

Introduction à Java

Philippe Genoud

Philippe.Genoud@imag.fr

Christophe Bruley

Christophe.Bruley@inrialpes.fr

janvier 2004

certaines éléments de cette présentation sont issus du cours
« Programmation Java » de M. Buffa (ESSI Nice)
et aussi du cours
« Comment Java ? Java Bien » de Patrick Itey (INRIA)

1

La technologie JAVA

- **Java c'est quoi ?**
 - Un environnement de programmation orienté objets développé par SUN et adapté à la distribution d'applications sur Internet et s'intégrant au Web. « *The network is the computer* »
- **4 éléments**
 - Un langage de programmation orienté objet
 - Une machine virtuelle (JVM) ou interpréteur
 - Des bibliothèques de classes standards
= API Application Programming Interface
(plus de 2500 classes dans java 1.4)
 - Ensemble d'outils (java, javac, jdb, javadoc, jar...)

Historique : Origines de Java

- **1990**
 - Internet très peu connu, World Wide Web inexistant
 - boom des PC (puissance)
 - Projet Oak de SUN Microsystems
 - Langage pour la communication des appareils électroniques de poche et domotique
- **1993**
 - mars : le NCSA lance MOSAIC, le premier navigateur internet (protocole http, langage html), le web décolle...
 - été : Oak change d'orientation et s'adapte à la technologie internet
- **1995**
 - mai 1995 : annonce officielle de la naissance de la technologie Java (issue de Oak)

Le langage JAVA

- **Dans un des premiers papiers sur le langage SUN décrit Java comme suit :**

« *Java : a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language* »

Le langage Java

un langage orienté-objet

- **Qu'est-ce qu'un objet ?**

- **Modélise** toute entité identifiable, concrète ou abstraite, manipulée par l'application logicielle
 - une chose tangible
ex: ville, véhicule, étudiant, un bouton sur l'écran
 - une chose conceptuelle
ex: date, réunion, planning de réservation
- **Réagit** à certains messages qu'on lui envoie de l'extérieur; la façon dont il réagit détermine le **comportement** de l'objet.
- Ne réagit pas toujours de la même façon à un même message; sa réaction dépend de **l'état dans lequel il se trouve**.

Le langage Java

un langage orienté-objet

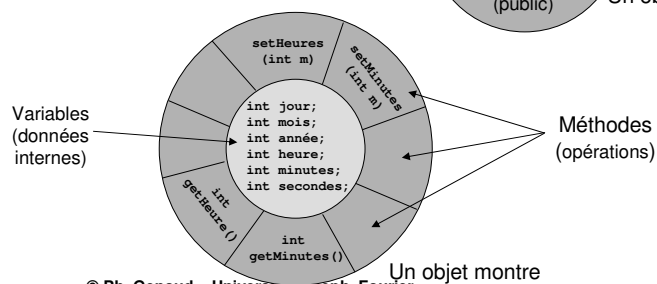
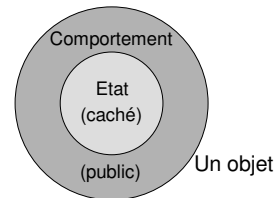
- **Un objet possède :**

- Une **identité unique** (permet de distinguer un objet d'un autre)
- Un **état interne** donné par des valeurs de **variables (ou attributs)**
 - Attributs décrivent l'état de l'objet à un instant donné
 - ex: patient mesure 1,82 m et pèse 75 Kg
 - Attributs sont typés et nommés
 - ex: `float hauteur; float poids;`
- Un **comportement** (capacités d'action de l'objet) donné par des fonctions ou sous-programmes, appelés **méthodes** (ou opérations).
 - Méthodes définissent le comportement de l'objet (ce qu'il peut faire, comment il peut le faire...) et ses réactions aux stimulations externes
 - ex: un étudiant passe un examen, etc...
 - Méthodes implémentent les algorithmes invocables sur cet objet

Le langage Java

un langage orienté-objet

- **Un objet = données + algorithmes**
 - Un objet est le regroupement de données (variables ou attributs) et des traitements (méthodes) associées
- **Principe d'encapsulation**
 - L'accès aux données (état) de l'objet ne peut être fait qu'au travers des méthodes.
 - Les données sont **privées** (cachées)
 - Les méthodes **publiques** définissent l'**interface** de de l'objet



Septembre 2004

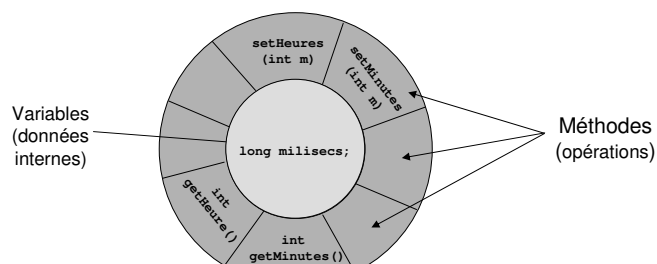
© Ph. Genoud – Université Joseph Fourier

7

Le langage Java

un langage orienté-objet

- **Intérêt de l'encapsulation**
 - Modification des structures de données n'affecte pas les programmes qui utilisent l'objet.



Septembre 2004

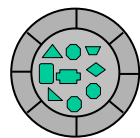
© Ph. Genoud – Université Joseph Fourier

8

Le langage Java

un langage orienté-objet

- Les objets **interagissent** et **communiquent** entre eux par l'envoi de **messages**
- Les **méthodes publiques** d'un objet correspondent aux messages que l'on peut lui envoyer
- Les messages sont caractérisés par
 - *objet cible (recepteur) du message*
 - *nom de la méthode à déclencher*
 - *paramètres de cette méthode*



objet *maMontre*

`maMontre.setHeure(14);`

Envoi d'un message à l'objet identifié par *maMontre*

`maMontre.setMinutes(45);`

Objet récepteur du message

Méthode invoquée

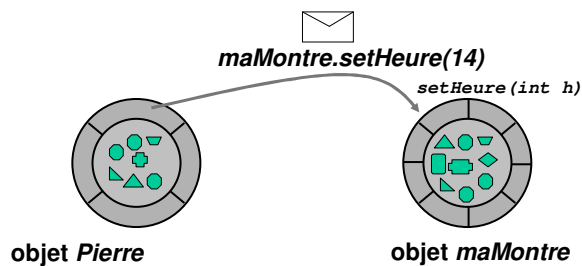
Paramètre du message

`maMontre.getMois();`

Le langage Java

un langage orienté-objet

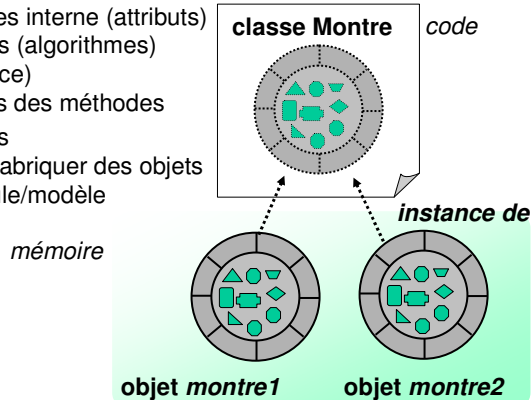
- Les objets **s'envoient des messages entre eux**



Le langage Java

un langage orienté-objet

- Les objets (instances) sont créés (**instanciés**) à partir de "moules" : les **classes**
- Classe = schéma/moule/modèle d'objets, elle décrit :
 - **partie privée**
 - structure de données interne (attributs)
 - corps des méthodes (algorithmes)
 - **partie publique** (interface)
 - noms et paramètres des méthodes
- Classe = générateur d'objets par **instanciation**, on peut fabriquer des objets obéissant à ce schéma/moule/modèle



Septembre 2004

© Ph. Genoud – Université Joseph Fourier

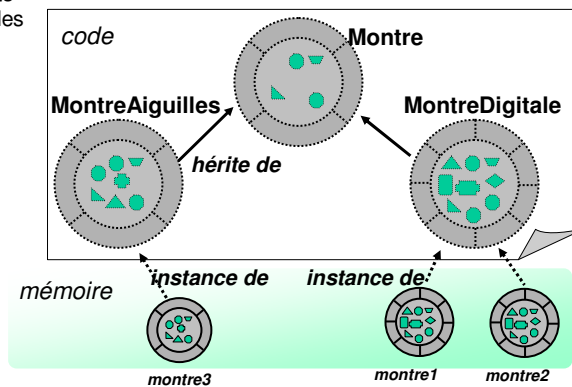
11

Le langage Java

un langage orienté-objet

- Classe = raffinement/spécialisation d'une classe existante
- Elles forment une **hiérarchie de classes**, où chaque classe :
 - **hérite** des attributs et méthodes de ses ancêtres/**super-classes**
 - ajoute de nouveaux attributs et/ou de nouvelles méthodes
 - peut modifier ou redéfinir les méthodes héritées

- **Intérêt héritage :**
 - Réutilisation du code
 - Pas besoin de réinventer la roue à chaque fois



Septembre 2004

© Ph. Genoud – Université Joseph Fourier

12

Le langage Java

un langage orienté-objet

- **Approche procédurale (C)**
 - Définir les structures de données
 - Définir les traitements
 - Analyse descendante
 - Le programme principal enchaîne les traitements.
- **Approche Objet**
 - Identifier les classes
 - Pour chaque classe
 - Définir son interface publique (signature des méthodes)
 - Définir son implémentation (attributs, corps des méthodes)
 - Le programme principal :
 - création (instanciation) d'objets en mémoire
 - lance exécution par envoi de messages aux objets créés
 - ces messages peuvent provoquer d'autres envois de messages et/ou la création d'autres objets


Septembre 2004

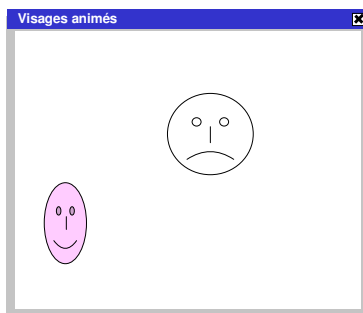
© Ph. Genoud – Université Joseph Fourier

13

Le langage Java

un langage orienté-objet

- **Exemple : les visages animés** 
 - Quels sont les concepts/les objets composant le jeu ?
 - Que doivent savoir faire ces objets ?
 - Quelles sont leurs propriétés ?



Septembre 2004

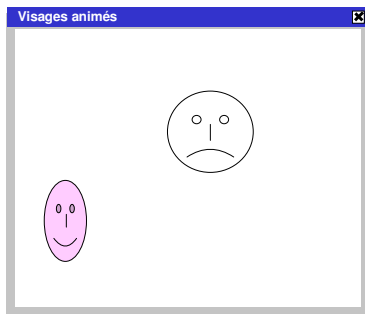
© Ph. Genoud – Université Joseph Fourier

14

Le langage Java

un langage orienté-objet

- **Exemple : les visages animés**
- Objets ?
- Capacité d'action des objets : que font-ils ?
- Attributs des objets : quelles sont leurs propriétés ?



Fenêtre

- capacités d'action
 - se fermer
 - s'icônifier
 - passer au premier plan
 - ...
- attributs
 - position
 - largeur, hauteur
 - ...

Utilisation d'une classe existante dans Java (JFrame)

Septembre 2004

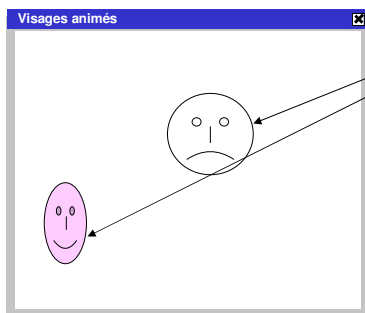
© Ph. Genoud – Université Joseph Fourier

15

Le langage Java

un langage orienté-objet

- **Exemple : les visages animés**
- Objets ?
- Capacité d'action des objets : que font-ils ?
- Attributs des objets : quelles sont leurs propriétés ?



Visages

- capacités d'action
 - sourire
 - faire la tête
 - avancer
 - changer de direction
 - s'afficher
- attributs
 - couleur
 - position
 - largeur, hauteur
 - ...

Ecriture d'une nouvelle classe (Visage)

Septembre 2004

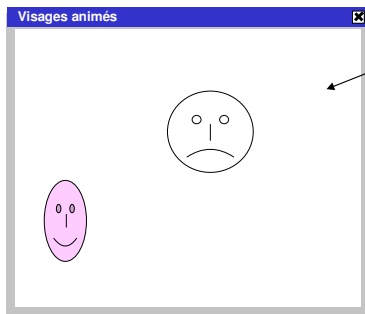
© Ph. Genoud – Université Joseph Fourier

16

Le langage Java

un langage orienté-objet

- Exemple : les visages animés
- Objets ?
- Capacité d'action des objets : que font-ils ?
- Attributs des objets : quelles sont leurs propriétés ?



Zone de dessin

- capacités d'action
 - s'effacer
 - afficher les objets qu'elle contient
 - ...
- attributs
 - fenêtre où elle se trouve
 - couleur du fond
 - ...

Écriture d'une nouvelle classe (ZoneDessin) en Réutilisant (héritage) une classe existante dans Java (JPanel)

Septembre 2004

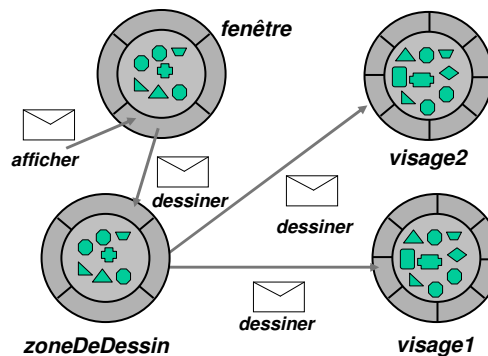
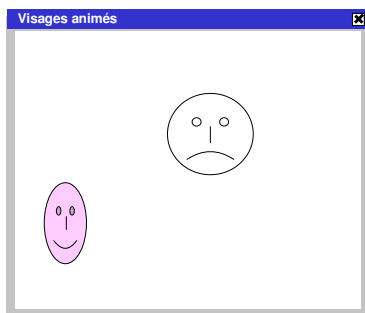
© Ph. Genoud – Université Joseph Fourier

17

Le langage Java

un langage orienté-objet

- Une application orientée objet consiste en :
 - création (instanciation) d'objets en mémoire
 - lancer exécution par envoi de messages aux objets créés
 - ces messages peuvent provoquer d'autres envois de messages et/ou la création d'autres objets



Septembre 2004

© Ph. Genoud – Université Joseph Fourier

18

Le langage Java

un langage orienté-objet : Exemple de programme Java

```
class DemoVisagesAnimes
{
    public static void main(String[] argv)
    {
        JFrame fenetre = new JFrame("Titre de la fenetre");
        // Création d'un objet de type Dessin. Cet objet est destiné à
        // contenir les objets graphiques gérés par l'application.
        Dessin dessin = new Dessin();
        // Insertion de objet de type Dessin dans la fenetre de l'appli.
        fenetre.add(dessin);
        // Affichage de la fenetre
        fenetre.show();
        // ajout, modification des objets visage
        Visage v1 = new Visage();
        Visage v2 = new Visage(200,100,200,200,5);
        dessin.ajouterObjet(v1);
        dessin.ajouterObjet(v2);
        v2.pleurer();
        dessin.repaint(10);
        while (true) {
            v1.deplacer();
            v2.deplacer();
            dessin.repaint(10);
        }
    }
}
```

Création des objets de l'application

Envoi de messages aux objets

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

19

Le langage Java

Java un langage simple (en apparence ?)

- offre toutes la fonctionnalité d'un langage de programmation puissant
- syntaxe familière (proche du C ou du C++)
- mais débarrassé des caractéristiques prêtant à confusion
 - pas de structures ni de macros (struct, typedef et #define)
 - exit les pointeurs
 - réclame un typage fort
- gestion automatique de la mémoire ("garbage collector")
 - malloc() et free() n'existent plus
- Simple à prendre en main mais pas limité,
- Mais comme pour tous les langages objets, il faut une certaine expérience avant de tirer parti au mieux de ses possibilités et surtout de son API très riche

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

20

Le langage Java

Java un langage distribué

- **au travers des classes du JDK :**
 - fournit un ensemble de classes intégrant une gestion plus ou moins transparente du réseau
 - il suffit de quelques lignes de code pour programmer :
 - récupération d'un fichier au travers du protocole HTTP ou FTP
 - un serveur de socket pouvant accepter des connexions en parallèle
 - ...
- **depuis version 1.1 : objets distribués , RMI (remote method invocation)**
 - possibilité d'envoyer des messages de manière transparente à un objet JAVA situé sur une machine distante
 - intégration future avec CORBA (OMG)

Le langage Java

Java un langage "multithreadé"

- **"thread" (processus léger)**
 - partie de code, un "flot d'instructions" s'exécutant en concurrence avec d'autres threads dans un même processus
 - cela permet à un seul programme d'effectuer plusieurs activités simultanément (programmes multitâches)
- **threads font partie intégrante du langage JAVA (jusque dans certains des mots clés)**
 - simplicité de mise en oeuvre
 - portabilité (plus directement d'appels système)
 - outils de gestion des ressources critiques (sémaphores...) et synchronisation

Le langage Java

Java un langage "multithreadé" : Exemple : l'application Visage

```
class DemoVisagesAnimes
{
    public static void main(String[] argv)
    {
        Frame fenêtre = new Frame("Titre de la fenêtre");
        // Création d'un objet de type Dessin. Cet objet est destiné à
        // contenir les objets graphiques gérés par l'application.
        Dessin dessin = new Dessin();
        // Insertion de objet de type Dessin dans la fenêtre de l'appli.
        fenêtre.add(dessin);
        // Affichage de la fenêtre
        fenêtre.show();
        // ajout, modification des objets visage
        Visage v1 = new Visage();
        Visage v2 = new Visage(200,100,200,200,5);
        dessin.ajouterObjet(v1);
        dessin.ajouterObjet(v2);
        v2.pleurer();
        while (true) {
            v1.deplacer();
            v2.deplacer();
            dessin.repaint(10);
        }
    }
}
```

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

23

Le langage Java

Un langage compilé / interprété

- **Compilation d'un programme JAVA : génération de byte-code**
- **Le byte-code est :**
 - proche d'un langage machine
 - indépendant de l'environnement d'exécution (matériel + OS)

```
// exemple pour désassembler
public class Test {
    public static void main(String[] args)
    {
        for (int i = 0; i < 10; i++)
            System.out.println("Hello " + i);
    }
}
```

code source : Test . java

javac

```
0 iconst_0
1 istore_1
2 goto 30
5 getstatic #10 <Field java.io.PrintStream out>
8 new #5 <Class java.lang.StringBuffer>
11 dup
12 ldc #1 <String "Hello ">
...
27 iinc 1 1
30 iload_1
31 bipush 10
33 if_icmplt 5
36 return
```

byte-code : Test . class

Septembre 2004

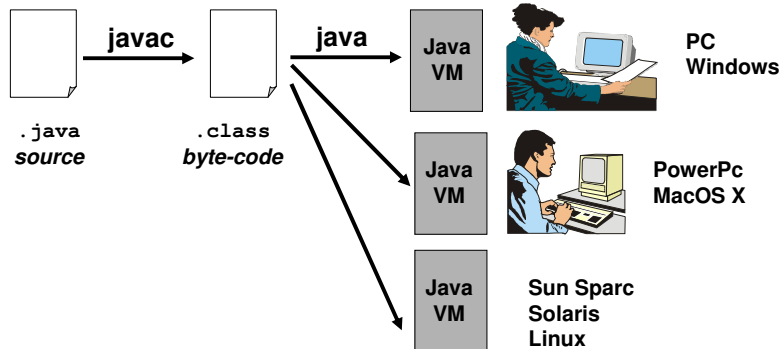
© Ph. Genoud – Université Joseph Fourier

24

La machine virtuelle Java

Exécution d'un programme Java compilé

- **byte-code assure la portabilité des programmes Java**
 - langage d'une Machine Virtuelle
 - à l'exécution un interpréteur simule cette machine virtuelle



Septembre 2004

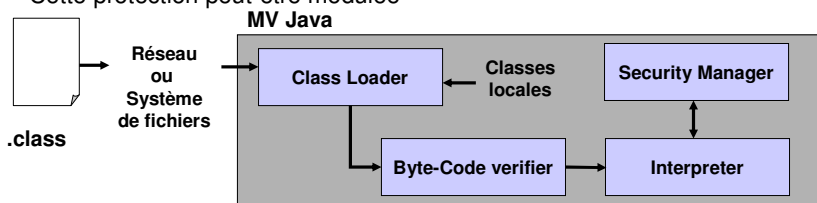
© Ph. Genoud – Université Joseph Fourier

25

La machine virtuelle java

Principes de fonctionnement

- **Chargement**
 - chargement sélectif et dynamique des classes
 - vérification statique du code (tentatives de modification de la machine virtuelle, ...)
- **Protection lors de l'exécution**
 - Le "security manager" possède un droit de veto (accès "sauvages" au système de fichiers, ...)
 - Cette protection peut-être modulée



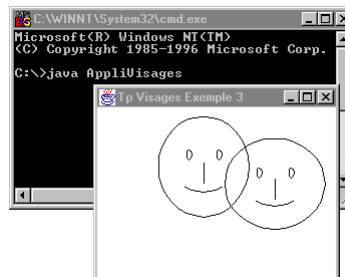
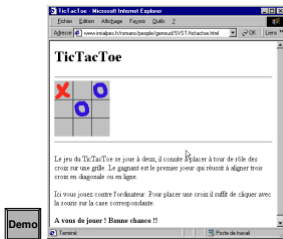
Septembre 2004

© Ph. Genoud – Université Joseph Fourier

26

Types de programmes Java

- **Les Applications indépendantes**
 - Programmes autonomes (stand-alone)



- **Les Applets**
 - Programmes exécutées dans l'environnement d'un navigateur Web et chargés au travers de pages HTML

Seuls diffèrent les contextes d'invocation et d'exécution

- Les droits des applets et des applications ne sont pas les mêmes

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

27

Types de programmes Java

Application indépendante

- **Application doit posséder une classe principale**
 - classe possédant une méthode de signature
- **Cette méthode sert de point d'entrée pour l'exécution**
 - l'exécution de l'application démarre par l'interprétation de cette méthode

ex : `java AppliVisage1`

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

28

Types de programmes Java

Applet

- Classe ne possédant pas de méthode `main()`
- Son bytecode réside sur un serveur http
- Elle est véhiculée vers un client http (navigateur Web) via une page html qui contient son url
- Lorsqu'un navigateur compatible Java (avec sa propre machine virtuelle java (JVM)) reçoit cette page HTML, il télécharge le code de la classe et l'exécute sur le poste client
 - l'applet doit posséder un certain nombre de méthodes pour permettre cette exécution
 - `init()`, `start()`, `stop()`, `paint()`, `destroy()`

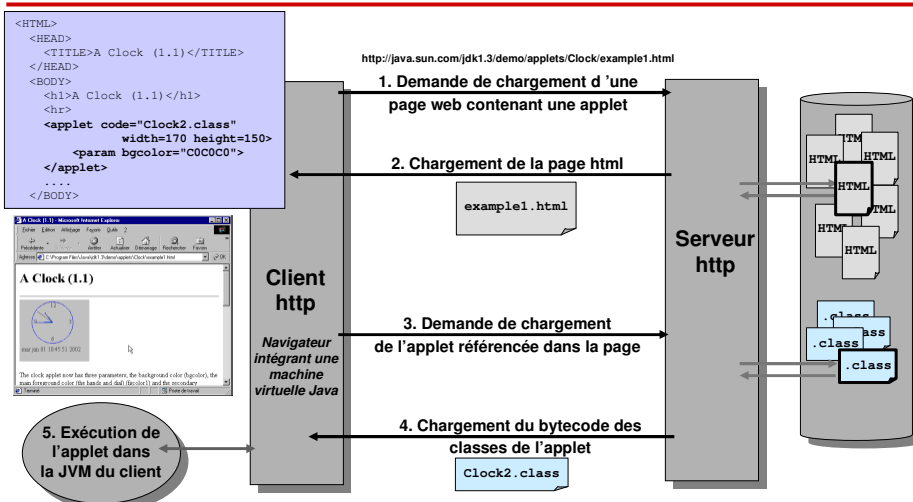
Septembre 2004

© Ph. Genoud – Université Joseph Fourier

29

Types de programmes Java

Applet : Principe de fonctionnement



Septembre 2004

© Ph. Genoud – Université Joseph Fourier

30

Performances

- **Exécution d'un programme Java**
 - le code Java est compact,
 - le chargement des classes nécessaires est sélectif et dynamique,
 - ... mais Java est interprété

- **Palier aux lenteurs de l'interprétation**
 - utilisation d'un JIT (compilateur « Just-in-Time »)
 - compilation à la volée du byte-code
 - réutilisation du code déjà compilé
 - intégration du JIT HotSpot dans JVM depuis version 1.3
 - *Performance of Java versus C++ J.P.Lewis and Ulrich Neumann*
(<http://www.idiom.com/%7Ezilla/Computer/javaCbenchmark.html>)

Les API de Java

Les packages (paquetages)

- **Package : regroupement de classes ayant un lien logique entre-elles**
 - pour les utiliser dans d'autres classes
 - pour les « spécialiser »

- **En plus d'un langage de programmation, l'environnement Java définit une API (Application Programmer's Interface) extrêmement riche au travers de nombreux packages standards.**

- **Programmer en Java nécessite une bonne connaissance de ces packages**

Les API de Java

Packages de classes standards

- **java.lang : classes essentielles**
 - objet, types de base, processus
- **java.util : structures de données (collections)**
 - listes, ensembles, hashtables, arbres, itérateurs
- **java.awt : interface graphique (Abstract Window Toolkit)**
 - fenêtres, boutons, événements...
- **java.io : entrées / sorties**
 - flot de données provenant : d'un fichier, d'un buffer, d'un "pipe »
- **java.net : réseau**
 - URL, sockets
- **java.rmi : objets distribués (Remote Method Invocation)**

Les API de Java

Packages de classes standards (suite)

- **java.sql : JDBC (Java Data Base Connectivity)**
 - connexion à une base de données relationnelle
 - envoi de requêtes SQL, récupération des résultats
- **java.beans : composants logiciels**
 - *Pièces logicielles autonomes pouvant être contrôlées dynamiquement et assemblées pour former des applications*
 - modèle de composants de SUN
 - environnement d'exécution des JavaBeans est intégré dans la plateforme Java
- **javax.swing : interface graphique**
 - composants d'interface de plus haut niveau que ceux de awt
 - look and feel indépendant de l'OS
 - exploitation du modèle MVC (Model View Controller)
-

Le JDK (SDK)

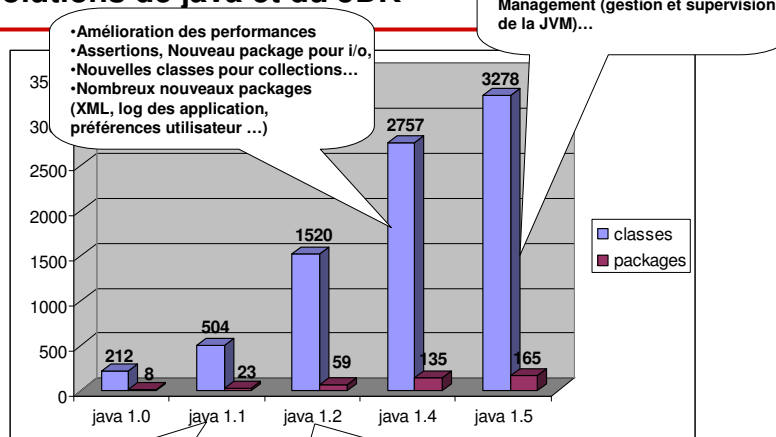
- **JDK Java Developer's Kit : environnement standard** (Windows ou Solaris) diffusé **gratuitement** par Sun (<http://java.sun.com>)
 - des outils de développement
 - compilateur , debugger, « interpréteur » de Byte-Code
 - les "bibliothèques" de classes standards
 - documentation
- **différentes évolutions du JDK correspondant à différentes évolutions de la plate-forme Java**
 - JDK1.02
 - Java 1.1 : JDK 1.1.8
 - Java 2 : décembre 1998 en fait java 1.2 (SDK 1.2.2)
 - La version actuelle Java 1.4.2 (2003)
 - La version 1.5 (Tiger) (SDK1.5.0, 30 septembre 2004)

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

35

Le JDK (SDK) Les évolutions de java et du JDK



- Modification du langage :classes internes
- Modification du modèle événementiel
- pour awt Java Beans
- Amélioration des performances de la VM

- Swings, Collections, JNDA
- Du fait de l'ajout de très nombreuses fonctionnalités la plateforme a été renommée Java 2

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

36

Les différentes éditions de Java

- 3 éditions de Java



**Standard Edition
J2SE**

Le dernier nom du jdk.
Fourni les compilateurs, outils, runtimes, et APIs pour écrire, déployer, et exécuter des applets et applications dans la langage de programmation Java



**Enterprise Edition
J2EE**

Destinée au développement d'applications « d'entreprise » («*business applications*») robustes et interopérables.
Simplifier le développement et le déploiement d'applications distribuées et articulées autour du web.



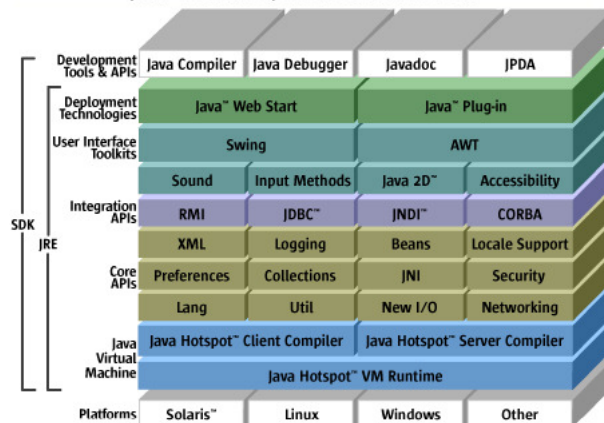
**Mobile Edition
J2ME**

Environnement d'exécution optimisé pour les dispositifs « légers » :

- Carte à puce (smart cards)
- Téléphones mobiles
- Assistants personnels (PDA)

Les différentes éditions de Java J2SE 1.4 (SDK 1.4)

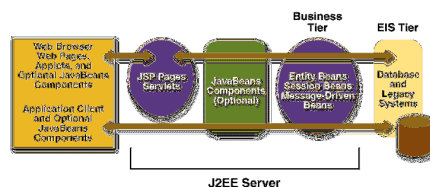
Java™ 2 Platform, Standard Edition v 1.4



Les différentes éditions de Java

J2EE Java 2 Enterprise Edition

- **Les applications d'entreprise ont besoin de délivrer des services**
 - correspondant à l'évolution actuelle du monde des affaires (globalisation)
 - protégeant la vie privée des utilisateurs et l'intégrité des données de l'entreprise
 - assurant un traitement rapide et sûr des transactions commerciales.
- **Ces services sont généralement conçus comme des applications distribuées constituées de plusieurs « parties » (tiers)**



- **La plateforme Java 2, Enterprise Edition (J2EE) a pour objectifs de réduire le coût et la complexité du développement de ces applications multi-tiers.**

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

39

Les différentes éditions de Java

J2EE Java 2 Enterprise Edition (suite)

- **La technologie J2EE**
 - Définit une plateforme standard pour le développement d'applications distribuées transactionnelles.
 - Fournit une approche à base de Composants logiciels pour la conception, le développement et le déploiement des applications d'entreprise.
 - Assure la disponibilité, la « scalabilité » et la « maintenabilité » des applications conçues sur la plateforme J2EE.
- **La plateforme J2EE**
 - Offre un modèle d'application distribuée multi-tiers.
 - Permet la réutilisation des composants logiciels
 - Fournit un support pour un échange des données basé sur XML
 - Permet contrôle flexible des transactions .

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

40

Les différentes éditions de Java

J2EE : principales technologies et API

- **Servlets Java (javax.jsdk : java server development kit)**
 - Servlets : extension d'un serveur Web à l'aide de classes Java, génération dynamique de contenu Web
- **JavaServer Pages (JSP) Technology**
 - Combinaison de documents statiques (HTML, XML) et de scripts java pour génération dynamique de contenu.
- **Enterprise JavaBeans Technology**
 - Modèles de composants pour objets métiers. *Par exemple pas besoin d'écrire de code JDBC pour accéder aux base de données, géré automatiquement par le conteneur d'EJB.*
- **Java Transaction API (JTA)**
 - Gestion des transactions
- **JavaMail**
 - Envoi et réception d'E-mail
- **Java API for XML (JAXP)**
 - Génération, lecture, manipulation de documents XML
- **Java Authentication and Authorization Service (JAAS)**
 - Gestion sécurisée de l'authentification et des autorisations pour des groupes d'utilisateurs

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

41

Les différentes éditions de Java

J2EE (situation actuelle)

- **Une spécification**
 - J2EE 1.4 (décembre 2003)
- **Une implémentation de référence** (fournie gratuitement par sun)
- **Des implémentations commerciales**
 - BEA WebLogics,
 - IBM WebSphere,
 - Sun ONE Application Server
 - Oracle ...
- **Ou open-source**
 - JBoss
 - WebObjects
 - Geronimo...

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

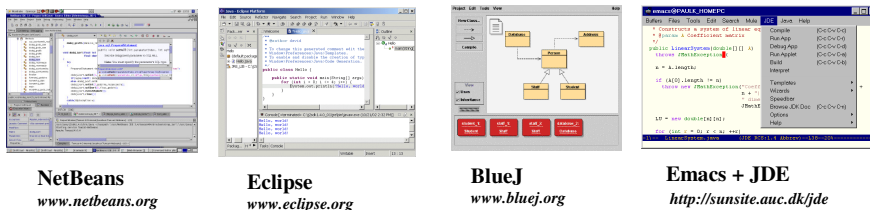
42

Environnements de développement intégrés

- **Nombreux IDE (Integrated Development Environment) commerciaux**



- **Des environnements open-source ou freeware**



NetBeans
www.netbeans.org

Eclipse
www.eclipse.org

BlueJ
www.bluej.org

Emacs + JDE
<http://sunsite.unc.dk/jde>

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

43

Situation de Java début 2004

- **Une maturité certaine**
 - Java dans de très nombreux domaines d'application des serveurs d'applications (J2EE) aux cartes à puces (J2ME)
 - Pénétration dans les entreprises (70% des Top 1000 entreprises aux USA ont des projets de développement en JAVA).
- **Evolutions de JAVA contrôlées par organisation indépendante Java Community Process (www.jcp.org)** (mais JAVA reste une marque propriétaire de SUN)
 - Évolutions du langage (généricité dans version 1.5(Tiger)
 - Nouvelles API
 - Intégration des web services dans J2EE 1.4 (release dec 2004)
- **Amélioration des performances**
 - JIT, Hotspot
- **La bataille avec .NET est engagée**

Septembre 2004

© Ph. Genoud – Université Joseph Fourier

44

Références

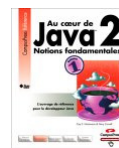
« Java - 1 Premières applications professionnelles en Java »
Emmanuel Puybaret, Eyrolles, mai 2003



« Introduction à Java », 2e édition
Pat Niemeyer et Jonathan Knudsen, O'Reilly, déc. 2002



« Au coeur de Java 2 - Volume 1 : Notions fondamentales »
Cay S. Horstmann, Gary Cornell, CampusPress, nov. 2003



« Thinking in Java »,
Bruce Eckel - Prentice-Hall (www.BruceEckel.com , www.penserenjava.free)

« JAVA in a nutshell, 4th Edition »,
David Flanagan - O'Reilly 2002

...

Références (suite)

URLs

- <http://java.sun.com> - Site officiel Java de SUN
 - JDK, Tutorials, Documentations, spécifications, ...
- <http://www.javaworld.com>
Magazine électronique
- <http://www.jguru.com>, <http://www.jdance.com> –
Sites dédiés à la technologie java
 - applets, applications, notes techniques, forums de discussions
- ...
- <http://cui.unige.ch/java/>
Le Coin Java Centre Universitaire d'Informatique (CUI) de l'Université de Genève
- www.developpez.com
des FAQ, des ressources....
- <http://penserenjava.free.fr>
une traduction du livre de BruceEckel « Thinking in Java »
- ...