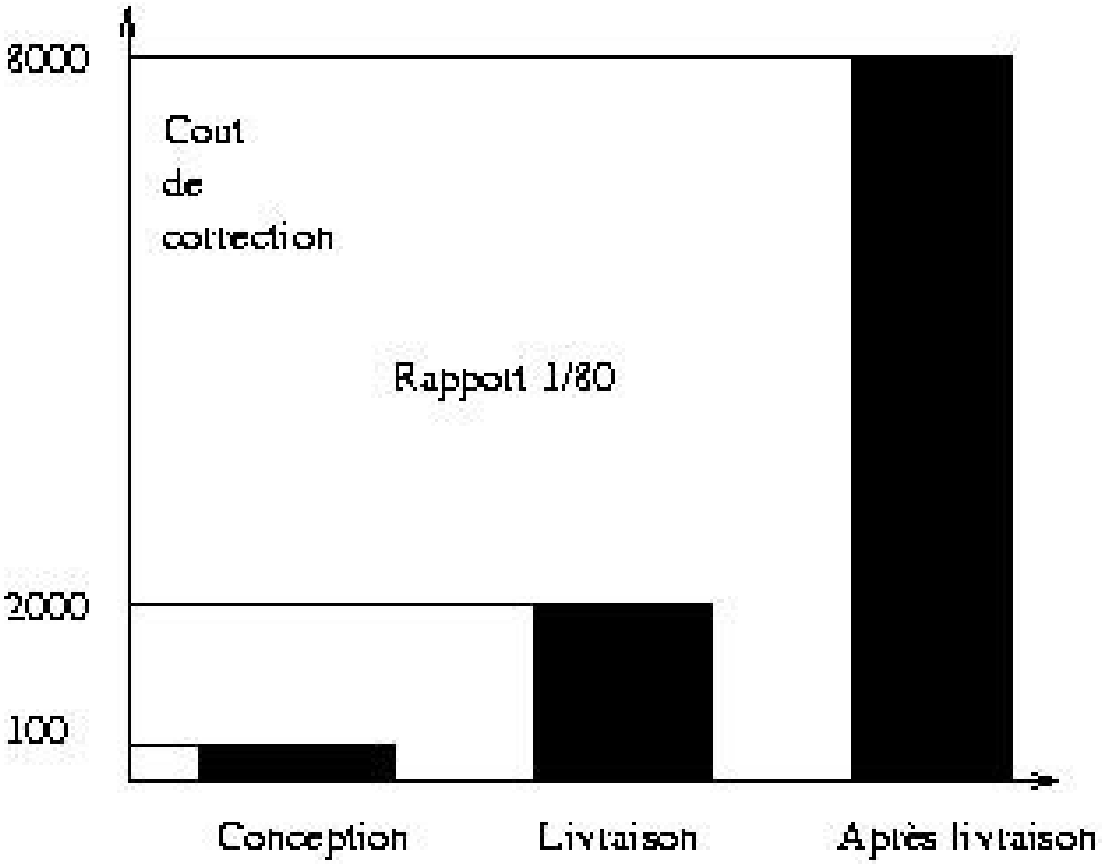
Les étapes du cycle de vie

- validation/vérification -

- ✓ tests unitaires (niveau composant) :
  - chaque composant fait, individuellement, l'objet de tests
- ✓ tests d'intégration (niveau système)

L'évolution du système est contrôlée à travers le processus conception/codage/test.

Qualités d'un logiciel



coût de la non qualité

Qualités d'un logiciel : - efficacité -

- ✓ Optimalité de l'utilisation des ressources (temps, espace)
- ✓ Importance de la macro-efficacité qui se reflète dans une bonne structure de la solution au dépens de la micro-efficacité

Qualités d'un logiciel :
- fiabilité -

L'industrie du logiciel est l'industrie la moins fiable.

- ✓ Un aspect de la fiabilité concerne la correction. On dit qu'un programme est correct si la solution répond bien au problème posé (spécification). L'application réalise les fonctions attendues par l'utilisateur dans les conditions normales et avec les performances attendues.

- ✓ Un autre aspect de la fiabilité concerne la récupération des pannes. Un programme est considéré comme fiable s'il propose un mode de fonctionnement dégradé, sans provoquer d'effets de bords.

Qualités d'un logiciel :
- lisibilité -

La base d'une bonne lisibilité est le maintien de la relation entre le problème et sa solution.

Au niveau du codage :

- bonne documentation de la solution
- bon style de codage

Au niveau de la conception :

- clarté de l'architecture

Qualités d'un logiciel : - extensibilité -

- ✓ Une application informatique évolue au fil du temps et des besoins de l'entreprise à laquelle elle appartient.
- ✓ Il est important de ne pas devoir réécrire une application lorsque les spécifications subissent des modifications.
- ✓ L'extensibilité est la qualité d'un logiciel peu sensible aux changements de spécification.

Qualités d'un logiciel :
- réutilisabilité, modularité -

Les composants logiciels qui forment l'application, conçus de manière à ne pas dépendre du contexte, deviennent réutilisables.

Une application est portable si elle est indépendante du système d'exploitation, du système dE/S et du matériel.

Les critères de modularité sont :

- couplage faible (interconnexion limitée)
- cohésion forte (éléments internes étroitement reliés)

Démarche d'analyse des problèmes

Déterminer le problème à résoudre :

- quelles sont les données d'entrée et leur nature ?
- quelles sont les données attendues en sortie et leur nature ?

Déterminer la méthode :

- comment modéliser les données (d'entrée, de sortie, intermédiaires) ?
- identifier, selon les données en entrée, les différents cas à traiter, et les traitement associés.
- exprimer sous-forme d'algorithme la manière d'obtenir le résultat final à partir des traitements sur les données.

Correction, complétude et lisibilité

- tous les cas des entrées et sorties ont-ils été prévus?
- obtient-on dans chaque cas ce que l'on voulait calculer ?
- l'algorithme est-il compréhensible par quelqu'un d'autre ?

Tests

- élaborer un jeu de tests représentatif de tous les cas possibles de données en entrée.

Notre approche

Afin de nous concentrer sur le coeur de l'activité de programmation, nous nous limiterons aux trois phases essentielles :

- spécification
- codage
- tests

Spécification

La spécification est la description du problème à résoudre.

Spécification informelle

- Le problème est décrit en langage naturel.
- La description conserve éventuellement quelques imprécisions, ambiguïtés.
- Elle est souvent incomplète

Spécification formelle

- Il s'agit d'une description complète et rigoureuse du problème, exprimée dans un langage formel (proche des maths).
- Elle permet de faire des preuves de correction et de terminaison d'un calcul.

Codage

Le codage correspond à la traduction de la solution d'un problème dans un langage de programmation (LP).

Par rapport à la phase de spécification, les problèmes posés sont de nature différente. Ils tiennent :

- aux caractéristiques physiques de la machine (E/S)
- à la représentation des données
- à la traduction des actions dans un LP

Tests

En général, un programme ne peut pas être testé pour toutes les données possibles.

Des jeux de tests sont élaborés de manière à visiter tous les chemins que peut prendre l'exécution du programme.

Cela ne fournit pas une preuve de la correction du programme, mais donne un indice de confiance dans son fonctionnement.

Exemple de spécification (1)

Spécification informelle

On veut savoir si un nombre entier n, non négatif, est un carré parfait.
Le résultat produit est :

- vrai si et seulement si `n` est un carré parfait
- un entier égal à la racine carrée entière de `n`.

Spécification formelle

```
{n>=0}    {(  m  [0..n]   vrai   m*m=n)    ( m  N
     faux   m*m<n<(m+1)*(m+1)}
```

Exemple de spécification (2)

Spécification informelle

Calcul de n^p, p est un entier naturel mais il peut être nul sauf si n est nul

Spécification formelle

$\{p \in [(p \geq 0) \ (p \quad 0 \quad n = 0)]\} \quad \{résultat = n^p\}$

Crise du logiciel

- complexité croissante
- coût
- qualité

En 1976, une étude met en lumière le coût prohibitif du logiciel dû à la complexité croissante :

- 75 $ par instruction développée
- 4000$ par instruction modifiée (après livraison)

Une étude (1)

Un autre étude menée en 1979 par le gouvernement américain et portant sur 487 projets illustre le coût exorbitant de la maintenance :

- 41,8% est dû à un changement de spécification
- 17,4% est dû à un changement dans le format des données
- 12,4% à des sorties d'urgence
- 9% à des défaillances nécessitant un débogage

Une étude (2)

La même étude indique que :

- 47% des applications délivrées ne sont jamais utilisées
- 29% payées mais non terminées
- 19% abandonnées ou réécrites
- 3% utilisées après modifications
- 2% utilisées sans changement

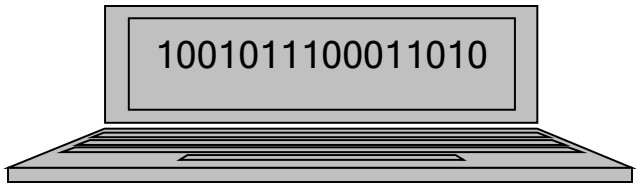
Une autre étude

Une enquête effectuée aux USA en 1986 auprès de 55 entreprises révèle que 53% du budget total d'un logiciel est affecté à la maintenance :

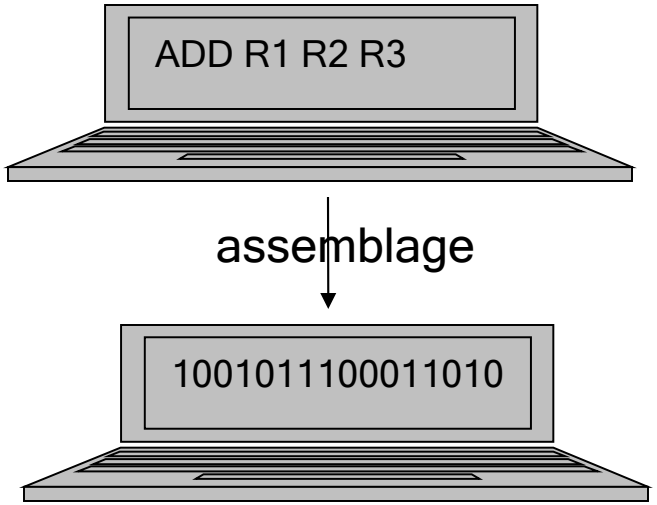
- 34% à la maintenance évolutive : modification des spécifications initiales
- 10% à la maintenance adaptative : nouvel environnement, nouveaux utilisateurs ;
- 17% à la maintenance corrective : correction des bogues ;
- 16% à la maintenance perfective : améliorer les performance sans changer les spécifications ;
- 6% à l'assistance aux utilisateurs ;
- 6% au contrôle qualité ;
- 7% l'organisation et au suivi ;
- 4% divers.

Langages et programmes (1/3)

langage machine/
langage natif
≡
ensemble d'instructions
primitives intégrées à une
machine
instructions sous forme binaire



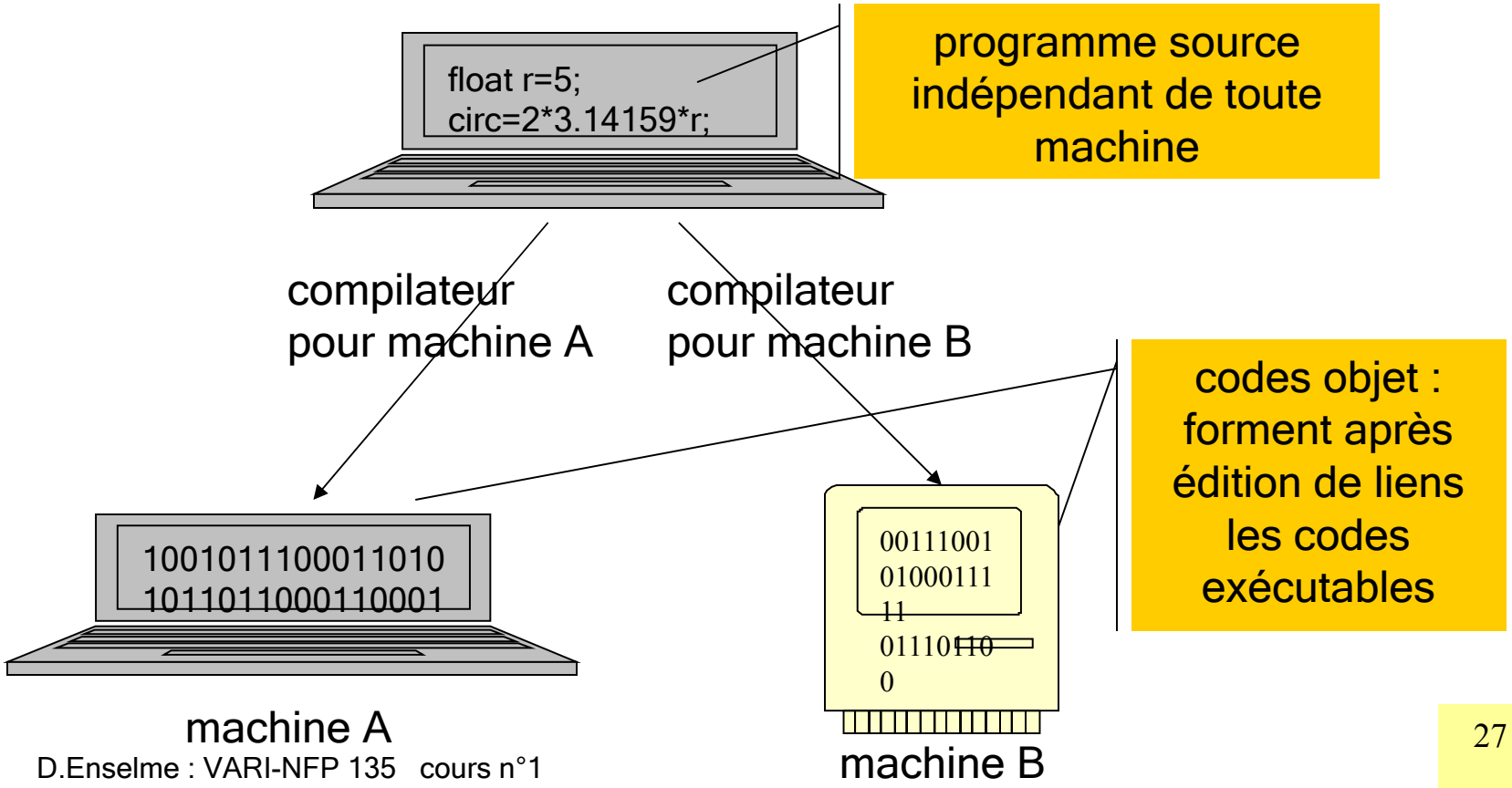
langage d'assemblage
≡
bas niveau
instructions machine symboliques
→
nécessité d'une traduction par un
assembleur



langage dépendant du processeur

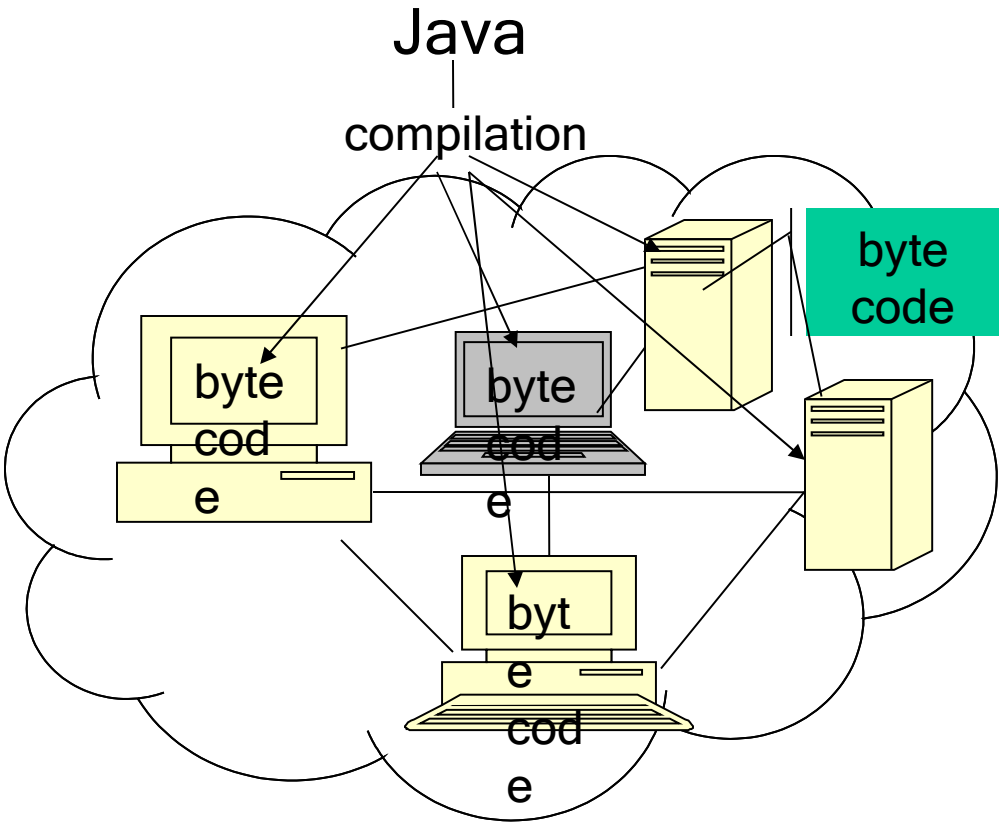
Langages et programmes (2/3)

langage de haut niveau
≡
plus proche du langage naturel



Langages et programmes (3/3)

Machines en réseau => créer des programmes exécutables sur n'importe quelle plate-forme sans recompilation



Choix de Java

Multithreading

Web et Applettes

Programmation évènementielle

Programmation réseau type client-serveur

Caractéristiques de Java

- C'est un langage simple qui hérite des constructions de C et C++
- C'est un langage **orienté objet** qui permet la conception et la réalisation d'applications complexes avec une architecture modulaire.
- C'est un langage **robuste** muni d'un mécanisme de gestion des exceptions qui permet de déceler et de traiter des erreurs pendant l'exécution du programme.
- Java est **indépendant de la plate forme** d'exécution, donc indépendant de la machine et de son système d'exploitation.
- Java est un langage **distribué**. Un programme peut être déployé sur plusieurs machines d'un réseau d'ordinateurs et exécuté sur celui-ci.
- C'est un langage **sûr**. Il possède des caractéristiques qui permettent une protection contre du code non fiable (virus). Il impose des contraintes aux applications Web dès que leur téléchargement dans un navigateur est effectué.

Java API

Application Program Interface (API) est un ensemble de classes et interfaces prédéfinies

3 éditions d'API :

- J2SE pour le développement d'applications coté client ou d'applettes
- J2EE pour le développement d'applications coté serveur
- J2ME développements pour mobiles

Outils de développement

Ces outils fournissent un environnement de développement intégrés dans une interface graphique (Integrated Development Environment)

Principaux IDE :

- JBuilder (Borland)
- NetBeans Open Source (Sun)
- Sun One (version commerciale de NetBeans)
- Eclipse Open Source (IBM)

Outil pédagogique : c'est celui que nous utiliserons en TPs

- Bluej (téléchargeable sur www.bluej.org)

Environnement Java

J2SE (Java 2 Standard Edition) disponible sur java.sun.com (actuellement J2SE 6.0)

comprend un JDK (Java Development Toolkit), ensemble de programmes (javac, java, javadoc, jar, appletviewer, ...) invocables à partir d'une ligne de commande rassemble :

- Environnement d'exécution (jre)
- Langage
- Application Programming Interfaces
- Bibliothèques

Compilation

Un compilateur traduit un programme source en langage machine

Inconvénient : le résultat dépend de l'ordinateur visé

L'idée de java est de placer un intermédiaire dans la phase de compilation

Au lieu de compiler dans un langage machine spécifique à un ordinateur particulier (langage natif), le compilateur Java transforme le programme source en un programme dans le langage machine de la machine virtuelle Java (JVM)

Ce langage est un langage de bas niveau appelé **bytecode.**
Il est indépendant de tout ordinateur

Interprétation

Avec un compilateur normal, le programme est exécuté directement par l'ordinateur. Le processeur lit les instructions et les exécute.

En Java, le programme compilé n'est pas directement compréhensible par un ordinateur.

Un programme, appelé interpréteur, traduit au vol (pendant l'exécution même du programme), instruction par instruction, le bytecode en instructions pour l'ordinateur.

Compilation vs interprétation

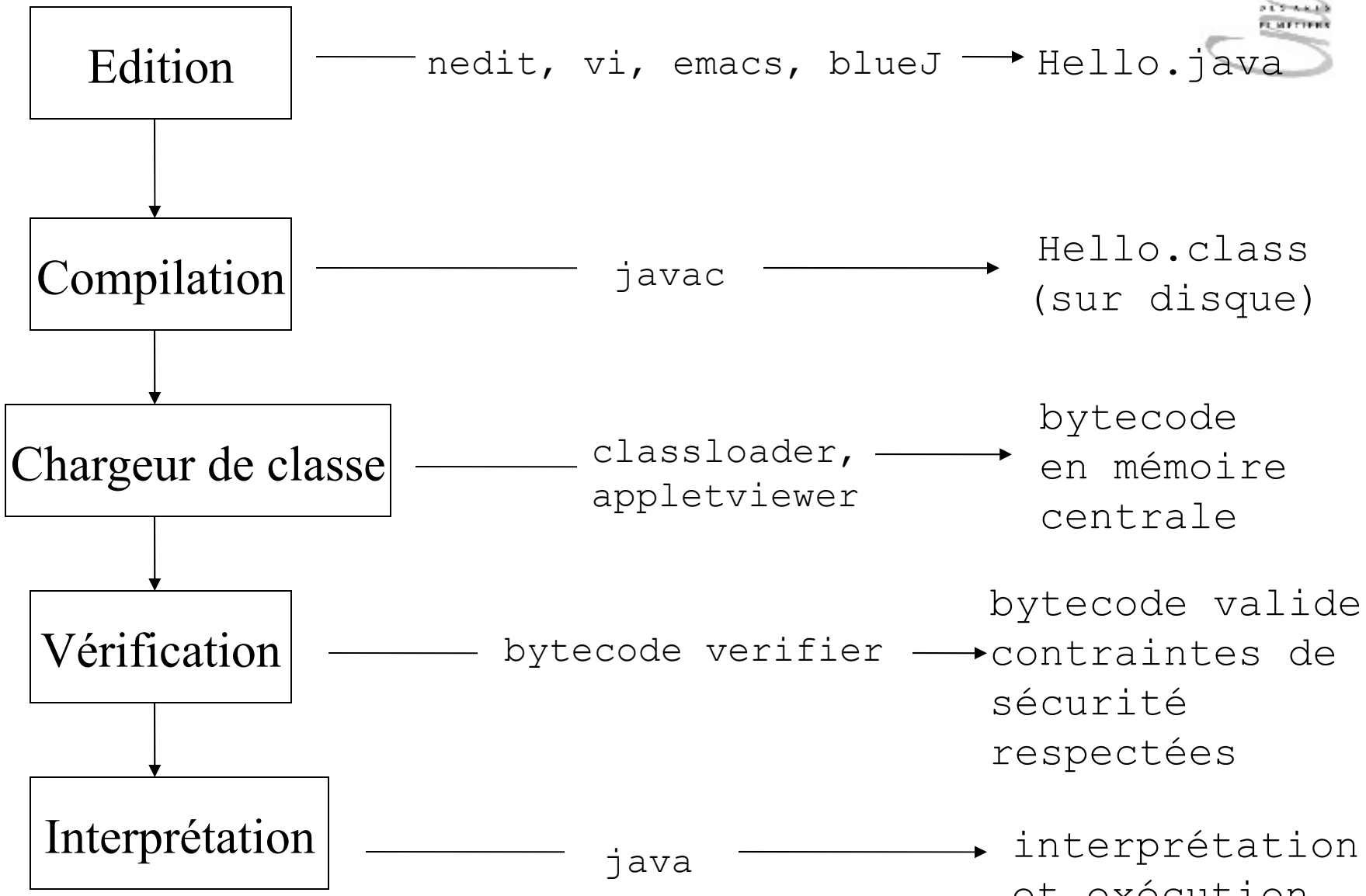
Chaque interprétation produit sa traduction. Le programme produit par l'interpréteur n'est pas conservé.

Le programme produit par le compilateur est conservé

L'interprétation est moins efficace (en temps) que la compilation

Un programme Java compilé peut être interprété sur tout ordinateur (la JVM est partout présente)

Pour exécuter un programme écrit dans un autre langage, il faut le recompiler


Edition
nedit, vi, emacs, blueJ
Hello.java
Compilation
javac
Hello.class
(sur disque)
Chargeur de classe
classloader,
appletviewer
bytecode
en mémoire
centrale
Vérification
bytecode verifier
bytecode valide
contraintes de
sécurité
respectées
Interprétation
java
interprétation
et exécution
du code

Un premier programme

Un programme Java est constitué de classes, au moins une appelée classe principale.

Une des classes doit contenir une méthode `main` chargée de recevoir les arguments de la commande de lancement du programme.

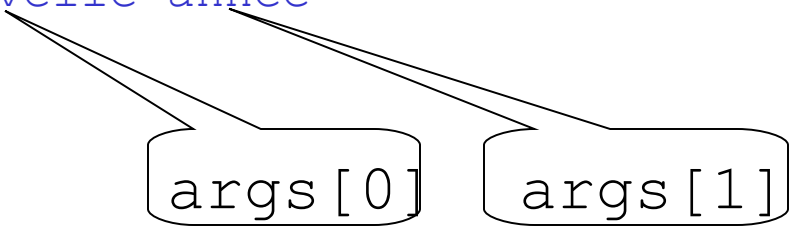
Le traditionnel premier programme est enregistré dans un fichier `Hello.java` contenant le code source suivant :

```
public class Hello{
  public static void main( String[] args ){
       System.out.println( "Hello!"+args[0]+args[1]+2007 );
  }
}
```

Production de programme

```
javac Hello.java
```

```
java Hello nouvelle année
```

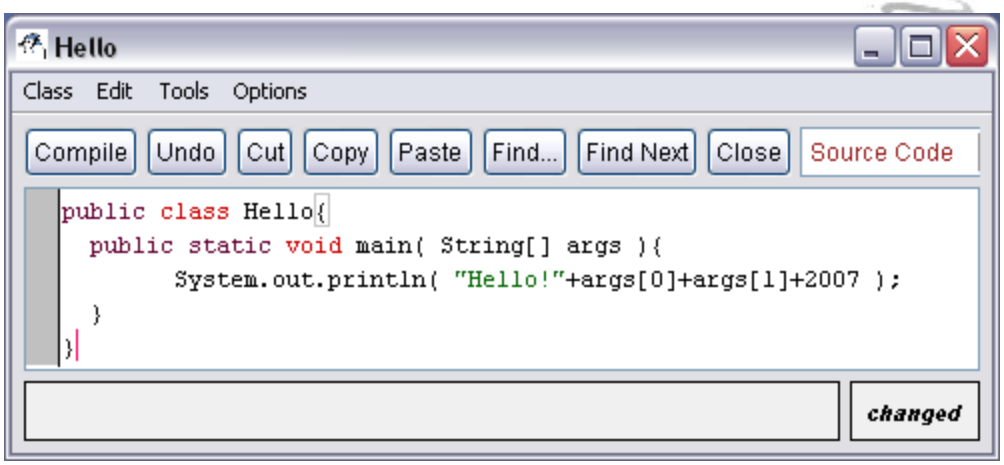


résultat affiché

```
Hello!nouvelleannée2007
```

Sous bluej

édition du programme

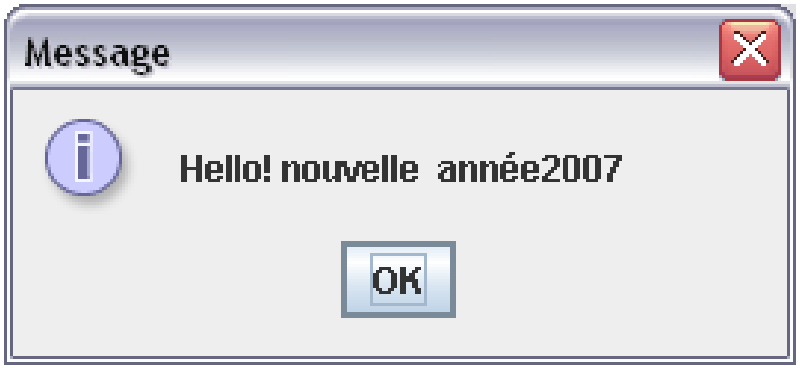


exécution du programme

résultat

Autre version

```
import javax.swing.JOptionPane;
public class Hello{
   public static void main( String[] args ){
        JOptionPane.showMessageDialog(null,
                  "Hello!"+args[0]+args[1]+2007 );
   }
}
```




```
javac Hello.java
java Hello nouvelle année
```

Quelques sites utiles

Cours en ligne

http://deptinfo.cnam.fr/new/spip.php?article808

Bibliographie

- Programmer en Java, C.Delannoy, Eyrolles
- Introduction to Java programming, Y.Daniel Liang, Pearson Prentice Hall
- Java how to program Deitel&Deitel, Prentice hall
- Java by Dissection, Ira Pohl & Charlie McDowell, Addison Wesley

Compilateurs, bibliothèques, tutoriels, documentation

http://java.sun.com