Bien programmer en Java 7

Avec plus de 50 études de cas et des comparaisons avec C++ et C#

Plus de 10 000 ex. vendus !

Édition en couleur

Emmanuel Puybaret

EYROLLES

Avant-propos

Configuration logicielle requise

Les études de cas présentées dans cet ouvrage peuvent être reproduites sur toute machine qui supporte Java 7 (ou une version ultérieure) et le système de gestion de bases de données MySQL, c'est-à-dire à peu près sur n'importe quel système, notamment Windows, Linux, Mac OS X, Solaris et FreeBSD.

Ces études de cas ont en particulier été testées avec succès avec Java SE 7, MySQL 5.5 et Tomcat 7 sous Windows XP/7, Linux (distribution Ubuntu) et Mac OS X (10.7).

Java est reconnu comme l'un des meilleurs langages de programmation objet. Suivant une démarche didactique progressive, cet ouvrage vous aidera à comprendre la modélisation objet telle qu'elle est appliquée en Java et dans sa bibliothèque. Chaque concept est abordé isolément et accompagné d'une application simple et aussi concrète que possible. Enfin, pour vous aider à percevoir l'environnement Java dans sa globalité, cet ouvrage met en œuvre la création d'un forum de discussion.

Organisation de l'ouvrage

Après une présentation des principales applications dans le **chapitre 1**, cet ouvrage est divisé en trois parties.

La première partie couvre les fondements objet du langage Java : son architecture, la création de classes, la programmation de traitements et les mécanismes de réutilisation mis à disposition.

- Le **chapitre 2** présente les principes de la programmation objet et leur application dans l'architecture de Java avant d'aborder l'installation des outils de développement.
- Le **chapitre 3** est consacré à la création des classes et des objets, avec leurs méthodes et leurs champs.
- Le **chapitre 4** aborde la programmation des traitements d'une méthode grâce aux opérateurs et aux instructions de contrôle.
- Le **chapitre 5** explore les possibilités de la composition, de l'héritage et du polymorphisme pour concevoir l'architecture de vos classes.

Télécharger le code source

Le code source des études de cas peut être téléchargé sur le site d'accompagnement, à l'adresse :

▸ http://www.editions-eyrolles.com

Aux programmeurs Java 5 ou Java 6

Si vous êtes contraint d'utiliser Java 5 ou Java 6, un code source compatible avec ces anciennes versions est aussi disponible sur le site d'accompagnement.

Pour aller plus loin

Si vous avez des remarques à faire ou si vous recherchez des informations complémentaires sur les sujets abordés dans cet ouvrage, utilisez le forum prévu à cet effet à l'adresse :

▸ http://www.eteks.com

Conventions

Les lignes de code réparties sur plusieurs lignes en raison de contraintes de mise en pages sont signalées par la flèche ➥.

Les portions de texte écrites avec une police de caractères à chasse fixe et en italique, comme *VERSION*, signalent des informations à remplacer par un autre texte.

Marques déposées

La deuxième partie de l'ouvrage met en œuvre les classes principales de la bibliothèque Java dans diverses applications, avant d'aborder les mécanismes d'abstraction et de traitement d'erreurs.

- Le **chapitre 6** est consacré aux classes de la bibliothèque Java qui permettent de manipuler des textes et des dates, d'effectuer des calculs mathématiques ou de gérer des tableaux et des ensembles d'objets. Ce chapitre introduit aussi les classes de base du forum de discussion.
- Le **chapitre 7** aborde des notions indispensables pour bien utiliser la bibliothèque Java, à savoir les classes abstraites et les interfaces.
- Le **chapitre 8** présente les exceptions, qui constituent le mécanisme de gestion des erreurs.

La troisième partie décrit comment exploiter en Java les informations enregistrées dans des fichiers ou une base de données et exposer ces informations aux utilisateurs grâce à une interface homme-machine.

- Le **chapitre 9** présente les possibilités offertes par Java pour lire et écrire des informations dans des fichiers sous forme de flux de données.
- Le **chapitre 10** est consacré à la création d'interfaces utilisateur graphiques avec Swing ; il montre comment mettre en page des composants Swing et gérer les interactions de l'utilisateur avec ces composants, puis comment créer une application de carnet d'adresses et une applet de calcul de mensualités d'emprunt.
- Le **chapitre 11** est consacré à l'enregistrement et à la lecture d'informations dans une base de données grâce à JDBC et SQL, avec une mise en pratique pour gérer dans MySQL les utilisateurs et les messages du forum de discussion.
- Le **chapitre 12** présente comment créer dynamiquement des pages HTML avec les servlets et les pages JSP sur un serveur tel que Tomcat.
- Le **chapitre 13** est consacré à la création de l'interface utilisateur du forum de discussion avec des pages JSP.
- Le **chapitre 14** explique les fondements de XML et les très nombreuses façons d'exploiter ce dernier en Java.
- Le **chapitre 15** montre comment ajouter au forum une applet de chat et la rendre réactive grâce aux fonctionnalités multitâches intégrées à Java.

À qui s'adresse cet ouvrage ?

Que vous ayez peu de connaissances en programmation ou que vous maîtrisiez sur le bout des doigts les langages C, C++ ou C#, ce livre a

pour objectif de vous apprendre à programmer en Java comme un « pro ». Les débutants comme les développeurs Java y trouveront une description des fonctionnalités clés de Java illustrées par des solutions prêtes à l'emploi et la programmation d'un forum de discussion. La démarche pédagogique de cet ouvrage vous guidera d'autant mieux qu'il utilise une mise en pages élaborée pour mettre en valeur l'information essentielle, en reléguant sous forme de nombreux apartés les compléments d'informations.

Aux programmeurs C/C++

Vous connaissez déjà le C ou, mieux encore, le C++ et vous désirez apprendre Java ? Tant mieux, car ces langages ont des syntaxes proches, ce qui accélérera d'autant plus votre apprentissage. Pour vous aider à passer du C++ à Java plus rapidement, vous retrouverez tout au long de cet ouvrage les principales différences qui distinguent ces deux langages sous forme d'apartés C++.

Remerciements

Je tiens d'abord à remercier toutes les personnes de mon entourage qui m'ont soutenu dans ce travail de longue haleine, ne serait-ce que par leur curiosité... et tout particulièrement Diem My, Thomas et Sophie.

J'aimerais remercier aussi les stagiaires de la Brigade des Sapeurs Pompiers de Paris et de l'ITIN qui m'ont permis d'expérimenter l'approche du langage Java exposée dans cet ouvrage.

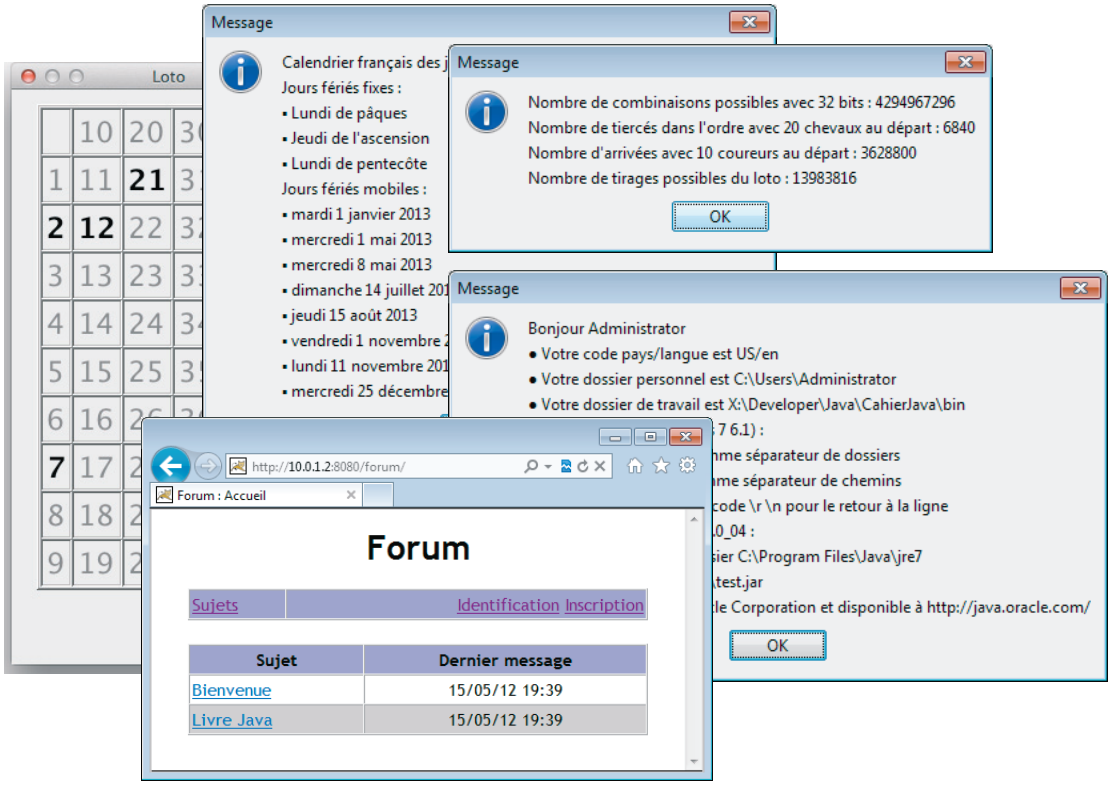
Finalement, un grand merci à l'équipe des éditions Eyrolles, tout particulièrement à Muriel, Jean-Marie, Gaël et Martine pour leur patience et leurs suggestions, ainsi qu'à Frédéric Baudequin, Régis Granarolo, Bernard Amade, Frédéric, Sophie, Anne-Lise, Géraldine et Laurène.

Aux programmeurs C#

Comme C# et Java sont des cousins très proches, vous vous rendrez rapidement compte que passer de l'un à l'autre n'est pas une tâche très ardue. Les principales différences entre ces deux langages sont mentionnées dans les apartés C#.

Table des matières

chapitre 1


Loto
Message
Calendrier français des j
Jours fériés fixes :
• Lundi de pâques
• Jeudi de l'ascension
• Lundi de pentecôte
Jours fériés mobiles :
• mardi 1 janvier 2013
• mercredi 1 mai 2013
• mercredi 8 mai 2013
• dimanche 14 juillet 201
• jeudi 15 août 2013
• vendredi 1 novembre
• lundi 11 novembre 201
• mercredi 25 décembre
Message
Nombre de combinaisons possibles avec 32 bits : 4294967296
Nombre de tiercés dans l'ordre avec 20 chevaux au départ : 6840
Nombre d'arrivées avec 10 coureurs au départ : 3628800
Nombre de tirages possibles du loto : 13983816
OK
Message
Bonjour Administrator
• Votre code pays/langue est US/en
• Votre dossier personnel est C:\Users\Administrator
• Votre dossier de travail est X:\Developer\Java\CahierJava\bin
7 6.1) :
me séparateur de dossiers
me séparateur de chemins
code \r \n pour le retour à la ligne
0_04 :
ier C:\Program Files\Java\jre7
test.jar
le Corporation et disponible à http://java.oracle.com/
OK
http://10.0.1.2:8080/forum/
Forum : Accueil
Forum
Sujets
Identification Inscription
Sujet
Dernier message
Bienvenue
15/05/12 19:39
Livre Java
15/05/12 19:39

Présentation des études de cas

Cet ouvrage décrit la création de différents types d'applications, depuis une simple application isolée mettant en pratique un concept Java, jusqu'au développement d'un forum de discussion détaillé sur plusieurs chapitres.

SOMMAIRE

MOTS-CLÉS

- Application
- Java
- Base de données
- MySQL
- Tomcat
- Forum
- Chat

Applications isolées

Le tableau 1-1 donne la liste des applications isolées (définies sur une ou deux sections qui se suivent) les plus intéressantes de cet ouvrage. Celles-ci pourront servir de socle pour le développement de vos propres applications.

Tableau 1–1 Description des applications isolées

Titre de l'application	Chapitre	Description
Convertir un nombre en toutes lettres	4	Montre comment convertir en toutes lettres un nombre compris entre 0 et 99 en tenant compte des exceptions de la langue française.
Quelques calculs de probabilité classiques	4	Calcule quelques probabilités connues en appliquant les formules mathématiques du calcul combinatoire.
Calculer les intérêts d'un compte épargne	5	Montre comment organiser deux types de comptes bancaires, l'un simple et l'autre permettant de calculer des intérêts cumulés.
Ce que connaît la JVM de votre système	6	Affiche les informations que connaît un programme Java sur votre système et son organisation.
Bon anniversaire	6	Calcule le nombre de jours avant votre prochain anniversaire.
Afficher les jours fériés de l'année	6	Affiche la liste des jours fériés français d'une année choisie par l'utilisateur.
Organiser les définitions d'un glossaire	6	Montre comment associer, dans un glossaire, un mot ou une expression à la définition correspondante.
Trier les événements d'un agenda dans l'ordre chronologique	7	Explique comment trier automatiquement les événements d'un agenda.
Quels numéros mettre dans ma grille de loto aujourd'hui ?	7	Tire aléatoirement 6 nombres entre 1 et 49 et les affiche dans une grille de loto.
Calculer le nombre de lignes de code d'un programme	9	Calcule le nombre de lignes de code, hors commentaires et lignes vides, des fichiers sources situés dans un dossier et ses sous-dossiers.
Enregistrer les factures de clients	11	Crée une table de factures dans une base de données puis retrouve les factures d'un client.
Créer une liste de courses	12	Montre comment créer sur un serveur web une liste de courses qui soit propre à chaque utilisateur du site.
Vérifier la validité d'un document XML	14	Vérifie si un document XML est bien formé et valide.
Afficher les nouvelles	15	Affiche un texte paramétrable défilant verticalement à l'écran.

Carnet d'adresses

L'application de carnet d'adresses permet de saisir les coordonnées d'un ensemble de contacts et de les afficher à l'écran dans un tableau.

Cette application vous montre comment créer une interface utilisateur avec les composants graphiques que vous avez l'habitude de trouver dans

la plupart des applications de votre ordinateur : fenêtres, menus, boîtes de dialogue, champs de saisie...

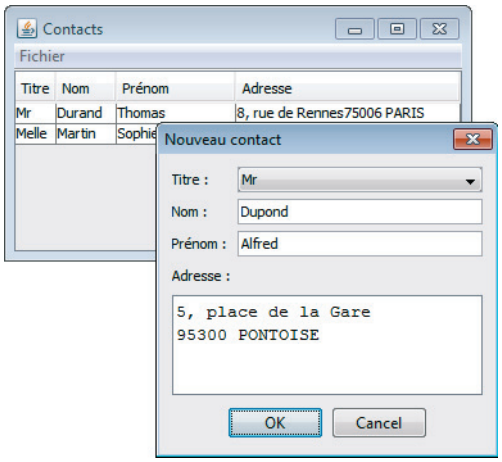


Figure 1–1
Saisie d'un contact dans l'application de carnet d'adresses

La programmation de l'application de carnet d'adresses sera décrite au chapitre 10, « Interfaces utilisateur avec Swing ».

Calcul des mensualités d'un emprunt

Cette application calcule le montant des mensualités et des intérêts d'un emprunt en fonction du capital emprunté, de la durée de l'emprunt et d'un taux d'intérêt.

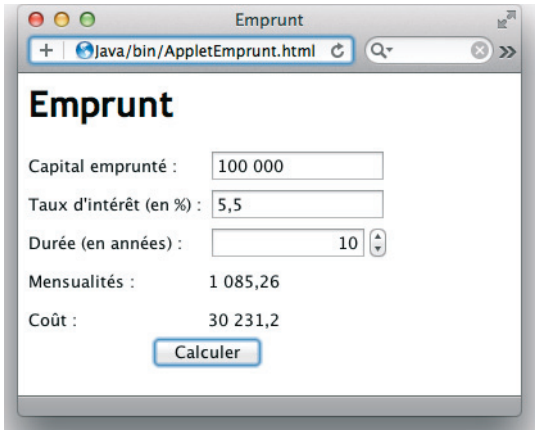


Figure 1–2
Calcul des mensualités d'un emprunt

Cette application sera développée aux chapitres 6, « Les classes de base de la bibliothèque Java » et 10 « Interfaces utilisateur avec Swing » :

- Dans la section « Calculer les mensualités d'un emprunt » du chapitre 6, il vous est d'abord montré comment calculer des mensualités en fonction de valeurs saisies par un utilisateur.
- L'interface utilisateur de cette application étant pour le moins rudimentaire (la saisie du capital, du taux d'intérêt et de la durée de l'emprunt se fait dans trois boîtes de dialogue affichées tour à tour), on montre en fin de chapitre 10 comment la transformer en une interface digne de ce nom.

Forum de discussion

Le forum de discussion présenté dans cet ouvrage reprend les fonctionnalités principales des forums disponibles sur Internet. Il permet à une communauté d'utilisateurs de partager des informations sous la forme de messages qui sont enregistrés par un serveur web. Ces messages sont regroupés par sujet, par exemple une question posée à la communauté ou un sujet de discussion lancé par un utilisateur. Les autres utilisateurs répondent à la question ou apportent leur contribution à la discussion lancée.

Principales fonctionnalités

La lecture des messages du forum est accessible à tout internaute connecté au serveur, mais la rédaction de nouveaux messages est réservée aux utilisateurs identifiés grâce à un pseudonyme et un mot de passe. Tout internaute peut devenir un membre de la communauté du forum en choisissant un pseudonyme unique. Une fois qu'un utilisateur est enregistré, le serveur lui attribue un mot de passe pour lui permettre de s'identifier avec le formulaire adéquat puis de contribuer au forum.

Un utilisateur identifié peut rédiger de nouveaux messages et modifier au besoin le contenu de ses anciens messages, grâce aux formulaires de rédaction prévus. Ses messages peuvent venir en réponse à d'autres ou lancer un nouveau sujet de discussion, chacun étant automatiquement daté du moment de sa création et signé du pseudonyme de son auteur. Pour éviter toute dérive dans les messages contraires à la netiquette, un utilisateur spécial, le modérateur, a le droit de modifier tous les messages du forum.

B.A.-BA Modérateur et netiquette

Le modérateur a la charge de modifier les messages des auteurs qui ne respectent pas la netiquette, pour éviter qu'ils ne portent atteinte aux bonnes mœurs (insulte, diffamation...) ou aux droits des personnes (non-respect des droits d'auteur, diffusion d'informations confidentielles...). Ce rôle de modérateur est d'autant plus nécessaire que les auteurs signent leurs messages avec leur pseudonyme pour assurer leur anonymat et que les messages du forum présenté dans cet ouvrage sont lisibles par tous les utilisateurs identifiés ou pas.

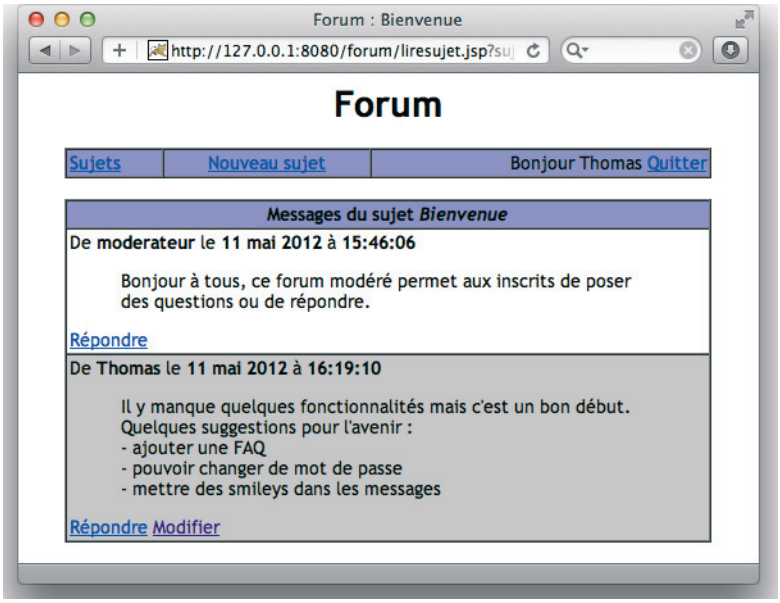


Figure 1–3
Exemple de page du forum affichant les messages d'un sujet

Le scénario complet d'utilisation du forum est décrit au début du chapitre 13, « Interface utilisateur du forum ».

Architecture technique

Le forum utilise une architecture qui fait intervenir les acteurs suivants :

- un serveur de base de données, pour enregistrer les utilisateurs et leurs messages ;
- un serveur web programmé en Java, pour gérer l'accès à la base de données et répondre aux requêtes des utilisateurs ;
- les navigateurs web des utilisateurs, pour afficher les pages renvoyées par le serveur web.

Le forum présenté ici utilise la base de données MySQL et le serveur Java Tomcat, mais la portabilité d'un programme Java permet en fait de déployer le programme prévu initialement pour Tomcat sur n'importe quel serveur qui prend en charge les pages JSP.

La base de données MySQL est elle aussi interchangeable avec la plupart des autres systèmes de gestion de bases de données du marché grâce au paramétrage du driver JDBC prévu pour le forum et décrit dans le chapitre 13, « Interface utilisateur du forum ».

Le forum étant l'application la plus complète de cet ouvrage, il est développé sur plusieurs chapitres comme suit :

- Une partie du **chapitre 6** montre comment décrire en Java un utilisateur et un message du forum et programmer différents outils nécessaires au forum, notamment pour calculer un mot de passe de façon aléatoire.
- Le **chapitre 11** est presque entièrement dédié à la gestion de l'enregistrement et de la lecture des utilisateurs et des messages dans une base de données comme MySQL.
- Le **chapitre 13** montre comment intégrer les outils décrits dans les chapitres précédents pour créer dynamiquement les pages HTML de l'interface utilisateur du forum sur le serveur web.
- Le **chapitre 14** explique comment retrouver une liste d'utilisateurs ou de messages dans des données au format XML.
- Enfin, le **chapitre 15** décrit comment créer un module de chat qui exploite les données au format XML fournies par le serveur web et l'intégrer au forum de discussion.

L'organisation de tous les fichiers nécessaires au fonctionnement du forum et du chat ainsi que le diagramme UML de leurs classes sont présentés dans l'annexe B.

Module de messagerie instantanée (chat)

À la différence du forum de discussion, le module de messagerie instantanée (chat) permet à chaque utilisateur identifié de dialoguer en direct avec les autres utilisateurs de la communauté. Ainsi, un utilisateur du chat voit apparaître dans son navigateur les messages postés dès leur rédaction, et ce, sans avoir à recharger la page dans son navigateur. Les conversations se déroulent « en temps réel » avec les autre utilisateurs.

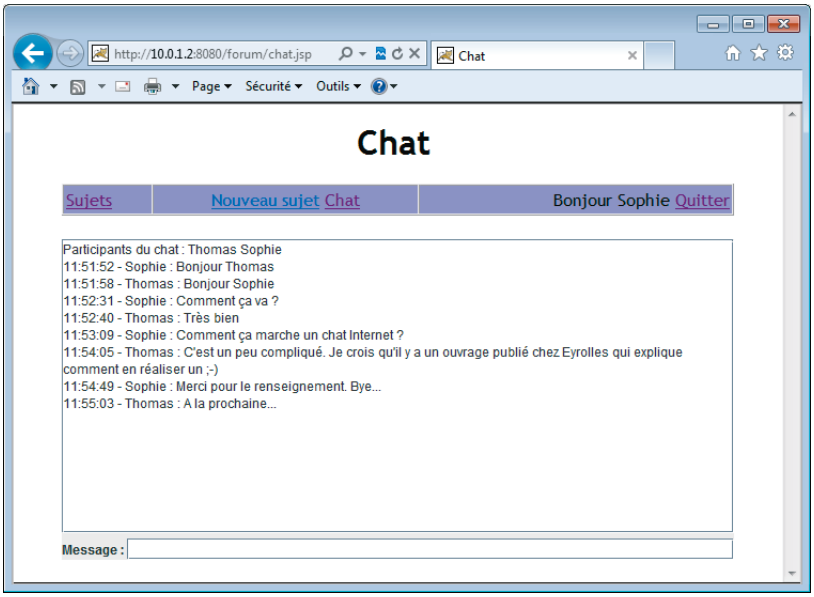


Figure 1–4
Exemple de conversation sur le chat

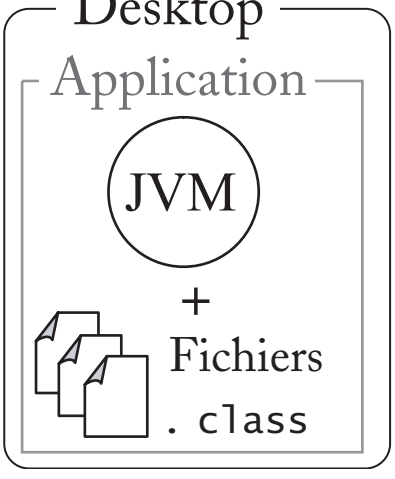
Le chat est développé au chapitre 15, « Messagerie instantanée avec la programmation multitâche » de cet ouvrage.

Ce module additionnel manipule aussi des notions d'utilisateur et de message et réutilise une partie des outils développés pour le forum.

En résumé…

De la plus simple à la plus complexe, les applications développées dans cet ouvrage vous donnent un aperçu réaliste des possibilités de Java et de sa très riche bibliothèque. Ces applications, nous l'espérons, vous permettront de démarrer vos premières applications Java sur des bases solides.

chapitre 2



Principes du langage et installation de l'environnement

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.

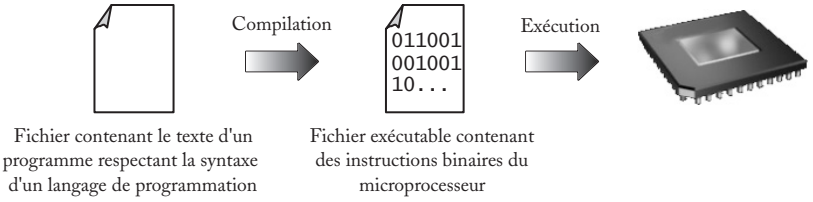


Figure 2–1
Compilation et exécution d'un programme

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 l'intègre 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 additionnent les valeurs de ces deux registres.

La programmation procédurale et structurée de langages comme le C, le Pascal, etc. 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.

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. Ce dernier 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.

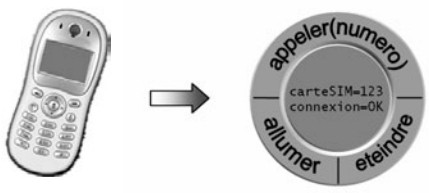


Figure 2–2
L'objet Téléphone portable et ses messages

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

Programme

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

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

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

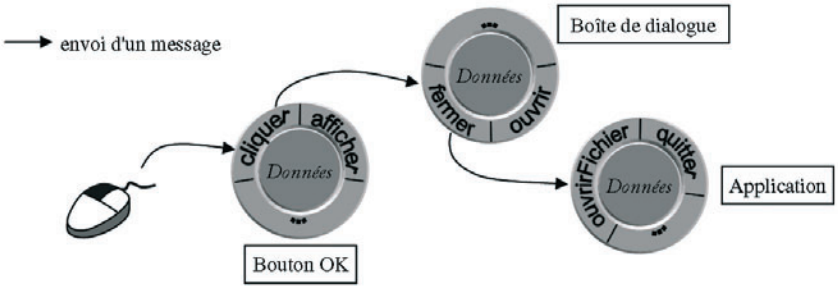
Le dernier message, `appeler`, est paramétrable. Il prend en paramètre un numéro de téléphone.

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.



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 y paraît et demande un bon esprit d'analyse.
- Le découpage d'un problème en objets qui soient les plus indépendants possibles 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 essaie 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.

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).

EXEMPLE

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

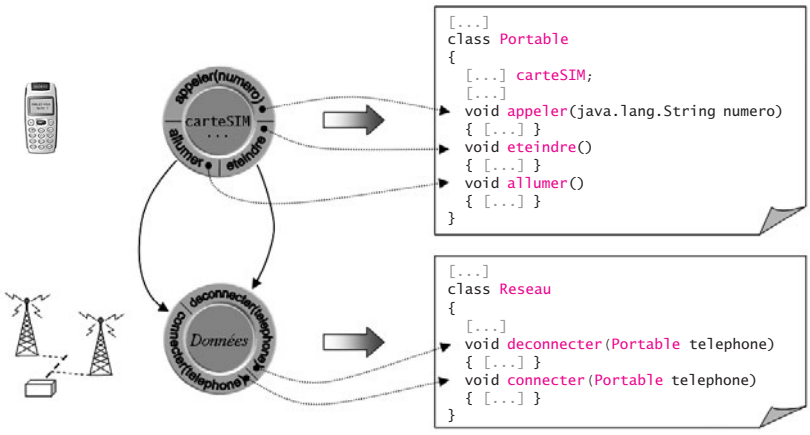


Figure 2–4 Identification des classes correspondant aux objets portable et réseau

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 à partir de Java 5). 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 !

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

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).

code binaire (*bytecode*) spécifique à Java. Un fichier `.class` décrit une classe, ses champs, ses méthodes et les instructions des méthodes.

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.

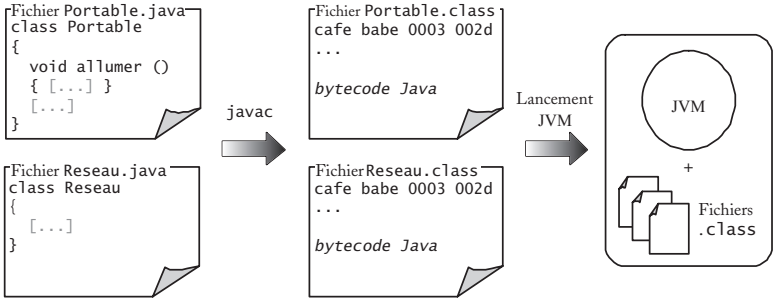


Figure 2–5 Cycle de développement Java

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*).

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 X traduit les instructions Java en instructions PowerPC ou Intel, etc.

À chaque besoin son environnement Java : applets, servlets et 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 classes 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

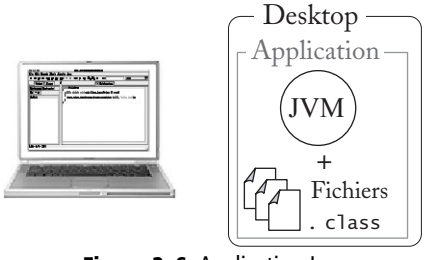


Figure 2–6 Application Java

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
  }
}
```

❷ Applet d'un fichier HTML lancée par un navigateur

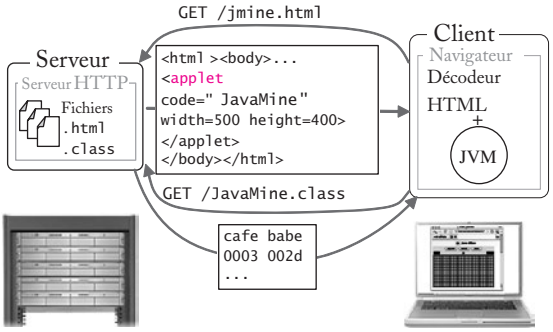


Figure 2–7 Applet Java

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
  }
}
```

❸ Servlet lancée par une requête sur un serveur web

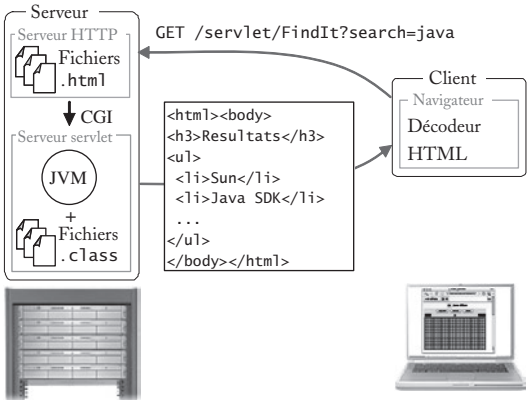


Figure 2–8 Servlet Java

Une servlet s'exécute sur un serveur web pour créer 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
  }
}
```

Télécharger et installer les programmes pour développer en Java

Les versions de Java pour les systèmes Windows, Linux, Solaris et Mac OS X sont disponibles en suivant le lien *Java SE* de la section *Software Downloads* sur le site web http://www.oracle.com/technetwork/java. Vous y trouverez 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

Sous Windows Modes de téléchargement

Le programme d'installation du JDK comme le JRE peut être téléchargé soit en un seul coup pour une installation *off line*, soit sous la forme d'un programme 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 quand une nouvelle version de Java est disponible sur le site d'Oracle.

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.

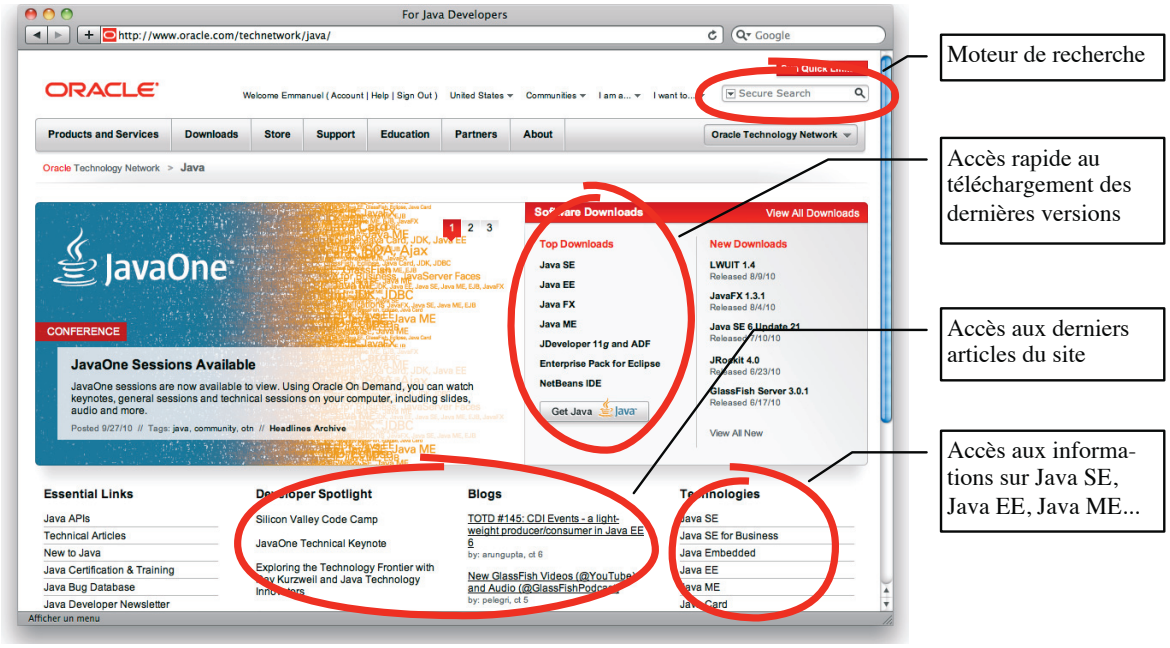


Figure 2–9 Documentation des API Java

VERSIONS **Les versions de Java depuis 1995**

Depuis sa première version en 1995, Sun Microsystems puis Oracle qui l'a racheté en 2009 sort régulièrement une version majeure de Java précédée de versions bêta 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 (1 232 classes)
- 2000 : 1.3 (1 466 classes)
- 2002 : 1.4 (2 097 classes)
- 2004 : 1.5 ou 5 (2 485 classes)
- 2006 : 1.6 ou 6 (2 821 classes)
- 2011 : 1.7 ou 7 (3 120 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'entreprises 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 crédit…

Finalement, depuis Java 6, ces appellations ont été remplacées par Java SE, Java EE et Java ME.

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

1. Téléchargez la version la plus récente du JDK. Ce fichier de plus de 60 Mo a un nom de la forme `jdk-VERSION-OS.ext`, où *VERSION* représente une suite de chiffres séparés par des points (par exemple `7u4`) et *OS* le système de destination du JDK (par exemple `windows-i586` pour Windows 32 bits).
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.

Installation sous Windows

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 sur le bouton *Démarrer* de Windows, choisissez *Panneau de configuration* et assurez-vous que toutes les icônes du panneau sont affichées.
3. Double-cliquez sur l'icône *Système*.
4. Sous Windows XP, choisissez l'onglet *Avancé* dans la boîte de dialogue des *Propriétés système* et sous Windows Vista/7, cliquez sur le lien *Paramètres système avancés*.
5. Cliquez sur le bouton *Variables d'environnement*.
6. Ajoutez une nouvelle 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
```

7. Confirmez votre saisie et fermez les boîtes de dialogue.

ATTENTION Mise à jour du PATH

Si le système vous indique que la commande `javac` 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, ouvrir une nouvelle fenêtre de commande ;
- sous Linux, exécuter la commande : `source ~/.bashrc`.

VERSIONS Java 8

Comme les versions 5 et 7 du Java SE ont enrichi le langage Java de nouveaux éléments syntaxiques détaillés dans cet ouvrage, la version 8 de Java SE apportera aussi son lot de nouveautés syntaxiques, comme les expressions Lambda. Prévu pour la fin 2013, vous pouvez suivre l'avancement de cette version sur le site suivant :

- http://openjdk.java.net/projects/jdk8/

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 dédiées au développement. 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`.

ASTUCE Copier un chemin

Pour être sûr de ne pas vous tromper en recopiant le chemin des commandes Java, ouvrez un *Explorateur* Windows, cherchez le dossier `bin` du JDK et copiez son chemin indiqué dans la barre d'adresse de l'explorateur.

ASTUCE DOSKEY et history

Cette fonctionnalité, disponible d'office sous Windows, 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.

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 :

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

6. Redémarrez votre session.

Installation sous Mac OS X

1. Exécutez le fichier d'installation `jdk-VERSION-OS.pkg` contenu dans l'archive que vous avez téléchargée et installez le JDK.
2. Lancez l'application *Préférences Java* située dans le dossier *Applications/Utilitaires*.
3. Assurez-vous que la version que vous venez d'installer apparaît bien en premier dans la liste des versions Java disponibles affichées dans les préférences.
4. Quittez l'application *Préférences Java*.

Pour les autres systèmes, consultez les sites de leurs éditeurs respectifs.

VERSIONS Java 6 sous Mac OS X

Sous Mac OS X 10.4 à 10.6, Java 5 ou 6 est installé d'office avec le système. Sous les versions ultérieures de Mac OS X, Java n'est pas installé avec le système, mais aussitôt que vous faites appel à une commande Java ou une applet, Java 6 est téléchargé et installé automatiquement. Dans les deux cas, ces versions sont maintenues par Apple et toute mise à jour éventuelle de celles-ci s'installe avec le module *Mise à jour de logiciels...* du menu *Pomme*.

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 packages 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, SVN ou d'autres outils.

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

- Eclipse http://www.eclipse.org/
- NetBeans http://www.netbeans.org/
- IntelliJ IDEA http://www.intellij.com/idea/

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

Télécharger les démos et la documentation

La page de téléchargement du JDK propose aussi de télécharger une archive compressée des démos Java (*Demos and Samples*) et de la documentation du JDK qui décrit notamment l'API (*Application Programming Interface*) des classes de la bibliothèque Java. Cette documentation 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 aussi en ligne à l'adresse :

http://docs.oracle.com/javase/7/docs/api

POUR ALLER PLUS LOIN **Autres documentations**

Parmi les nombreuses documentations en anglais fournies par Oracle 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.

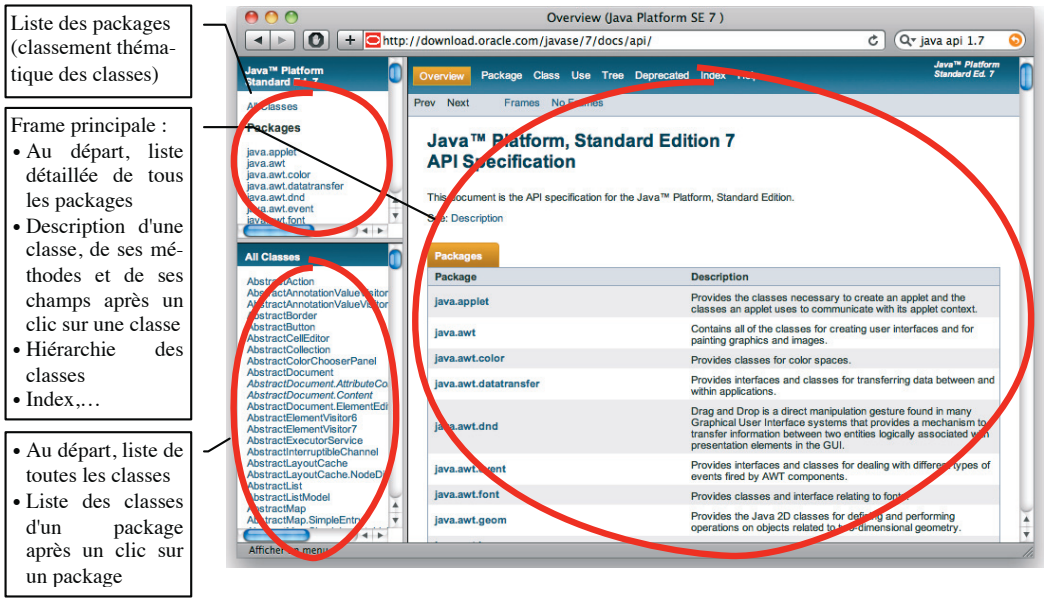


Figure 2–10 Documentation des API Java

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.

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 crée un fichier `.class` pour chacune des classes qu'il(s) contien(nen)t.
- **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 fournit 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 repérables 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 Java SE. 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`.

JAVA
Espace, retour à la ligne et casse Java

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 dans un programme, car Java traite différemment une lettre selon qu'elle est écrite en minuscules ou en majuscules.

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).

SOUS WINDOWS Ouverture d'une fenêtre de commandes

L'un des moyens les plus simples pour ouvrir une fenêtre de commandes consiste sous Windows Vista/7 à taper `cmd` dans le champ de saisie de recherche du menu *Démarrer,* et sous Windows XP à sélectionner l'élément *Exécuter* dans le menu *Démarrer*, puis à taper `cmd`.

sous Windows

sous Mac OS X

Figure 2–11
Icônes de la fenêtre de commande

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

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.

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`.

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 renvoyé par `javac`, car les dernières erreurs sont souvent liées aux premières de la liste.

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 cinq erreurs de compilation les plus fréquentes

1 Commande inconnue

Sous Windows :

Commande `javac` inconnue

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

Sous Linux :

Commande `javac` inconnue

```
javac: command not found
```

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

Impossible de lire le fichier `Bienvenue.java`.

```
error: cannot read: 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 celui où se trouve le fichier.

3 Argument inconnu

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

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

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

4 Syntaxe : symbole oublié

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

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

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

5 Symbole introuvable

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

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

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.

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* et le programme s'arrêtera ensuite de lui-même après quelques secondes.

Figure 2–12
Application Bienvenue

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.

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 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 **Quel éditeur utiliser pour débuter ?**

Bien que cet ouvrage présente en annexe les fonctionnalités d'Eclipse, un 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 (si l'anglais vous gêne, les options de cet éditeur permettent de choisir un affichage en français), disponible à http://www.contexteditor.org/ ;
- *KEdit* sous Linux (ou *vi* si vous le préférez) ;
- *Xcode* sous Mac OS X (fourni avec les outils de développement du système).

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 des 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 puis Oracle optimisent régulièrement le système de gestion automatique de la mémoire. Ainsi le ramasse-miettes a-t-il été revu plusieurs fois, ce qui a permis de bien meilleures performances.
- 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.
- Apparu avec Java 5, 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 fonctionnant simultanément.

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 X 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.

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...

Annexes

A. Types de licences logicielles

De nombreuses classes Java développées par des entreprises ou des particuliers sont disponibles sur Internet ou sur des CD-Rom de démonstration. Que le code source de ces classes soit disponible ou non, n'oubliez pas qu'elles sont utilisables uniquement sous les conditions de la licence concédée même si celle-ci n'est pas citée.

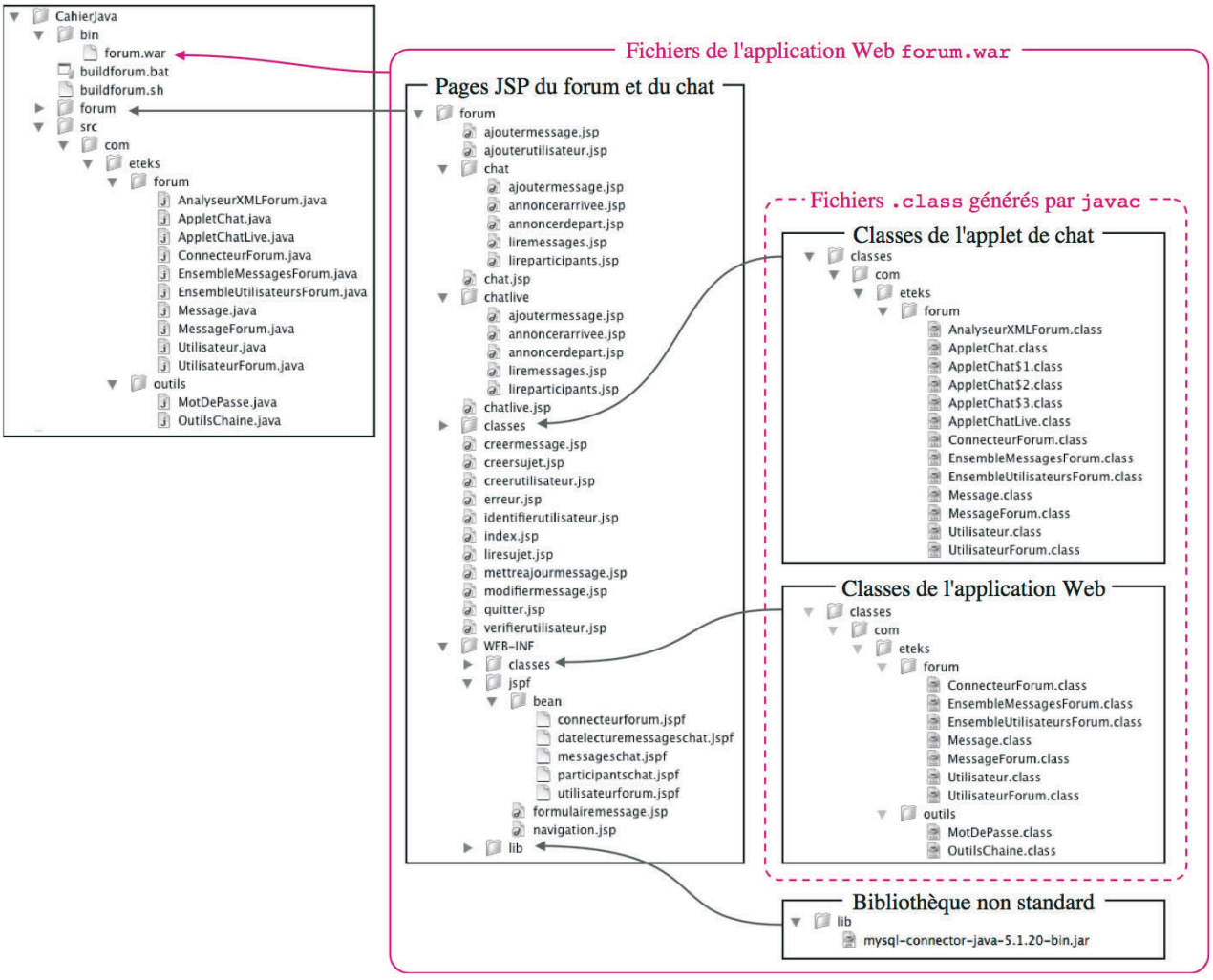
Suite à l'essor des logiciels libres (*free* en anglais à ne pas confondre avec gratuit !), on distingue aujourd'hui quatre grandes catégories de logiciels :

- Les logiciels du domaine public. Ils peuvent être utilisés, modifiés et distribués complètement librement, leur(s) auteur(s) ayant abandonné leurs droits.
- Les logiciels libres distribués sous licence Apache ou GNU LGPL. Ils peuvent être utilisés, modifiés et distribués en respectant certaines conditions assez peu contraignantes. Vous pouvez notamment réutiliser les classes ou les bibliothèques distribuées sous cette licence dans des logiciels non libres ou propriétaires (voir aussi http://www.apache.org/foundation/licence-FAQ.html pour plus de détails).
- Les logiciels libres distribués sous licence GNU GPL. Contrairement aux précédents, vous ne pouvez réutiliser les classes ou les bibliothèques distribuées sous cette licence que dans des logiciels libres eux aussi et disponibles sous une licence comparable. Ceci vous interdit donc de les réutiliser dans des logiciels propriétaires (voir aussi http://www.gnu.org/philosophy/philosophy.fr.html pour plus de détails)
- Les logiciels propriétaires. Ces logiciels ne peuvent être généralement réutilisés que sous certaines conditions contraignantes même s'ils sont distribués gratuitement. C'est la licence par défaut.

En cas de doute sur la licence des classes que vous désirez réutiliser dans votre programme, écrivez à son auteur pour plus d'information. Si vous avez l'intention de distribuer vos propres classes, n'hésitez pas à opter pour l'une des licences précédentes.

B. Fichiers du forum de discussion

Les fichiers nécessaires au fonctionnement du forum et du chat sont organisés sous forme d'une application web Java, ce que montre la figure ci-dessous.



La création du fichier `forum.war` de l'application web est effectuée grâce au fichier de commandes `buildforum.sh`. Celui-ci effectue les actions suivantes :

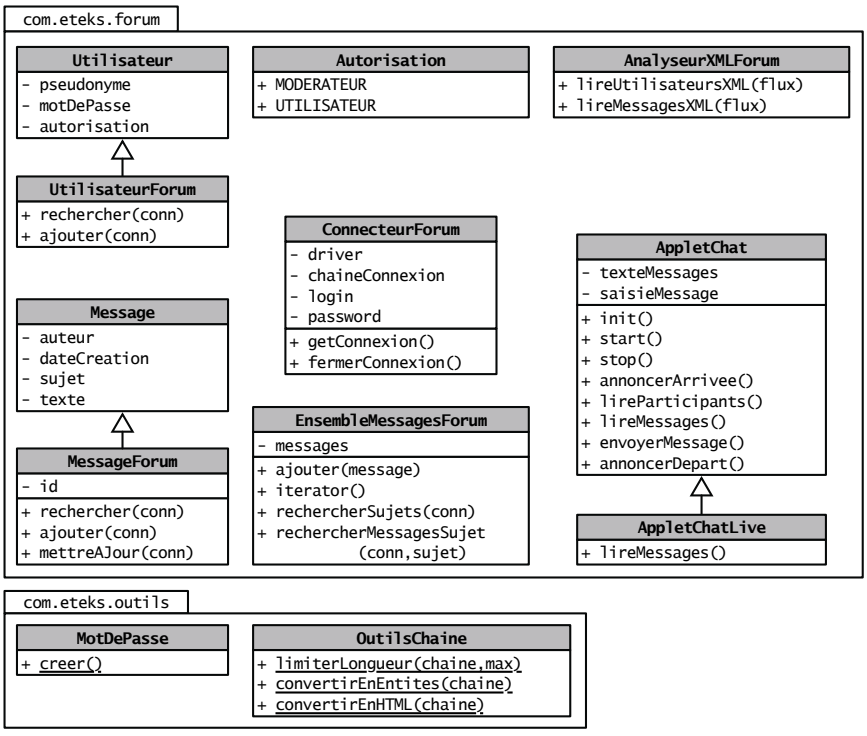
1. Compilation des classes nécessaires à l'application web en les rangeant dans le dossier `forum/WEB-INF/classes`.
2. Compilation des classes nécessaires à l'applet de chat en les rangeant dans le dossier `forum/classes`.
3. Création du fichier d'archive `bin/forum.war` avec le contenu du dossier `forum`.

```
javac -sourcepath ./src -d ./forum/WEB-INF/classes
  ➥ ./src/com/eteks/forum/UtilisateurForum.java
  ➥ ./src/com/eteks/forum/EnsembleMessagesForum.java
  ➥ ./src/com/eteks/outils/MotDePasse.java
  ➥ ./src/com/eteks/outils/OutilsChaine.java

javac -sourcepath ./src -d ./forum/classes
  ➥ ./src/com/eteks/forum/AppletChatLive.java

jar -cfM ./bin/forum.war -C./forum.
```

Le diagramme de classes UML ci-dessous présente les différentes classes du forum et du chat, avec leurs champs et/ou leurs méthodes principales.



Mise en route du forum

Il suffit de déposer le fichier `forum.war` dans le dossier `webapps` de Tomcat pour déployer le forum.

Le fichier `buildforum.bat` contient les mêmes commandes avec des caractères `\` à la place des caractères `/`.

C. Précisions sur les commentaires javadoc

Un commentaire entre `/**/` est un commentaire `javadoc` utilisé avant la déclaration d'une classe, d'une interface, d'un champ, d'une méthode ou d'un constructeur.

Ce commentaire est un texte descriptif au format HTML suivi éventuellement de balises `javadoc` précédées du caractère `@` comme `@param` ou `@return`. Par convention, un commentaire `javadoc` répète le caractère `*` à chaque début de ligne, caractère omis dans la documentation produite.

Balise javadoc	Usage

`@author` *auteur*	Décrit l'auteur d'une classe ou d'une interface. Peut être répété pour citer plusieurs auteurs. Exemples :
	`@author Alfred Dupont`
	`@author Georges Durand`

`@version` *version*	Décrit la version d'une classe ou d'une interface. Exemple : `@version 1.1.3`

`@see` *Classe*
`@see` *Classe#champ*
`@see` *Classe#Classe*
`@see` *Classe#methode*
`@see` *Classe#methode(typeParam)*
`@see` *Interface*
`@see` *Interface#methode*
	Crée dans la documentation un lien hypertexte vers une classe, une interface, un champ, une méthode ou un constructeur en rapport avec la classe, l'interface, le champ, la méthode ou le constructeur commenté. Exemples :
	`@see com.eteks.outils.Service#Service`
	`@see com.eteks.outils.Payant`
	`@see com.eteks.outils.Payant#getPrix`

`@param` *parametre commentaire*	Décrit un paramètre d'une méthode ou d'un constructeur. Exemple : `@param prix nouveau prix du produit.`
`@return` *commentaire*	Décrit la valeur retournée par une méthode. Exemple : `@return le prix de ce produit.`
`@exception` *ClasseEx commentaire*	Décrit les circonstances dans lesquelles une méthode ou un constructeur est susceptible de déclencher l'exception de classe *ClasseEx*. Exemple : `@exception java.lang.IllegalArgumentException si le parametre est negatif ou plus grand que 20.`

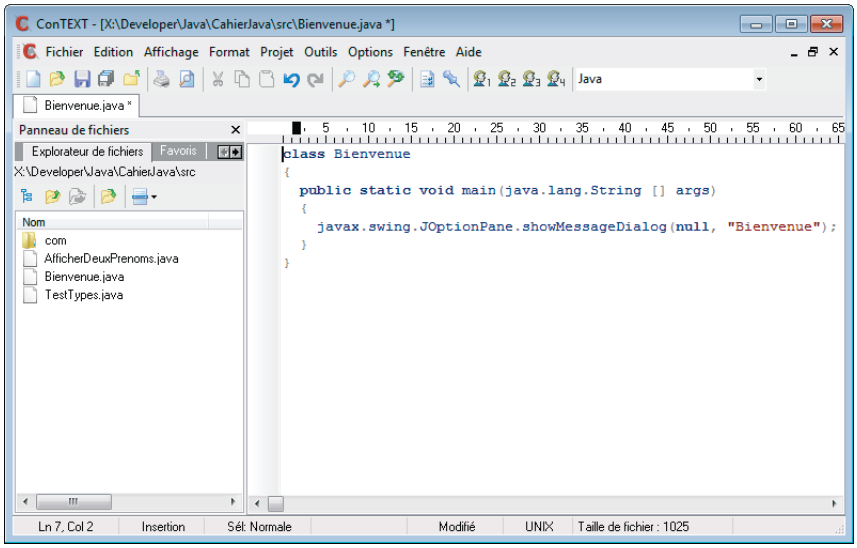
La première phrase d'un commentaire `javadoc` est affichée dans le résumé de la documentation d'une classe.

D. Mise en route de ConTEXT et d'Eclipse

Vous trouverez dans cette section une description de ConTEXT, un éditeur de textes pour écrire vos premiers programmes Java sous Windows, ainsi qu'une introduction à Eclipse, l'un des IDE les plus puissants du marché.

ConTEXT

Disponible sur le site http://www.contexteditor.org/fr/, ConTEXT est un éditeur gratuit suffisant pour débuter la programmation en Java ou pour éditer des programmes sur une configuration matérielle ancienne.



Installation

Lancez le programme d'installation `ConTEXTVERSION.exe` que vous aurez téléchargé, puis laissez-vous guider par l'assistant d'installation. ConTEXT s'installe dans le dossier de votre choix.

Démarrage

Lancez le programme *Context*. Si vous le désirez, le français peut être utilisé comme langue d'affichage en sélectionnant l'élément *Environment options...* du menu *Options*, puis en choisissant le français dans la liste proposé en bas de la boîte de dialogue affichée et en relançant l'éditeur. Sélectionnez Java comme langage par défaut de l'éditeur dans l'option *Syntaxe* de l'onglet *Éditeur* de cette même boîte de dialogue pour que les nouveaux fichiers bénéficient de la coloration syntaxique Java.

Création des classes

Comme ConTEXT est un éditeur de textes général, vous n'avez qu'à sélectionner l'élément *Nouveau* du menu *Fichier* pour créer une classe dans un nouveau fichier.

ASTUCE Gérer les fichiers courants

Le *Panneau de fichiers* de ConTEXT intègre un explorateur de fichiers, une liste de fichiers favoris et un historique qui vous aide à retrouver les fichiers dont vous vous servez le plus souvent. Il est aussi possible de créer de nouvelles listes personnalisées grâce aux éléments du menu *Projet*.

Édition des classes

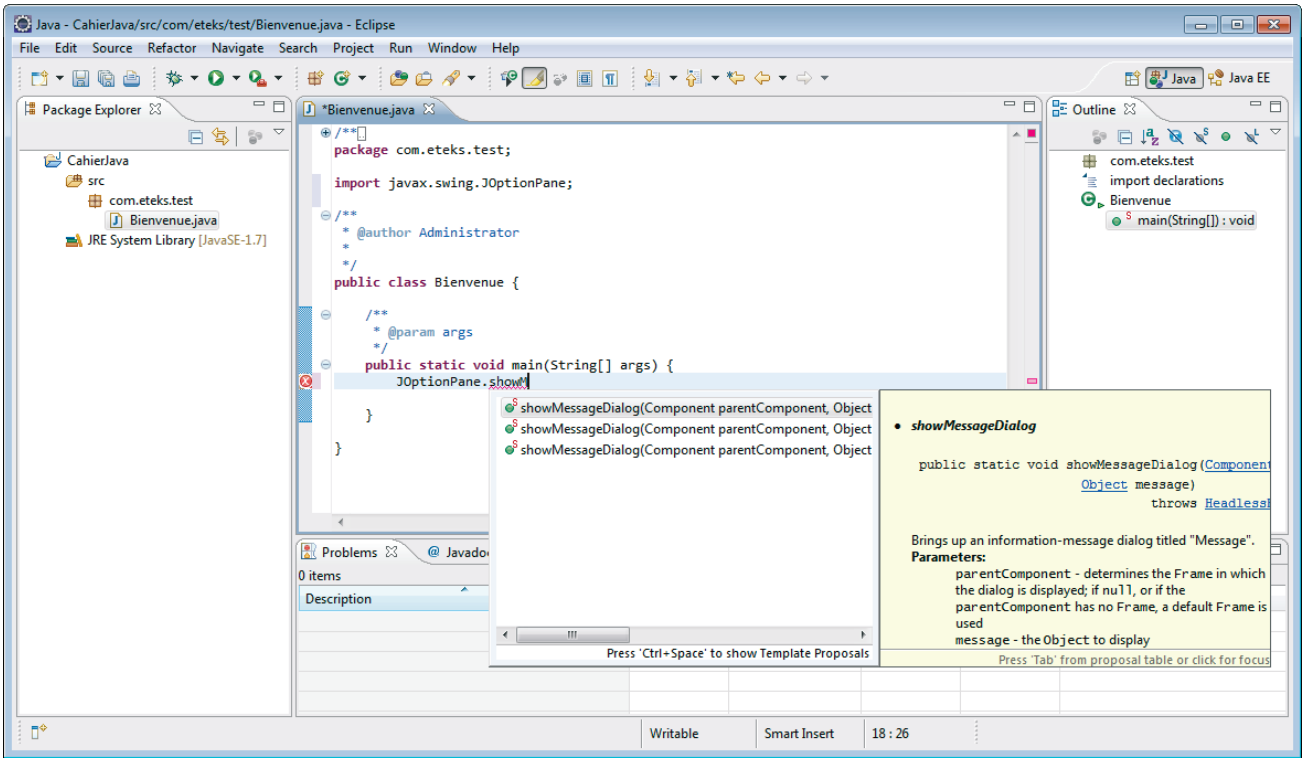
Outre les outils classiques d'un éditeur que vous retrouverez dans le menu *Édition*, quelques outils sont mis à votre disposition dans le menu *Format* pour accélérer l'édition des classes : indentation en bloc de plusieurs lignes, mise en commentaire de code mais aussi l'option *Insérer code depuis modèle* qui permet d'écrire des portions de code en ne tapant que quelques lettres, par exemple pour créer une boucle à partir du mot `for`. La liste des modèles est disponible par le biais de l'élément *Modèles de code...* du menu *Options*.

Compilation et exécution

Si vous souhaitez compiler ou exécuter une application directement à partir de ConTEXT, vous devez construire les commandes `javac` et `java` correspondantes grâce aux outils de l'onglet *Touches d'exécution* de la boîte de dialogue *Options d'environnement*.

Eclipse

Disponible pour Windows, Mac OS X et Linux sur le site http://www.eclipse.org, Eclipse est un IDE Open Source parmi les plus utilisées.



Installation

Une fois téléchargé le fichier d'installation de la version *Eclipse IDE for Java EE Developers* pour votre système, décompressez-le dans le dossier de votre choix puis installez le JDK comme indiqué dans le chapitre 2, si ça n'est pas déjà fait.

Démarrage

Lancez le programme *Eclipse* de votre système situé dans le dossier `eclipse` (`eclipse.exe` sous Windows, `eclipse` sous Linux ou `Eclipse.app` sous Mac OS X).

Au premier lancement, Eclipse vous demande de renseigner le chemin de votre dossier de travail (*Workspace*) où seront rangés par défaut les projets. Une page d'accueil vous présentant les fonctionnalités d'Eclipse est finalement affichée.

Vous pouvez franciser l'interface utilisateur d'Eclipse en installant un *language pack* disponible sur leur site.

Création d'un projet

Pour créer un projet, choisissez l'élément *Project...* dans le sous-menu *New* du menu *File*. Après avoir sélectionné le type de projet *Java Project*, l'assistant (*wizard*) *New Project* qui s'affiche vous propose alors de choisir le nom de votre projet, le dossier dans lequel il sera enregistré, la version du JDK (7 ou autre) avec lequel il est compatible, ainsi que les sous-dossiers où seront rangés les fichiers sources `.java` et les fichiers `.class` si vous désirez les séparer. Cet assistant permet aussi de sélectionner les sous-projets et les bibliothèques nécessaires à votre projet. Une fois la création du projet confirmée, si Eclipse vous propose de passer en perspective Java, répondez par l'affirmative.

POUR ALLER PLUS LOIN **Autres options d'un projet**

D'autres options comme le chemin où sont rangés les fichiers produits par `javadoc` et certaines options de compilation sont disponibles sur un projet. Vous retrouverez ces options dans la boîte de dialogue affichée en sélectionnant l'élément *Properties...* du menu *Project*. Les options communes à tous les projets comme celles relatives au formatage du code pour la position des accolades, l'indentation, la gestion des retours à la ligne... dépendent de la boîte de dialogue affichée en sélectionnant l'élément *Preferences...* du menu *Window*.

Création des classes

La création de classes s'effectue en sélectionnant l'élément *Class* dans le sous-menu *New* du menu *File* (si l'élément *Class* n'apparaît pas, sélectionnez l'élément *Other...* puis *Class* dans la liste qui s'affiche). L'assistant *New Java Class* qui s'affiche vous permet de renseigner l'identificateur de la nouvelle classe, son package, sa super-classe et diverses options comme l'ajout d'une méthode `main`, l'implémentation automatique des méthodes abstraites... À la confirmation de cette boîte de dialogue, la classe est créée dans le dossier des sources du projet et les sous-dossiers correspondant à son package sont créés automatiquement si nécessaire.

B.A.-BA **Perspective Eclipse**

Eclipse propose différentes perspectives et vues sur un même projet : une perspective *Resource* pour visualiser les fichiers du projet, des perspectives Java pour visualiser les packages et les classes du projet ou leur contenu, des perspectives CVS... Le choix d'une perspective s'effectue grâce aux éléments du sous-menu *Open perspective...* du menu *Window*.

Édition des classes

Outre les outils classiques d'un éditeur que vous retrouverez dans les menus *Edit*, deux fonctionnalités sont particulièrement utiles pendant l'édition de vos classes :

- La complétion automatique qui ajoute automatiquement les clauses `import` nécessaires à la saisie d'une classe, vous propose les méthodes disponibles sur un objet ou une classe avec des extraits de leur documentation `javadoc`... La complétion se déclenche soit volontairement grâce au raccourci clavier *Ctrl* + *Espace*, soit automatiquement dans certaines situations.
- Les modèles de code Java (*templates*) qui écrivent des portions de code en ne tapant que quelques lettres, par exemple pour créer une boucle d'itération à partir du mot `for`. Cette fonctionnalité s'obtient grâce au raccourci clavier *Ctrl* + *Espace* après avoir saisi l'un des mots de la liste des modèles disponibles. Cette liste est visible dans la section *Java* / *Editor* / *Templates* de la boîte de dialogue *Preferences* lancée par l'élément *Preferences...* du menu *Window*.

Les menus *Source* et *Refactoring* donnent accès à des fonctionnalités plus poussées comme l'élément *Generate getters and setters...* du menu *Source* qui ajoute automatiquement les accesseurs `get` et les mutateurs `set` d'une classe, ou l'élément *Rename...* du menu *Refactoring* qui renomme la classe, la méthode ou la variable en cours de sélection dans l'ensemble des fichiers `.java` d'un projet.

Compilation et exécution

Un fichier Java est compilé automatiquement au moment où vous l'enregistrez si l'élément *Build Automatically...* du menu *Project* est coché, ou en utilisant les éléments *Build* du menu *Project*.

Dans le menu *Run*, les éléments du sous-menu *Run As* et ceux du sous-menu *Debug As* permettent de lancer une application ou de la déboguer.

B.A.-BA Warning

En plus des erreurs de compilation, le compilateur d'Eclipse peut vous signaler des warnings mais vous n'êtes pas obligé de les prendre en compte. Un warning correspond à une instruction superflue, comme une clause `import` inutile, ou peut révéler un problème potentiel comme le fait de laisser un type de retour devant un constructeur, ce qui en fait une méthode. La liste des warnings est visible dans la section *Java* / *Compiler* de la boîte de dialogue *Preferences*.

E. Erreurs de compilation les plus fréquentes

Voici une liste des erreurs les plus fréquentes retournées par `javac` à la compilation de fichiers `.java`. Cette liste complète les autres erreurs décrites dans les différents chapitres de cet ouvrage.

Symbole introuvable

```
ClasseXxxx.java:numLigne: package com.eteks.outils does not exist
```

Le package `com.eteks.outils` n'existe pas.

Vérifiez si le dossier racine cité par l'option `-sourcepath` (ou le dossier courant si cette option n'est pas utilisée) contient bien l'arborescence de dossiers `com/eteks/outils`. Si l'erreur est provoquée par la clause `importcom.eteks.outils.*;` vérifiez par ailleurs que le dossier `com/eteks/outils` contient au moins un fichier `.java`.

```
ClasseXxxx.java:numLigne: cannot resolve symbol
symbol : class ClasseYyyy
location: package com.eteks.outils
```

Classe *ClasseYyyy* du package `com.eteks.outils` introuvable.

Vérifiez que le dossier `com/eteks/outils` contient bien un fichier *ClasseYyyy*`.java` qui déclare la classe `public` *ClasseYyyy*.

```
ClasseXxxx.java:numLigne: cannot resolve symbol
symbol : class flot
location: class ClasseXxxx
```

Classe ou type inconnu.

Vérifiez la syntaxe du type primitif (ici `float` à la place de `flot`) ou de la classe.

```
ClasseXxxx.java:numLigne: cannot resolve symbol
symbol : variable getXxxx
location: class com.eteks.test.ClasseYyyy
```

Variable *getXxxx* introuvable. Une méthode sans paramètre est toujours suivi d'un couple de parenthèses vide.

Vérifiez si l'appel à la méthode *getXxxx* est suivi d'un couple de parenthèses.

```
ClasseXxxx.java:numLigne: cannot resolve symbol
symbol : method showMessageDialog(java.lang.String)
location: class javax.swing.JOptionPane
```

`showMessageDialog` avec un seul paramètre de classe `java.lang.String` n'existe pas.

Ajoutez `null` ou un composant en premier paramètre. Pour d'autres méthodes, vérifiez le nombre et le type des paramètres requis par la méthode.

```
ClasseXxxx.java:numLigne: cannot resolve symbol
symbol : constructor ClasseYyyy()
location: class com.eteks.test.ClasseYyyy
```

Le constructeur de la classe `com.eteks.test.ClasseYyyy` sans paramètre n'existe pas.

Si vous avez déclaré un constructeur avec paramètre dans la classe `com.eteks.test.ClasseYyyy` vous devez lui passer les valeurs attendues ou ajouter un constructeur sans paramètre à la classe `com.eteks.test.ClasseYyyy`.

Déclaration de classe incorrecte

Une classe `public` doit être déclarée dans un fichier du même nom suivi d'une extension `.java`.

```
ClasseXxxx.java:numLigne: class ClasseYyyy is public, should be
declared in a file named ClasseYyyy.java
```

Vérifiez si le nom du fichier correspond au nom de la classe.

Conflit entre les noms du package et de la classe.

```
ClasseXxxx.java:numLigne: package com.eteks.test.ClasseXxxx
clashes with class of same name
package com.eteks.test.ClasseXxxx;
```

Vérifiez si le nom du fichier correspond au nom de la classe.

`javac` a trouvé deux classes de même identificateur dans le même package.

```
ClasseXxxx.java:numLigne: duplicate class: class ClasseXxxx
```

Vérifiez si le nom du fichier correspond au nom de la classe.

Déclaration de méthode incorrecte

Déclaration de la méthode non valide, type de retour exigé. Seuls les constructeurs ne sont pas précédés d'un type de retour.

```
ClasseXxxx.java:numLigne: invalid method declaration; return type
required
```

Ajoutez `void` ou le type renvoyé devant le nom d'une méthode.

Identifiant attendu. L'identifiant d'une méthode est précédé de `void` si elle ne renvoie pas de valeur ou d'un type si elle renvoie une valeur.

```
ClasseXxxx.java:numLigne: <identifier> expected
  public void float getYyyy()
              ^
```

Éliminez `void` ou le type (ici `float`) en fonction de ce que doit renvoyer la méthode.

Modificateur d'accès incorrect

La classe `com.eteks.test.ClasseYyyy` n'étant pas `public` elle est inaccessible en dehors de son package.

```
ClasseXxxx.java:numLigne: com.eteks.test.ClasseYyyy is not public in
com.eteks.test; cannot be accessed from outside package
```

Ajoutez le modificateur d'accès `public` à la classe `com.eteks.test.ClasseYyyy`.

La méthode *`methodeZzz`* de la classe `com.eteks.test.ClasseYyyy` n'étant pas `public` elle est inaccessible en dehors de son package.

```
ClasseXxxx.java:numLigne: methodeZzz() is not public in
com.eteks.test.ClasseYyyy; cannot be accessed from outside package
```

Ajoutez le modificateur d'accès `public` à la méthode *`methodeZzz`*.

```
ClasseXxxx.java:numLigne: getXxxx() in ClasseXxxx cannot override
getXxxx() in SuperClasseXxxx; attempting to assign weaker access
privileges; was public
```

Une méthode ne peut pas avoir un modificateur d'accès qui restreint la portée de la méthode redéfinie de sa super-classe (plus faible = *weaker*). L'ordre de priorité des modificateurs d'accès est du plus restrictif au moins restrictif : `private`, *package protected*, `protected` et `public`.

Utilisez le même modificateur d'accès ou un modificateur d'accès moins restrictif dans la sous-classe. Ici, n'oubliez pas d'ajouter `public` à la déclaration de la méthode *`getXxxx`* dans la classe *`ClasseXxxx`*.

Déclaration de variable locale incorrecte

```
ClasseXxxx.java:numLigne: illegal start of expression
    private int somme;
```

Expression invalide. `somme` ne peut pas être déclarée `private` si c'est une variable locale.

Supprimez `private` si `somme` est une variable locale.

```
ClasseXxxx.java:numLigne: texte is already defined in
main(java.lang.String[])
```

La variable locale `texte` est déjà déclarée dans la méthode `main`.

Déclarez votre variable avant l'instruction ou incluez-la dans un bloc. Vérifiez si vous n'avez pas oublié des accolades.

```
ClasseXxxx.java:numLigne: not a statement
      java.lang.String message;
```

Les instructions `if` `else` `for` `while` ou `do` doivent être suivies d'une instruction ou d'un bloc d'instructions. Une déclaration de variable locale n'est pas une instruction.

Déclarez votre variable avant l'instruction ou incluez-la dans un bloc. Vérifiez si vous n'avez pas oublié des accolades.

Utilisation de variable incorrecte

```
ClasseXxxx.java:numLigne: variable x might not have been initialized
```

Tentative d'utilisation de la variable locale `x` déclarée mais pas initialisée.

Initialisez la variable `x` avant de l'utiliser.

```
ClasseXxxx.java:numLigne: possible loss of precision
found : double
required: float
```

Perte possible de précision en passant du type `double` au type `float`. Attention les valeurs littérales décimales ont un type `double` par défaut !

Ajoutez un `f` à la fin d'une valeur littérale décimale pour indiquer qu'elle est de type `float`.

Erreur avec return

```
ClasseXxxx.java:numLigne: missing return statement
```

Manque une instruction `return` pour renvoyer le résultat de la méthode.

Ajoutez l'instruction `return` suivie du résultat de la méthode.

```
ClasseXxxx.java:numLigne: unreachable statement
```

Instruction impossible à atteindre.

Vérifiez la logique des instructions de la méthode : une instruction `return` ne doit pas être suivie d'une autre instruction.

Erreur dans les conditions des instructions if, for ou while

Une expression avec l'opérateur = doit avoir une variable à gauche du symbole =. Cette erreur survient quelquefois quand on utilise = au lieu de == pour une comparaison.

```
ClasseXxxx.java:numLigne: unreachable statement
```

Vérifiez si l'opérateur = est vraiment l'opérateur requis.

Types incompatibles. Cette erreur survient quelquefois quand on utilise = au lieu de == dans une condition de l'instruction if.

```
ClasseXxxx.java:numLigne: incompatible types
found   : int
required: boolean
```

Vérifiez si l'opérateur = est vraiment l'opérateur requis.

Équilibre incorrect entre accolades ouvrantes et fermantes

Déclaration d'une classe attendue. Cette erreur survient parfois quand il y a une accolade fermante de trop.

```
ClasseXxxx.java:numLigne: 'class' or 'interface' expected
```

Vérifiez l'équilibre entre les accolades ouvrantes et fermantes avant la ligne mise en cause.

Expression non valide. Si une instruction d'appel à une méthode suit l'accolade fermante, cette instruction est utilisée en dehors d'une méthode.

```
ClasseXxxx.java:numLigne: illegal start of expression
  }
```

Vérifiez si vous n'avez pas fermé trop tôt l'accolade de la méthode qui doit contenir l'instruction.

Déclaration incorrecte. Si la ligne où l'erreur survient est une instruction commençant par if else for while do ou return, cette instruction est utilisée en dehors d'une méthode.

```
ClasseXxxx.java:numLigne: illegal start of type
```

Vérifiez si vous n'avez pas fermé trop tôt l'accolade de la méthode qui doit contenir l'instruction.

Chaîne littérale non fermée

Chaîne de caractères littérale non fermée.

```
ClasseXxxx.java:numLigne: unclosed string literal
```

Vérifiez si vous n'avez pas oublié le caractère " à la fin de votre chaîne de caractères.
Attention : une chaîne de caractères littérale ne peut pas être répartie sur plusieurs lignes en Java.

Commentaire non fermé

Commentaire non fermé.

```
ClasseXxxx.java:numLigne: unclosed comment
```

Vérifiez si votre commentaire commençant par /* est bien fermé par */.

F. Bibliographie

[1] *UML, le langage de modélisation objet unifié* – (http://uml.free.fr/)

L'introduction sur la programmation objet de ce site expose la problématique posée par l'ajout d'un nouveau type d'ouvrage à une bibliothèque. Un modèle du genre qui vous permettra en plus d'apprendre UML !

[2] *Cahier du programmeur UML – Modéliser un site e-commerce*, Pascal Roques, Eyrolles 2008.

Pour ceux qui croient que l'apprentissage d'UML est ardu, une introduction limpide à la modélisation UML sur un cas qui concerne tous les développeurs web : la modélisation d'un site e-commerce.

[3] *The Java Tutorial* – Mary Campione, Kathy Walrath,... – Oracle (http://docs.oracle.com/javase/tutorial/, existe aussi en version papier aux éditions Addison Wesley)

De très bonnes documentations pour démarrer en Java et pour utiliser sa bibliothèque.

[4] *Thinking in Java* – Bruce Eckel – Mind View (http://www.mindview.net/ & http://penserenjava.free.fr/, existe aussi en version papier aux éditions Prentice Hall)

S'appuyant sur de nombreux exemples, les premiers chapitres de cet ouvrage de plus de 1000 pages traitent du noyau du langage Java avec une approche originale très efficace pour les personnes ayant déjà programmé. Son plus gros défaut est son manque d'illustrations (vous ne trouverez pas une seule capture d'écran dans le chapitre consacré à Swing !).

[5] *Java Look and Feel Design Guidelines* – Sun Microsystems – Addison Wesley, 1999

Ce bel ouvrage explique comment concevoir une interface utilisateur avec les composants Swing.

[6] *Conception de sites web : l'art de la simplicité* – Jakob Nielsen – CampusPress, 2000

S'appuyant sur des exemples de sites Internet existants (ou ayant existé), cet ouvrage expose les règles à utiliser pour créer un site web ergonomique.

[7] *Java efficace* – Joshua Bloch – Vuibert, 2002

Réservé aux programmeurs expérimentés, cet ouvrage très intéressant donne 57 recettes pour développer de meilleures classes en Java en s'appuyant notamment sur les design patterns.

G. Glossaire

Mot anglais ou mot-clé	Synonymes et traduction	Définition
`abstract`	Abstrait	Modificateur d'une classe interdite à l'instanciation ou d'une méthode non implémentée.
Access modifier	Modificateur d'accès	Mot-clé (`private`, rien, `protected` ou `public`) modifiant la portée d'un champ, d'une méthode ou d'une classe.
Accessor	Accesseur	Méthode généralement préfixée par `get` ou `is` renvoyant la valeur d'un champ.
API	*Application Programming Interface*	Liste des classes d'une bibliothèque mises à la disposition des programmeurs, avec leurs champs et leurs méthodes.
Cast	Conversion, transtypage	Opérateur utilisé pour convertir la représentation binaire d'une donnée d'un type primitif numérique dans un autre ou pour changer la classe d'une référence.
`class`	Classe, modèle, type d'objet	Type définissant un ensemble de champs et de méthodes communs à un ensemble d'objets.
Collection	Collection	Instance d'une classe gérant un ensemble d'éléments.
Constructor	Constructeur	Groupe d'instructions appelées pour initialiser un objet à sa création.
Encapsulation	Encapsulation	Protection des champs et des méthodes par l'utilisation du modificateur d'accès `private`.
`enum`	Énumération	Type de classe définissant un ensemble homogène de constantes dont le type est l'énumération elle-même.
Exception	Exception	Objet de diagnostic créé en cas d'erreur exceptionnelle.
Field	Champ, donnée, attribut, variable	Donnée déclarée dans une classe. Un champ d'instance mémorise une donnée pour chaque objet, un champ de classe mémorise une donnée globale d'une classe.
`final`	Non modifiable, constant	Modificateur d'une classe, d'une méthode, d'un champ, d'un paramètre ou d'une variable locale non modifiables.
Framework	Environnement, structure	Modèle de traitement requérant l'utilisation d'un ensemble de classes et d'un type d'implémentation.
Garbage collector	Ramasse-miettes	Tâche de la JVM collectant les objets inutiles pour libérer leur mémoire.
Heap	Tas	Zone de la mémoire utilisée pour stocker les objets Java.
Implement	Implémenter	Programmer les champs d'une classe et les instructions de ses méthodes.
Inherit	Étendre, hériter, dériver	Relation créée entre une classe et une autre sous catégorie de la première.
Instance	Instance	Objet créé à partir d'une classe.
`interface`	Interface	Ensemble de méthodes et de constantes que peut implémenter une classe pour accomplir une fonctionnalité.
Iterator	Itérateur	Outil utilisé pour énumérer les éléments d'une collection.
JVM	*Java Virtual Machine*	Interpréteur des fichiers `.class` Java.
Lifetime	Durée de vie	Période d'existence en mémoire d'une variable locale, d'un paramètre, d'un champ ou d'une classe.
Listener	Écouteur, auditeur	Classe utilisée pour suivre les événements émis par un composant réutilisable.
Member	Membre	Champ ou méthode d'une classe.

Index

C

D

E

F

G

H

I

M

N

O

P

R

S

Z

Mot anglais ou mot-clé	Synonymes et traduction	Définition
Method	Méthode, message, fonction membre	Traitement défini dans une classe répondant aux besoins d'une fonctionnalité. Une méthode d'instance manipule les champs d'instance d'un objet, une méthode de classe est un traitement global à une classe.
Mutator	Mutateur, modificateur	Méthode généralement préfixée par `set` modifiant la valeur d'un champ.
`native`	Natif	Modificateur d'une méthode dont l'implémentation est donnée dans une bibliothèque dynamique native du système d'exploitation.
Object	Objet	Module regroupant des données et les traitements s'y appliquant. Instance d'une classe.
Overload	Surcharge	Définition dans une classe de méthodes avec le même identificateur mais ayant des paramètres de types différents.
Override	Redéfinir, outrepasser, spécialiser, supplanter	Définition de méthodes d'instance avec le même identificateur et ayant les mêmes types de paramètres dans deux classes héritant l'une de l'autre.
`package`	Paquetage	Module rassemblant les classes traitant du même thème (application, bibliothèque).
Package protected	*Friendly*, portée par défaut	Modificateur d'accès d'un champ ou d'une méthode limitant sa portée aux classes du même package que sa classe.
Polymorphism	Polymorphisme	Faculté qu'a une classe de prendre plusieurs formes grâce à l'héritage, la relation *est un* et la redéfinition de méthodes.
`private`	Privé	Modificateur d'accès d'un champ ou d'une méthode limitant sa portée à sa classe.
Promotion	Promotion	Conversion d'un type primitif numérique dans un autre avec gain de précision.
Primitive type	Type primitif	L'un des types de données `byte`, `short`, `int`, `long`, `float`, `double`, `char` ou `boolean`.
Property	Propriété	Donnée d'un composant réutilisable accessible par un accesseur et éventuellement un mutateur.
`protected`	Protégé, héritable	Modificateur d'accès d'un champ ou d'une méthode limitant sa portée aux sous-classes de sa classe et aux classes du même package que sa classe.
`public`	Public	Modificateur d'accès d'un champ ou d'une méthode ayant la même portée que sa classe.
Reference	*Handle*, référence, pointeur	Variable locale, paramètre ou champ désignant un objet ou égal à `null`.
Scope	Portée, étendue	Zone du programme où une variable locale, un paramètre, un membre ou une classe est utilisable.
Signature	Signature	Combinaison de l'identificateur et des types des paramètres d'une méthode. Chaque méthode d'une classe doit avoir une signature unique. Une méthode qui redéfinit une autre méthode a la même signature.
Stack	Pile	Zone de la mémoire où sont empilés les variables locales et les paramètres, rendant plus rapide l'allocation et la libération de mémoire.
`static`	Statique, global	Modificateur des champs et méthodes de classe.
Subclass	Sous-classe, classe dérivée	Classe héritant d'une autre classe.
Super-class	Super-classe, classe de base, classe mère	Classe dont héritent d'autres classes. Toute classe hérite de la classe `java.lang.Object`.
Thread	Tâche, processus	Suite d'instructions exécutées en parallèle d'autres sur une même machine.
Wrapping class	Classe d'emballage, classe d'enveloppe	Classe mémorisant et manipulant une donnée d'un type primitif sous forme d'objet..