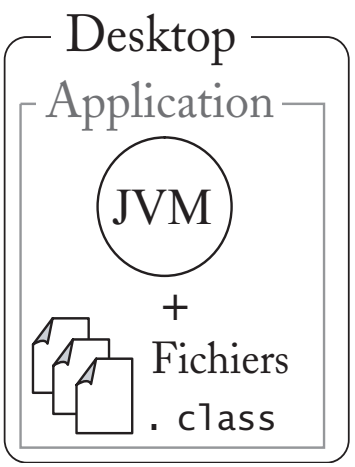
Principes du langage et installation de l'environnement

2



Java intègre les concepts les plus intéressants des technologies informatiques récentes dans une plate-forme de développement riche et homogène. L'approche objet de ce langage, mais aussi sa portabilité et sa gratuité, en font un des outils de programmation idéaux pour s'initier à la programmation objet.

SOMMAIRE

MOTS-CLÉS

- Objets et classes
- JVM
- JDK
- javadoc

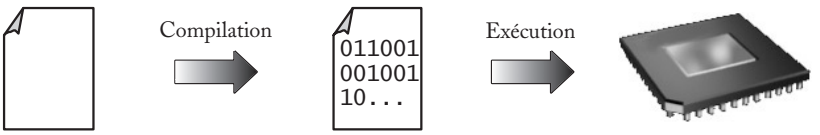
Programmer en Java : une démarche objet

Du binaire à l'objet, 50 ans d'évolution de la programmation

La programmation identifie les données d'une information et les traitements qui s'y appliquent puis les codifie pour les rendre compréhensibles par un ordinateur. Le microprocesseur d'un ordinateur ne manipulant que des instructions et des données codées en binaire, différents langages de programmation ont été créés pour permettre aux programmeurs de coder des concepts plus humains que des 0 et des 1. Le texte d'un tel programme est traduit par un compilateur ou un interpréteur en instructions que le microprocesseur peut alors exécuter.

B.A.-BA **Vocabulaire de la programmation objet**

L'une des difficultés de la programmation en Java passe par l'utilisation des nombreux termes associés aux concepts de la programmation objet. Ces termes, décrits au fur et à mesure de cet ouvrage, sont repris dans le glossaire en annexe si vous voulez vous rafraîchir la mémoire en cas de besoin.



Fichier contenant le texte d'un programme respectant la syntaxe d'un langage de programmation

Fichier exécutable contenant des instructions binaires du microprocesseur

Figure 2–1
Compilation et exécution d'un programme

REGARD DU DÉVELOPPEUR **Les atouts de Java**

Mis au point par Sun Microsystems, Java est un langage de programmation utilisé dans de nombreux domaines.
Son succès est dû à un ensemble de caractéristiques dont voici un aperçu :

- Langage de programmation objet et fortement typé : contraignants pendant le développement, l'approche objet et le typage fort du langage Java rendent plus robuste un programme Java dès sa conception.
- Syntaxe proche du C et C++ : en reprenant une grande partie de la syntaxe de ces deux langages, Java facilite la formation initiale des programmeurs qui les connaissent déjà.
- Gestion de la mémoire simplifiée : le ramasse-miettes (*garbage collector* en anglais) intégré à Java détecte automatiquement les objets inutilisés pour libérer la mémoire qu'ils occupent.
- Gestion des exceptions : Java intègre la gestion des exceptions autant pour faciliter la mise au point des programmes (détection et localisation des bogues) que pour rendre un programme plus robuste.
- Multitâche : grâce aux threads, Java permet de programmer l'exécution simultanée de plusieurs traitements et la synchronisation des traitements qui partagent des informations.
- Système de sécurité : Java protège les informations sensibles de l'utilisateur et le système d'exploitation de sa machine en empêchant l'exécution des programmes conçus de façon malintentionnée (contre un virus par exemple).
- Bibliothèque très riche : la bibliothèque fournie en standard avec Java couvre de nombreux domaines (gestion de collections, accès aux bases de données, interface utilisateur graphique, accès aux fichiers et au réseau, utilisation d'objets distribués, XML…, sans compter toutes les extensions qui s'intègrent sans difficulté à Java !)
- Exécutable portable : comme l'exprime l'accroche *Write Once Run Anywhere*, un programme Java, une fois écrit et compilé, peut être exécuté sans modification sur tout système qui prend en charge Java.
- Gratuit : développement gratuit avec les commandes de bases Java, ou certains outils plus évolués, et exécution gratuite des programmes.

Les langages de programmation ont évolué pour permettre aux programmeurs d'utiliser des concepts de plus en plus proches de la réalité et du langage naturel. La programmation en assembleur a remplacé le codage en binaire des données par un codage en hexadécimal, et les instructions codées en binaire du microprocesseur par des instructions symboliques.

Exemple

```
MOVE 02 R1
MOVE 10 R2
ADD R1 R2
```

Ces instructions écrites en assembleur Motorola 68000 placent la valeur `2` dans le registre `R1`, la valeur `16` (`10` en hexadécimal) dans le registre `R2`, puis additionne les valeurs de ces deux registres.

La programmation procédurale et structurée de langages comme le C, le Pascal…, identifie les groupes logiques de données et les procédures décrivant des suites cohérentes d'instructions.

Exemple

Voici la trame d'un programme écrit en C qui pourrait être utilisé sur un téléphone `portable` pour l'`allumer`. Ce `portable` a ici pour données sa `carte SIM` et l'état de sa `connexion`.

```
typedef struct
{
  char * carteSIM;
  char   connexion;
}  Portable;

void allumer (Portable * telephone)
{
  /* Instructions C à exécuter au cours de la mise en marche */
}
```

La programmation orientée objet regroupe les groupes de données et les traitements qui s'y appliquent sous forme d'entités nommées *objets*. À un objet physique avec son état et son comportement correspond un objet informatique avec ses données et ses traitements. La programmation objet est aussi utilisée pour des concepts abstraits, par exemple la gestion de comptes bancaires.

Le traitement d'un objet est programmé sous la forme d'un message.

Exemple

Un téléphone portable peut être représenté sous la forme d'un objet doté des messages suivants :

```
allumer
eteindre
appeler(numero)
```

Le dernier message, `appeler`, est paramétrable. Il prend en paramètre un numéro de téléphone.

Assembleur et langage d'assemblage

On appelle assembleur le programme qui transforme en code binaire ou en exécutable un programme écrit en langage d'assemblage. Le langage d'assemblage se compose de mnémoniques (plus lisibles que le code binaire) représentant les instructions binaires d'un microprocesseur.
Un programme directement écrit en langage d'assemblage exploite de façon optimale les capacités du microprocesseur mais n'est pas portable d'une puce à l'autre, chaque famille de microprocesseurs (Intel x86, PowerPC,...) ayant son propre jeu d'instructions.

B.A.-BA Hexadécimal

En hexadécimal ou base 16, les *nombres* décimaux 10, 11, 12, 13, 14 et 15 sont représentés par les *chiffres* hexadécimaux A, B, C, D, E et F. La notation hexadécimale continue à être souvent utilisée en informatique pour manipuler les informations binaires des images ou de sons digitalisées, car elle est pratique pour noter chaque groupe de 4 bits ou chiffres binaires sous forme d'un seul chiffre hexadécimal. En voici quelques exemples :

- 8 en décimal = 8 en hexa = 1000 en binaire ;
- 20 en décimal = 14 en hexa = 1 0100 en binaire ;
- 255 en décimal = FF en hexa = 1111 1111 en binaire ;
- 1 024 en décimal = 400 en hexa = 100 0000 0000 en binaire.

À RETENIR

Appeler un traitement d'un objet, c'est envoyer un message à cet objet.

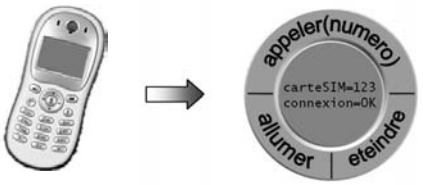


Figure 2–2 L'objet `Téléphone portable` et ses messages

Ce que fait un objet et comment il le fait… interface et implémentation

Un objet est une boîte noire, munie d'une interface et de son implémentation. L'interface spécifie la liste des messages disponibles pour un objet donné, tandis que l'implémentation correspond à la programmation proprement dite des messages de l'objet avec ses données et ses traitements.

On pourra souvent considérer que l'interface est la liste des services proposés par l'objet, et l'implémentation la manière de réaliser ces services. Quand un objet reçoit un message disponible dans son interface, les traitements implémentés par ce message sont exécutés.

Un message reçu par un objet provoque souvent une réaction en chaîne. Par exemple, le message `clic` envoyé au bouton `OK` d'une boîte de dialogue enverra le message `fermer` à la boîte de dialogue puis provoquera une action qui correspondra au choix proposé.

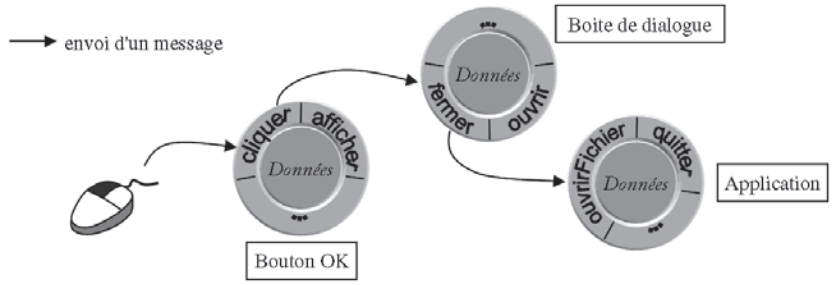


Figure 2–3 Ensemble d'objets d'un programme communiquant par messages

Programme

Dans un programme objet, les objets sont mis en relation et communiquent entre eux par messages.

À chaque métier ses objets

La liste des messages de l'interface d'un objet est fixée en fonction des besoins du programme où cet objet sera utilisé. Selon le type d'application, l'analyse des besoins peut aboutir à une interface différente pour un même objet.
Par exemple, un téléphone portable pourrait être doté d'une interface objet avec les messages suivants :

- pour le programme du téléphone : `allumer eteindre appeler(numero)`
- pour l'application de l'exploitant du réseau : `joindre getId`
- pour le programme de gestion du fabricant du téléphone : `getNumeroSerie getPrix getDescriptif`

DANS LA VRAIE VIE Penser objet, une démarche qui demande de l'expérience

Bien que basée sur un concept simple, la maîtrise de la programmation orientée objet et du mode d'analyse qui lui est associé ne va pas sans pratique et demande donc que l'on y consacre du temps. Voici les principales difficultés que vous aurez à surmonter :

- L'identification des objets, pour un problème donné, requiert un niveau d'abstraction élevé.
- Réfléchir à l'interface et aux messages des objets avant d'étudier leur implémentation n'est pas une démarche si naturelle qu'il n'y paraît et demande un bon esprit d'analyse.
- Le découpage d'un problème en objets qui soient le plus indépendants possible les uns des autres permet d'obtenir un programme plus simple à maintenir et des objets que l'on va pouvoir réutiliser dans plusieurs programmes. Cette démarche gagnante sur le long terme demande plus de temps d'analyse au départ.
- Dans un programme où les objets sont mis en relation, les liens que l'on crée entre eux doivent être clairs et limités pour éviter une interdépendance trop complexe entre les objets.
- Quand un message met en œuvre plusieurs objets, la décision d'ajouter le message à un objet plutôt qu'à un autre n'est pas toujours évidente à prendre.

De l'analyse objet à l'écriture des classes Java

Pendant la phase de conception des objets, on essaiera d'identifier des catégories d'objets ayant les mêmes messages et les mêmes types de données (par exemple, tous les téléphones portables, tous les boutons d'une boîte de dialogue).

Plutôt que de programmer individuellement chaque objet avec ses messages et ses données, un développeur Java programme un modèle, ou *classe*, pour chaque catégorie d'objets et crée les objets à partir de leur modèle. Chaque classe implémente les messages et les types de données d'une catégorie d'objets. En fait, tout objet est créé à partir d'une classe (on dit aussi qu'un objet est une instance d'une classe) ; même un objet doté de messages et de types de données uniques est une instance unique d'une classe.

Le concept de classe est très important puisqu'en Java tout se programme à l'intérieur des classes.

Exemple

Un téléphone portable connecté à un réseau pourrait être représenté par les objets et classes suivants :

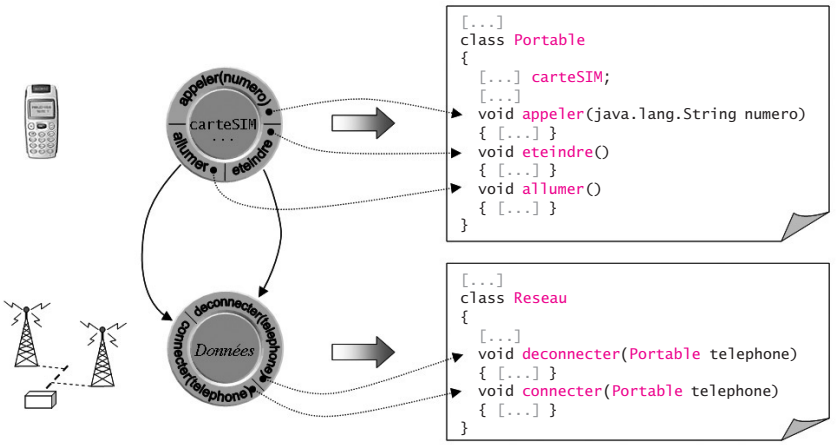


Figure 2–4
Identification des classes correspondant aux objets portable et réseau

À RETENIR Terminologie

Identifier une catégorie d'objets (mêmes messages, mêmes types de données), c'est identifier une classe avec ses membres (méthodes et champs).

Écriture, compilation, exécution

De la conception à l'exécution d'un programme Java, on compte trois phases :

1. Écriture des classes dans des fichiers portant une extension `.java`.
2. Compilation des fichiers `.java` avec la commande `javac`. Le compilateur crée pour chaque classe un fichier d'extension `.class` contenant du code binaire (*bytecode*) spécifique à Java. Un fichier `.class` décrit une classe, ses champs, ses méthodes et les instructions des méthodes.

À RETENIR

Programmer en Java, c'est donc :

- écrire les classes du programme, leurs méthodes et leurs champs ;
- instancier les classes (créer les objets du programme) ;
- appeler les méthodes de ces objets (leur envoyer des messages).

C++ Pas de variables ou de fonctions globales en Java

La structure d'un fichier `.java` est très simple car il n'est permis de définir, au niveau global d'un fichier, que des classes, des interfaces (sortes de classes dont toutes les méthodes sont virtuelles pures) ou des énumérations (disponibles uniquement à partir de Java 5.0). Il n'existe pas en Java de notion de constante globale, de variable globale, de fonction globale, de macro, de structure, d'union ou de synonyme de type : `#define`, `struct`, `union` et `typedef` n'existent pas en Java. Les classes Java n'ont même pas besoin d'être déclarées dans des fichiers `header` séparés pour les utiliser dans d'autres fichiers sources !

C++ Pas d'édition de liens en Java

Il n'y a pas de phase d'édition de liens en Java ; chaque classe d'un fichier `.class` peut être vue comme une petite DLL (*Dynamically Linked Library*) dynamiquement chargée à l'exécution par la JVM, la première fois qu'elle est utilisée.

C# Équivalent bytecode/JVM

Le *bytecode* Java est l'équivalent du MSIL C# et la machine virtuelle Java l'équivalent du CLR C# (*Common Language Run time*).

3 Lancement de la machine virtuelle Java (JVM, pour Java Virtual Machine). La JVM charge les fichiers `.class` nécessaires à l'exécution du programme et interprète le code binaire des instructions des méthodes en instructions du microprocesseur de la machine sur laquelle tourne le programme.

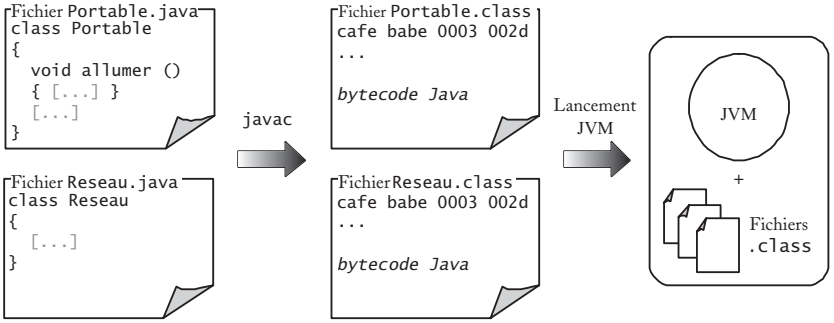


Figure 2–5 Cycle de développement Java

À chaque besoin son environnement Java : applets, servlets, applications

Les trois principaux environnements d'exécution Java (*frameworks* en anglais) sont les applications 1, les applets 2 et les servlets 3. Chaque environnement utilise une catégorie de classe et un point d'entrée différents ; le point d'entrée d'un programme est la méthode appelée par la JVM pour exécuter un programme.

1 Application batch ou interface homme-machine lancée avec la commande java

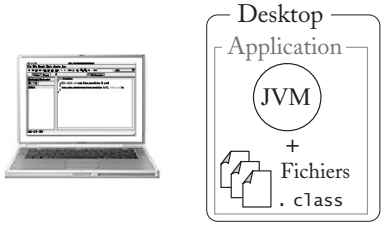


Figure 2–6 Application Java

Une application s'exécute sur une machine isolée ou raccordée à un réseau.
La JVM et les fichiers `.class` d'une application doivent être installés sur la machine.
Le point d'entrée d'une application est la méthode `main` d'une classe respectant la syntaxe décrite ci-après.

```
class Editeur
{
  public static void main(java.lang.String [] args)
  {
    // Votre programme
  }
}
```

2 Applet d'un fichier HTML lancée par un navigateur

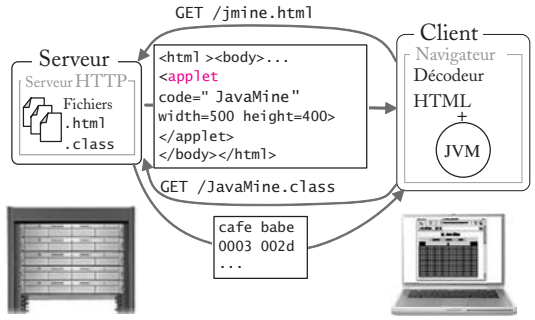


Figure 2–7 Applet Java

Une applet s'exécute dans une page HTML sur une machine cliente raccordée à un serveur Web.
La JVM, installée sur la machine cliente, est lancée par le navigateur.
Les fichiers `.class` d'une applet sont installés sur le serveur Web et téléchargés par le navigateur.
Le point d'entrée d'une applet est la méthode `init` d'une classe respectant la syntaxe décrite ci-après.

```
public class JavaMine extends javax.swing.JApplet
{
  public void init ()
  {
    // Votre programme
  }
}
```

3 Servlet lancée par une requête sur un serveur Web

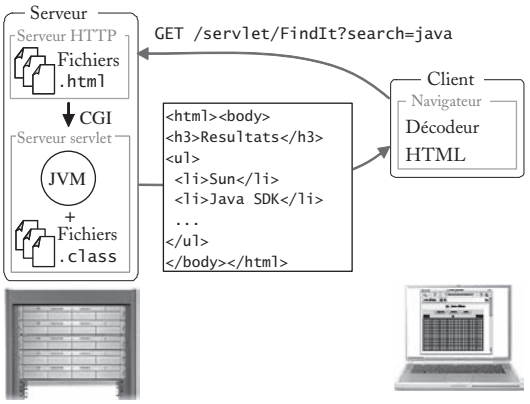


Figure 2–8 Servlet Java

Une servlet s'exécute sur un serveur Web pour générer dynamiquement des pages HTML ou des images.
La JVM et les fichiers `.class` d'une servlet doivent être installés sur le serveur Web.
Le point d'entrée d'une servlet est la méthode `doGet` d'une classe respectant la syntaxe décrite ci-après.

```
public class FindIt extends javax.servlet.http.HttpServlet
{
  public void doGet
     (javax.servlet.http.HttpServletRequest  request,
      javax.servlet.http.HttpServletResponse response)
     throws
      javax.servlet.ServletException, java.io.IOException
  {
    // Votre programme
  }
}
```

B.A.-BA Machine virtuelle Java (JVM)

L'architecture d'exécution Java permet d'exécuter les instructions Java d'un fichier `.class` sur n'importe quelle machine avec la machine virtuelle (JVM) qui correspond à son système d'exploitation et son microprocesseur. La JVM Windows, par exemple, traduit les instructions Java en instructions Intel, la JVM Mac OS traduit les instructions Java en instructions PowerPC, etc.

VERSIONS **Un mode de téléchargement enfin moderne**

Depuis la version 1.4.2, Java dispose d'outils modernes d'installation et de mise à jour. Il est maintenant possible soit de télécharger le JDK (ou le JRE) en un seul coup pour une installation *off line*, soit de télécharger un logiciel de quelques centaines de Ko dont le rôle est de télécharger le reste du JDK (ou du JRE) avant de l'installer. Une fois installée, la JVM est capable de se mettre à jour d'elle-même si une nouvelle version de Java est disponible sur le site de Sun Microsystems (sous Mac OS X, utilisez le module Mise à jour de logiciels du système).

Télécharger et installer les programmes pour développer en Java

Les versions de Java pour les systèmes Windows, Solaris et Linux sont disponibles sur le site Internet de Sun Microsystems à http://java.sun.com. Sun fournit chaque version de Java sous deux formes :

- L'une pour les développeurs : le JDK (Java Development Kit) ou SDK (Software Development Kit) comprenant la machine virtuelle Java pour un système d'exploitation, la bibliothèque des classes Java et les commandes pour développer en Java.
- L'autre pour les utilisateurs : le JRE (Java Runtime Environment) comprenant la machine virtuelle Java et la bibliothèque des classes.

Figure 2–9
Page d'accueil
http://java.sun.com

Voici les instructions qu'il faut suivre pour installer le JDK fourni par Sun.

1. Téléchargez la version la plus récente du JDK. Ce fichier de plus de 40 Mo a un nom de la forme `jdk-`*`VERSION`*`-`*`OS`*`.ext`, où *`VERSION`* représente une suite de chiffres séparés par des points (par exemple `1.5.0_06`), et *`OS`* le système de destination du JDK (par exemple `windows-i586`).
2. Installez le JDK et ajoutez au `PATH` de votre système le chemin d'accès au sous-dossier `bin` du JDK contenant les commandes Java comme cela est précisé ci-après.

VERSIONS **Les versions de Java depuis 1995**

Depuis sa première version en 1995, Sun Microsystems sort une version majeure de Java tous les 18 mois précédée de versions beta et *pre release* (*Release Candidate*) publiques. Chaque nouvelle version ajoute des fonctionnalités grâce à de nouvelles classes et améliore la vitesse d'exécution de Java :

- 1995 : 1.0 (170 classes)
- 1997 : 1.1 (391 classes)
- 1998 : 1.2 (1462 classes)
- 2000 : 1.3 (1732 classes)
- 2002 : 1.4 (2367 classes)
- 2004 : 1.5 ou 5.0 (2800 classes)

Java respecte la compatibilité ascendante des classes, autorisant le fonctionnement des anciennes classes avec les versions les plus récentes.

Depuis la version 1.2, la version standard de Java est dénommée J2SE (Java 2 Standard Edition) et la technologie Java s'est décomposée en trois éditions différentes :

- J2SE (Java 2 Standard Edition) est destinée au marché des ordinateurs de bureau (*desktop*).
- J2EE (Java 2 Enterprise Edition) a une bibliothèque de classes plus riche et est destinée au marché des serveurs d'entreprise prenant en charge les EJB (Enterprise JavaBeans).
- J2ME (Java 2 Micro Edition) est une version allégée de Java qui n'est pas compatible avec J2SE et est dédiée au marché des téléphones portables, PDA, cartes de crédits…

Installation sous Windows 95/98/ME, NT, 2000/XP

Sous Windows 95/98/ME

1. Exécutez le fichier d'installation `jdk-VERSION-OS.exe` et installez le JDK dans le dossier proposé `C:\Program Files\Java\jdkVERSION`.
2. Éditez le fichier `C:\AUTOEXEC.BAT` et ajoutez-y la ligne :

```
PATH=%PATH%;"C:\Program Files\Java\jdkVERSION\bin"
```

3. Redémarrez votre machine.

Sous Windows NT

1. Exécutez le fichier d'installation `jdk-VERSION-OS.exe` et installez le JDK dans le dossier proposé `C:\Program Files\Java\jdkVERSION`.
2. Cliquez avec le bouton droit de la souris sur l'icône de votre poste de travail et choisissez le menu Propriétés.
3. Choisissez l'onglet Environnement dans la boîte de dialogue des Propriétés.
4. Ajoutez la variable d'environnement `PATH` avec la valeur :

```
%PATH%;C:\Program Files\Java\jdkVERSION\bin
```

Si la variable `PATH` existe déjà, modifiez-la en ajoutant à la fin de sa valeur :

```
;C:\Program Files\Java\jdkVERSION\bin
```

5. Confirmez votre saisie en cliquant sur Modifier et fermez la boîte de dialogue.

VERSIONS **Java 6.0**

Contrairement à la version 5.0 du J2SE qui a entre autres choses, enrichi le langage Java de nouveaux éléments syntaxiques détaillés dans cet ouvrage, la version 6.0 du J2SE de nom de code Mustang n'apportera que des modifications à la bibliothèque standard Java. Jusqu'à sa finalisation prévue pour le courant 2006, le site suivant lui est dédié :

- https://mustang.dev.java.net/

VERSIONS **Sous Windows 95, Java 1.4.0**

Sun ne proposant plus d'assistance technique pour Java sous Windows 95, seule une ancienne version de Java 1.4 est disponible en archive à l'adresse suivante :

- http://java.sun.com/products/archive/j2se/1.4.0_04/

ASTUCE **Utilisation de DOSKEY**

Si elle n'y figure pas déjà, ajoutez aussi à votre fichier AUTOEXEC.BAT la ligne :

DOSKEY

Cette fonctionnalité, disponible d'office sous Windows NT/2000/XP, Linux et Mac OS X, permet au système de mémoriser les commandes récentes. Dans une fenêtre de commandes, vous pouvez faire défiler ces commandes avec les flèches haut et bas.

B.A.-BA **PATH**

Bien qu'il ne soit pas obligatoire de modifier le PATH pour faire fonctionner Java, il vous est conseillé de respecter les instructions ci-contre pour simplifier l'utilisation des commandes Java. En effet, la variable d'environnement PATH décrit la liste des dossiers parmi lesquels votre système va chercher un programme pour l'exécuter en ligne de commande quand vous ne donnez pas le chemin pour accéder à ce programme. Ceci vous permettra par exemple de lancer le compilateur Java avec la commande `javac` au lieu de `C:\Program Files\Java\jdkVERSION\bin\javac`.

B.A.-BA Environnements de développement intégrés (IDE) Java

Dédiés au développement d'un programme Java, les IDE (*Integrated Development Environment*) Java simplifient grandement l'édition et la gestion d'un programme. Ils intègrent pour la plupart les fonctionnalités suivantes :

- Éditeur de textes avec mise en couleur des mots-clés Java, des commentaires, des valeurs littérales...
- Complétion automatique (menus contextuels proposant la liste des méthodes d'un objet).
- Génération automatique des dossiers nécessaires à l'organisation d'un programme et des paquetages des classes.
- Intégration des commandes Java et de leurs options dans des menus et des boîtes de dialogue appropriés.
- Débogueur pour exécuter pas à pas un programme en phase de mise au point.
- Gestion de versions avec CVS ou d'autres outils.

Les IDE les plus importants du marché et fonctionnant sur tous les systèmes d'exploitation :

- Borland JBuilder http://www.borland.fr/jbuilder/
- Eclipse http://www.eclipse.org/
- IntelliJ IDEA http://www.intellij.com/idea/
- NetBeans http://www.netbeans.org/
- Oracle JDeveloper 10g http://otn.oracle.com/products/jdev
- Sun Java Studio http://wwws.sun.com/software/sundev/jde/

Voir aussi en annexe une description de JBuilder et d'Eclipse fournis sur le CD-Rom qui accompagne cet ouvrage.

Sous Windows 2000/XP

1. Exécutez le fichier d'installation `jdk-VERSION-OS.exe` et installez le JDK dans le dossier proposé `C:\Program Files\Java\jdkVERSION`.
2. Cliquez avec le bouton droit de la souris sur l'icône de votre poste de travail et choisissez le menu Propriétés.
3. Choisissez l'onglet Avancé dans la boîte de dialogue des Propriétés.
4. Cliquez sur le bouton Variables d'environnement.
5. Ajoutez la variable d'environnement `PATH` avec la valeur :

   ```
   %PATH%;C:\Program Files\Java\jdkVERSION\bin
   ```

   Si la variable `PATH` existe déjà, modifiez-la en ajoutant à la fin de sa valeur :

   ```
   ;C:\Program Files\Java\jdkVERSION\bin
   ```

6. Confirmez votre saisie et fermez la boîte de dialogue.

Installation sous Linux

1. Ouvrez une fenêtre de terminal.
2. Déplacez-vous avec la commande `cd` dans le dossier où vous voulez installer le JDK.
3. Rendez le fichier `jdk-VERSION-OS.bin` exécutable avec la commande :

   ```
   chmod +x jdk-VERSION-OS.bin
   ```

4. Exécutez le fichier d'installation `jdk-VERSION-OS.bin`.
5. Éditez le fichier `~/.bashrc` et ajoutez-y les lignes :

   ```
   PATH=$PATH:/chemin/vers/jdkVERSION/bin
   export PATH
   ```

6. Redémarrez votre session.

Installation sous Mac OS X

Le JDK est préinstallé sous Mac OS X et les commandes Java sont disponibles dans l'application Terminal sans avoir à modifier le PATH. Les mises à jour éventuelles de Java s'installent simplement avec le module Mise à jour de logiciels... du menu Pomme, mais la version maximale de Java que vous aurez à disposition varie en fonction de la version de Mac OS X de votre machine : sous Mac OS 10.4, la version la plus récente est Java 5.0, sous Mac OS 10.3, Java 1.4.2, sous Mac OS 10.2, Java 1.4.1.

Pour les autres systèmes, consultez le site de leurs éditeurs respectifs.

VERSIONS Sous Mac OS 9, le JDK 1.1.8

Le JDK 1.1.8 est la version la plus récente disponible pour Mac OS 9. Apple a préféré concentrer ses efforts de développement pour Java exclusivement sur Mac OS X.

Télécharger, installer et utiliser la documentation

La documentation des API Java (Application Programming Interface) décrit les fonctionnalités des classes de la bibliothèque Java. Elle se présente sous forme de fichiers HTML dont l'organisation permet de retrouver rapidement la description d'une classe et de ses méthodes grâce à de nombreux liens hypertextes. Cette documentation indispensable peut être consultée en ligne à http://java.sun.com/j2se/1.5/docs/api ou téléchargée pour la consulter hors connexion.

Voici comment installer la documentation du J2SE, qui décrit les API Java entre autres choses. Téléchargez la documentation qui correspond à la version de votre JDK. Ce fichier de plus de 30 Mo a un nom de la forme `jdk-`*`VERSION`*`-doc.zip`, *`VERSION`* représentant une suite de chiffres séparés par des points (par exemple `1.5.0`).

Décompressez le fichier avec l'outil de votre choix ou utilisez la procédure suivante pour décompresser la documentation :

1. Ouvrez une fenêtre de commandes (sous Mac OS X, démarrez l'application Terminal du dossier Applications/Utilitaires).
2. Déplacez-vous avec la commande `cd` dans le dossier d'installation du JDK ou dans un autre.
3. Décompressez le fichier de documentation `jdk-`*`VERSION`*`-doc.zip` en exécutant la commande :

```
jar xf /chemin/vers/jdk-VERSION-doc.zip
```

La documentation du J2SE est décompressée dans le sous-dossier `docs`.

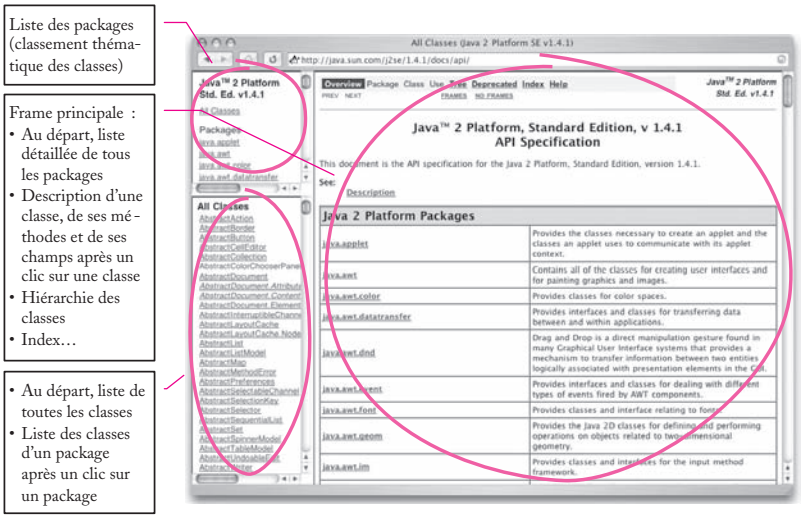


Figure 2–10 Documentation des API Java

REGARD DU DÉVELOPPEUR

Ne vous laissez pas impressionner !

Ne vous inquiétez pas devant la quantité d'informations que vous présente la documentation des API Java. Utilisez-la surtout comme outil de référence sur les classes Java et leurs méthodes.
La documentation des API montrée par la capture d'écran de la figure 2-10 s'obtient en cliquant successivement sur les liens API & Language Documentation puis Java 2 Platform API Specification de la page `index.html` du dossier d'installation de la documentation du J2SE. Ajoutez la page des API Java tout de suite à vos favoris/signets/*bookmarks* et apprenez à vous en servir car vous en aurez souvent besoin.

ATTENTION **Mise à jour du PATH**

Si le système vous indique que la commande `jar` est inconnue, vérifiez que le PATH a été correctement modifié en exécutant la commande :

- Sous Windows : `PATH`
- Sous Linux : `echo $PATH`

Le texte affiché doit refléter les modifications opérées sur le PATH dans le point précédent.
Pour que toute modification du PATH soit prise en compte dans une fenêtre de commande, il vous faut :

- sous Windows 95/98/ME, redémarrer votre machine,
- sous Windows NT/2000/XP, ouvrir une nouvelle fenêtre de commande,
- sous Linux, exécuter la commande : `source ~/.bashrc`

POUR ALLER PLUS LOIN **Autres documentations**

Parmi les nombreuses documentations en anglais fournies par Sun Microsystems sur son site, notez bien les deux suivantes à ne pas manquer :

- *Java Tutorial* : cours sur Java très complet et régulièrement mis à jour.
- *Java Language Specification* : spécifications détaillées du langage.

JAVA **Commandes du JDK les plus utilisées**

Voici un aperçu des commandes du JDK qui sont le plus utilisées. Ces commandes se lancent dans une fenêtre de commandes ou un terminal Unix :

- **javac** : le compilateur Java vérifie la syntaxe du (des) fichier(s) `.java` passé(s) en paramètres et génère un fichier `.class` pour chacune des classes qu'ils contiennent.
- **java** : cette commande lance la machine virtuelle Java qui charge la classe passée en paramètre puis appelle sa méthode `main`.
- **appletviewer** : cette commande lit le fichier HTML passé en paramètre puis affiche dans une fenêtre l'applet de chaque balise `<applet>` de ce fichier.
- **jar** : cette commande crée et lit des archives au format ZIP.
- **javadoc** : cette commande génère la documentation au format HTML d'un ensemble de classes à partir de leurs commentaires au format javadoc. C'est avec cet outil que la documentation des API Java est créée.

Chaque commande Java propose un ensemble d'options repérable au tiret (-) qui les précède. La liste de ces options s'obtient en tapant une commande Java seule dans une fenêtre de commandes ou en cliquant sur le lien Tool Docs de la documentation du J2SE. Par exemple, la version de la JVM est obtenue en tapant la commande :

```
java -version
```

Bien sûr, ces commandes et le paramétrage de leurs options sont intégrées dans les IDE Java disponibles sur le marché.

Tester l'installation : votre première application Java

Recopiez le programme suivant dans un fichier texte dénommé `Bienvenue.java`. Respectez la casse des caractères du fichier et son extension `.java`.

EXEMPLE **Bienvenue.java**

```
class Bienvenue
{
  public static void main (java.lang.String [] args) 1
  {
    javax.swing.JOptionPane.showMessageDialog(null, "Bienvenue"); 2
  }
}
```

Cette application 1 affiche dans une boîte de dialogue le texte *Bienvenue* 2.

JAVA **Espace, retour à la ligne et casse**

Les espaces, retours à la ligne, tabulations, ne sont pas significatifs en Java sauf pour séparer un mot d'un autre. En revanche, vous devez faire attention à la *casse* des lettres (minuscule ou majuscule) dans un programme, car Java traite différemment une lettre selon qu'elle est écrite en minuscule ou en majuscule.

SOUS WINDOWS

Ouverture d'une fenêtre de commandes

L'un des moyens les plus simples pour ouvrir une fenêtre de commandes sous Windows consiste à sélectionner l'élément Exécuter dans le menu Démarrer, puis de taper `command` sous Windows 95/98/ME ou `cmd` sous Windows NT/2000/XP.

Figure 2–11
Icônes de la fenêtre de commande

sous Windows 98

sous Windows XP

du Terminal sous Mac OS X

Compilation de l'application

Pour compiler le fichier `Bienvenue.java` :

1. Ouvrez une fenêtre de commandes (sous Mac OS X, démarrez l'application Terminal du dossier Applications/Utilitaires).
2. Déplacez-vous avec la commande `cd` dans le dossier où se trouve le fichier `Bienvenue.java`.
3. Exécutez la commande suivante :

```
javac Bienvenue.java
```

Si votre programme est correctement compilé, la commande `javac` n'affiche aucun message et crée le fichier `Bienvenue.class` dans le dossier du fichier `Bienvenue.java`. Si le compilateur détecte une erreur, un texte décrivant l'erreur apparaît à l'écran. Remontez toujours à la première erreur du texte généré par `javac`, car les dernières erreurs sont souvent liées aux premières erreurs de la liste.

Les cinq erreurs de compilation les plus fréquentes

1 Commande inconnue

Sous Windows 95/98/ME :

```
Commande ou nom de fichier incorrect
```

Commande `javac` inconnue

Sous Windows NT/2000/XP :

```
'javac' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.
```

Commande `javac` inconnue

Sous Linux :

```
javac: command not found
```

Commande `javac` inconnue

La modification que vous avez apportée au `PATH` est incorrecte, ce qui empêche le système de retrouver la commande `javac`. Vérifiez le `PATH` et modifiez-le comme indiqué précédemment.

2 Fichier absent

```
error: cannot read: Bienvenue.java
```

Impossible de lire le fichier `Bienvenue.java`.

Vérifiez que le dossier courant contienne bien le fichier `Bienvenue.java`. Renommez le fichier exactement comme cela en cas de besoin, ou changez de dossier courant pour aller dans le dossier où se trouve le fichier.

3 Argument inconnu

```
javac: invalid argument: Bienvenue
Usage: javac <options> <source files>
...
```

`javac` ne connaît pas l'argument `Bienvenue`.

N'oubliez pas l'extension `.java` des fichiers.

4 Syntaxe : symbole oublié

```
Bienvenue.java:5: ';' expected
(recopie de la ligne 5)
```

`javac` s'attend à trouver le caractère cité (ici `;`) mais ne l'a pas trouvé.

Il vous faut certainement vérifier que vous ayez bien écrit le caractère attendu.

5 Symbole introuvable

```
Bienvenue.java:3: cannot resolve symbol
symbol  : class string
location: package lang
(recopie de la ligne 3)
```

`javac` ne retrouve pas le symbole cité, ici `string`. Cette erreur peut survenir dans de nombreux cas : mauvais nom de paquetage, de classe, de champ, de méthode, de variable, mauvais paramètres d'une méthode, casse incorrecte…

Vérifiez l'orthographe du symbole cité, ici `string` qui doit s'écrire `String`.

Autres erreurs de compilation

Voir aussi en annexe une liste plus complète des erreurs de compilation les plus fréquentes.

JAVA **main et showMessageDialog**

Les termes qui accompagnent les méthodes `main` et `showMessageDialog` seront expliqués au fur et à mesure de cet ouvrage. Il s'agit là des seuls termes qu'il vous soit demandé d'admettre dans un premier temps, la démarche de cet ouvrage étant de décrire systématiquement chaque élément de la syntaxe de Java avant sa première utilisation dans un programme.

C++ **Différences sur le main**

Le point d'entrée d'une application porte les mêmes noms en Java et en C++, mais c'est bien là leur seule ressemblance ! En effet, comme il est interdit de définir une fonction globale en Java, le point d'entrée d'une application doit être une méthode `main` définie dans une classe et cette méthode doit être déclarée comme dans la classe `Bienvenue`. C'est la raison pour laquelle les applications de cet ouvrage sont définies dans des classes utilisées uniquement comme contenant de leur `main`.
Cette architecture permet de créer et de faire cohabiter n'importe quel nombre de classes définissant une méthode `main`. La classe dont le `main` est utilisée comme point d'entrée est alors déterminée au lancement de la JVM avec la commande `java`.

C# **Différences sur le main**

Pour être utilisable comme point d'entrée d'une application Java, la méthode `main` d'une classe doit toujours être écrite tout en minuscules et être précédée de `public static void`. Elle doit aussi déclarer en paramètre un tableau de chaînes de caractères, même s'il ne sert pas dans l'application.

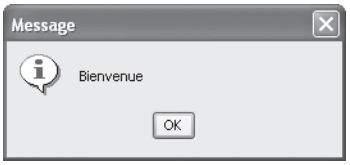


Figure 2–12 Application `Bienvenue`

Exécution de l'application

Exécutez ce programme avec la commande suivante :

```
java Bienvenue
```

Une fois que la fenêtre affichant *Bienvenue* est à l'écran, cliquez sur Ok. Si vous utilisez une version antérieure à Java 5.0, il vous faudra arrêter la JVM en tapant simultanément sur les touches Ctrl et C de votre clavier (nous verrons plus avant comment procéder pour quitter plus simplement un programme).

B.A.-BA **Commandes système les plus utiles**

Comme cet ouvrage prône l'utilisation de la ligne de commande pour débuter en Java, voici une liste des commandes les plus utiles sous Windows comme sous Unix (ce qui comprend Linux et Mac OS X).
Toute manipulation des fichiers et des dossiers (création, copie, renommage, suppression) via le bureau de votre système, l'Explorateur Windows ou le Finder Mac OS X est immédiatement disponible dans toute fenêtre de commandes ou tout terminal ouvert.

Effet	Sous Windows	Sous Unix
Lister les fichiers du dossier courant	`dir`	`ls`
Lister les fichiers d'extension `.java`	`dir *.java`	`ls *.java`
Lister les fichiers d'un dossier	`dir` *dossier*	`ls` *dossier*
Changer de dossier	`cd` *dossier*	`cd` *dossier*
Copier un fichier	`copy` *fichier1 fichier2*	`cp` *fichier1 fichier2*
Renommer un fichier ou un dossier	`ren` *fichier1 fichier2*	`mv` *fichier1 fichier2*
Supprimer un fichier	`del` *fichier*	`rm` *fichier*
Créer un dossier	`md` *dossier*	`mkdir` *dossier*
Supprimer un dossier vide	`rd` *dossier*	`rmdir` *dossier*

Les trois erreurs d'exécution les plus fréquentes

Si une erreur survient lors de l'exécution du programme, une exception est déclenchée empêchant généralement la JVM de poursuivre son exécution.

1 Définition de classe non trouvée (1)

```
Exception in thread "main" java.lang.NoClassDefFoundError: Bienvenue/
class
```

la JVM n'a pas trouvé la définition de la classe `Bienvenue`.

Vous avez dû taper la commande `java Bienvenue.class` (la commande `java` demande en paramètre une classe pas un fichier).

2 Définition de classe non trouvée (2)

```
Exception in thread "main" java.lang.NoClassDefFoundError: Bienvenue
```

la JVM n'a pas trouvé la définition de la classe `Bienvenue`.

Vérifiez que le dossier courant contienne bien un fichier `Bienvenue.class`. Si le problème persiste, assurez-vous que la variable d'environnement `CLASSPATH` soit égale à rien.

3 Méthode `main` non trouvée

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

la JVM n'a pas trouvé la méthode `main`.

Vérifiez la déclaration de la méthode `main` puis recompilez le programme.

REGARD DU DÉVELOPPEUR **Quel éditeur utiliser pour débuter ?**

Bien que cet ouvrage présente en annexe les fonctionnalités de deux des IDE les plus puissants, il vous est conseillé dans un premier temps d'apprendre Java en vous servant d'un éditeur de textes et de la ligne de commande. Cette approche vous évitera de vous laisser noyer par toutes les possibilités de ces IDE, tout en vous permettant de mieux comprendre comment s'organisent les fichiers d'un programme et comment s'utilisent les options des commandes Java que l'on retrouve dans les boîtes de dialogue des IDE.
Toutefois, si le Bloc-notes Windows (notepad.exe) ou TextEdit sous Mac OS X conviennent pour éditer quelques lignes de code, rien que l'absence de numéros de lignes dans ces éditeurs risque de vous pénaliser pour corriger les erreurs de compilation qui y font référence. Utilisez plutôt un éditeur de textes plus élaboré comme ceux de la liste suivante, gratuits et peu consommateurs de mémoire :

- ConTEXT sous Windows, disponible à http://www.context.cx/ et sur le CD-Rom qui accompagne cet ouvrage (si l'anglais vous gêne, les options de cet éditeur permettent de choisir un affichage en français) ;
- KEdit sous Linux (ou *vi* si vous le préférez) ;
- Xcode sous Mac OS 10.3 ou ProjectBuilder sous Mac OS 10.2 (fournis avec les outils de développement du système).

ASTUCES **Simplifiez-vous la vie avec les raccourcis !**

Tous les systèmes d'exploitation reproduisent le nom d'un fichier avec son chemin dans une fenêtre de commandes si vous glissez-déposez (*drag and drop*) l'icône de ce fichier dans la fenêtre. Très pratique aussi, vous pouvez utiliser le copier/coller dans une fenêtre de commandes via son menu contextuel. Finalement, Windows XP et Unix proposent la *complétion* automatique sur les fichiers et les dossiers dans une fenêtre de commandes pour vous éviter d'écrire entièrement leur nom : après avoir tapé les premières lettres d'un fichier, laissez le système compléter son nom en appuyant sur la touche de tabulation.

REGARD DU DÉVELOPPEUR **Performances de la JVM, portabilité**

Les performances de Java, langage principalement interprété, dépendent essentiellement de la machine virtuelle Java (JVM). Elles ont souvent été décriées par rapport à d'autres langages objet compilés tel que C++. Ce reproche est de moins en moins fondé :

- Les performances de la JVM dépendent étroitement des performances matérielles des machines (mémoire et processeur) et celles-ci ne cessent de progresser.
- Sun optimise régulièrement le système de gestion automatique de la mémoire (ramasse-miettes). Ainsi dans Java 1.4, le ramasse-miettes a-t-il été revu, ce qui a permis de bien meilleures performances, surtout sur les machines multi-processeurs.
- L'apparition de compilateurs « juste à temps » (JIT, *Just In Time compilers* en anglais) avec le JDK 1.1 a permis d'améliorer les performances. Une JVM classique interprète au fur et à mesure chaque instruction du bytecode Java en instructions du processeur de la machine, sans garder de trace de cette interprétation. En revanche, une JVM avec un compilateur JIT traduit à la volée (*on-the-fly*) le bytecode d'une méthode en instructions du processeur, garde en mémoire le résultat de cette traduction, puis exécute directement ces instructions chaque fois que nécessaire. Bien que cette *compilation* initiale ralentisse l'exécution d'une méthode à son premier appel, le gain en performances est d'autant plus grand qu'une méthode est exécutée plusieurs fois ou qu'elle contient des boucles d'instructions. Pour vous convaincre de l'efficacité du compilateur JIT, vous pouvez essayer vos applications Java en mode interprété pur grâce à l'option `-Xint` de la commande `java` pour voir la différence !
- Pour utiliser de façon optimale le compilateur JIT, la technologie HotSpot apparue avec J2SE 1.3 décide de façon intelligente s'il vaut mieux compiler une méthode avec le compilateur JIT ou bien l'interpréter car le temps perdu pour cette compilation n'en vaudrait pas la peine.

▸ http://java.sun.com/products/hotspot/

Et qu'en est-il de la portabilité des programmes Java, l'un de ses atouts les plus mis en avant ?

Dans les faits, le passage d'un même programme Java d'un système d'exploitation à l'autre sans la moindre modification ne pose généralement aucun problème pour les programmes à base de servlets/JSP comme les serveurs Web.

Économiquement, cet argument est très intéressant pour les entreprises qui développent des serveurs pour Internet : ceci leur permet de mettre à disposition des développeurs de simples ordinateurs sous Windows, Linux ou Mac OS au lieu de multiplier à grands frais les clones du serveur où sera déployé le programme final.

Du côté des programmes qui mettent en œuvre une interface utilisateur graphique avec les composants Swing ou AWT, les éventuels problèmes de portabilité se posent plus sous la forme d'une intégration correcte aux différents systèmes d'exploitation (et aux navigateurs pour les applets).

Vous devez donc prendre le temps d'étudier les spécificités de chaque système et adapter si nécessaire votre programme pour respecter le plus possible leurs différences, tout en gardant le même code Java.

JAVA 5.0 **Optimisation de la JVM**

Apparu avec Java 5.0, le partage de classe (*Class Data Sharing*) entre plusieurs JVM a pour but de réduire le temps de lancement des programmes Java (surtout pour les plus petits) et la quantité de mémoire nécessaire à leur fonctionnement. Pour ce faire, le programme d'installation de Java crée un fichier qui contient la représentation en mémoire des classes de la bibliothèque standard Java, ce qui accélère ensuite le temps de chargement de ces classes et permet de les partager entre différentes JVM en train de tourner simultanément. Le partage de classe n'est pas pris en charge sous Windows 95/98/ME.

En résumé...

Après avoir passé en revue les concepts de base de la programmation orientée objet, ce chapitre vous a montré comment ils se traduisent dans l'architecture Java. Les outils de développement Java étant installés, on peut maintenant étudier comment créer des classes et manipuler des objets...