

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

© Groupe Eyrolles, 2012, ISBN : 978-2-212-12974-8

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

Les appellations suivantes sont des marques commerciales ou déposées des sociétés ou organisations qui les produisent :

- Java, JDBC, JSP, JVM, JDK, Java SE, Java EE, JavaBeans, Solaris, MySQL et Sun Microsystems d'Oracle Corporation.
 - Windows de Microsoft Corporation.
 - Mac OS X de Apple Computer Inc.
-

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 un 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 un 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.

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/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++.

/// 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

AVANT-PROPOS	V
Organisation de l'ouvrage • V	
À qui s'adresse cet ouvrage ? • VI	
Remerciements • VII	
1. PRÉSENTATION DES ÉTUDES DE CAS	1
Applications isolées • 2	
Carnet d'adresses • 2	
Calcul des mensualités d'un emprunt • 3	
Forum de discussion • 4	
Principales fonctionnalités • 4	
Architecture technique • 5	
Module de messagerie instantanée (chat) • 6	
En résumé... • 7	
2. PRINCIPES DU LANGAGE ET INSTALLATION DE L'ENVIRONNEMENT	9
Programmer en Java : une démarche objet • 10	
Du binaire à l'objet, 50 ans d'évolution de la programmation • 10	
Ce que fait un objet et comment il le fait... interface et implémentation • 12	
De l'analyse objet à l'écriture des classes Java • 13	
Écriture, compilation, exécution • 13	
À chaque besoin son environnement Java : applets, servlets et applications • 14	
Télécharger et installer les programmes pour développer en Java • 15	
Installation sous Windows • 17	
Installation sous Linux • 18	
Installation sous Mac OS X • 18	
Télécharger les démos et la documentation • 19	
Tester l'installation : votre première application Java • 20	
Compilation de l'application • 21	
Les cinq erreurs de compilation les plus fréquentes • 22	
Exécution de l'application • 23	
Les trois erreurs d'exécution les plus fréquentes • 23	
En résumé... • 25	
3. CRÉATION DE CLASSES	27
Typage : pourquoi et comment ? • 28	
Types de données objet et références • 29	
Écrire une valeur littérale • 29	
Affectation de variable • 30	
Par l'exemple : déclarer et utiliser quelques variables • 31	
Encapsuler pour protéger les données des objets • 32	
Portée d'utilisation et durée de vie • 33	
Manipuler des chaînes avec les méthodes de la classe <code>java.lang.String</code> • 34	
Par l'exemple : construire un texte avec plusieurs chaînes • 36	
Définir une nouvelle classe • 36	
Structure d'un fichier <code>.java</code> • 37	
Commenter une classe • 37	
Déclarer les champs d'une classe • 38	
Déclarer les méthodes d'une classe • 39	
Paramétrage d'une méthode • 40	
Implémenter les méthodes • 40	
Par l'exemple : une classe simulant une télécarte • 40	
Créer des objets • 42	
Par l'exemple : une histoire de télécarte empruntée... • 42	
Initialiser les champs d'un objet • 44	
Initialiser un objet avec un constructeur • 44	
Par l'exemple : une classe simulant un service • 46	

Surcharger les méthodes et les constructeurs • 47	
Organiser les fichiers des classes • 49	
Automatiser la compilation avec un fichier de commandes • 50	
Exécuter une application • 52	
Simplifier l'écriture des classes avec import • 52	
Par l'exemple : afficher les unités restantes d'une télécarte • 52	
En résumé... • 53	
4. CONTRÔLE DES TRAITEMENTS AVEC LES OPÉRATEURS, BOUCLES ET BRANCHEMENTS 55	
Opérateurs à connaître • 56	
Conversions numériques avec l'opérateur de cast • 58	
Par l'exemple : conversion euro/franc français • 59	
Priorité des opérateurs • 61	
Par l'exemple : comparer la somme de montants convertis • 61	
Piloter le programme avec les instructions de contrôle : boucles et branchements • 63	
Tester et décider sur condition avec if et switch • 63	
Syntaxe des instructions if et if else • 63	
Syntaxe de l'instruction switch • 63	
Par l'exemple : convertir un nombre en toutes lettres • 64	
Répéter un traitement avec les boucles while, do et for • 67	
Par l'exemple : quelques calculs de probabilité classiques • 69	
Portée des variables locales et des paramètres • 71	
En résumé... • 73	
5. RÉUTILISATION DES CLASSES 75	
Réutiliser en composant : la relation « a un » • 76	
Par l'exemple : une même adresse pour deux personnes • 76	
Réutiliser en héritant : la relation « est un » • 78	
Définir une sous-classe • 79	
Initialisation en deux temps pour les objets d'une sous-classe • 80	
Par l'exemple : alcoolisée ou non, choisissez votre boisson • 80	
Réutiliser en implémentant différemment : le polymorphisme • 82	
Relation « est un » et conversion de référence • 82	
Par l'exemple : boisson et boisson alcoolisée, ne mélangez pas les genres... • 83	
Modifier l'implémentation d'une méthode avec la redéfinition • 84	
Par l'exemple : changer de message • 85	
Modifier l'implémentation sans oublier la méthode redéfinie • 86	
Par l'exemple : calculer les intérêts d'un compte épargne • 87	
Réutiliser sans créer d'objet avec les méthodes de classe • 89	
Par l'exemple : afficher l'état d'un compte • 89	
Limiter la réutilisation avec final • 91	
Déclarer des constantes • 92	
Typer des constantes avec une énumération • 92	
Par l'exemple : tester le titre d'un contact • 93	
En résumé... • 94	
6. LES CLASSES DE BASE DE LA BIBLIOTHÈQUE JAVA 97	
La super-classe de toutes les classes : java.lang.Object • 98	
La méthode equals • 98	
La méthode hashCode • 98	
La méthode toString • 99	
Forum : utilisateur du forum de discussion • 99	
Manipuler les chaînes de caractères (java.lang.String) • 104	
Forum : outils de traitement pour les textes du forum • 104	
Communiquer avec la machine virtuelle (java.lang.System) • 107	
Par l'exemple : ce que connaît la JVM de votre système... • 108	
Effectuer des calculs mathématiques (java.lang.Math) • 110	
Par l'exemple : quelques valeurs mathématiques remarquables • 110	
Utiliser un type primitif sous forme d'objet avec les classes d'emballage • 111	
Par l'exemple : calculer les mensualités d'un emprunt • 112	
Gérer la date et l'heure • 114	
Mémoriser la date et l'heure (java.util.Date) • 115	
Afficher la date et l'heure (java.text.DateFormat) • 115	
Forum : message du forum • 116	
Fixer et manipuler la date et l'heure (java.util.GregorianCalendar) • 119	
Par l'exemple : bon anniversaire ! • 120	
Les tableaux pour gérer des ensembles d'éléments • 121	
Déclarer et créer un tableau • 122	
Utiliser un tableau • 123	
Forum : créer le mot de passe d'un utilisateur • 124	
Boucle itérative • 124	
Par l'exemple : afficher les jours fériés de l'année • 125	
Tableau multidimensionnel • 126	
Manipuler les tableaux avec java.util.Arrays • 126	
Par l'exemple : trier les paramètres d'une application • 127	
Les collections pour gérer des ensembles d'objets • 128	
Typer les objets d'une collection avec la généricité • 130	
Listes ordonnées d'objets (java.util.ArrayList et java.util.LinkedList) • 130	
Par l'exemple : casier à bouteilles ou cave à vin ? • 132	
Ensembles d'objets uniques (java.util.HashSet et java.util.TreeSet) • 133	

- Dictionnaires d'objets
(`java.util.HashMap` et `java.util.TreeMap`) • 134
- Par l'exemple : organiser les définitions d'un glossaire • 135
- En résumé... • 137
- 7. ABSTRACTION ET INTERFACE..... 139**
- Créer des classes abstraites pour les concepts abstraits • 140
- Par l'exemple : comparer les surfaces de différentes figures • 140
- Séparer l'interface de l'implémentation • 143
- Définir une interface • 143
- Par l'exemple : donner un prix à un objet • 144
- Implémenter une interface • 144
- Par l'exemple : implémenter le prix d'un objet • 145
- Utilisation des interfaces • 146
- Conversion de référence, suite et fin • 146
- Par l'exemple : boisson ou service, tout se paie • 146
- Par l'exemple : l'addition s'il vous plaît ! • 148
- Implémenter l'interface `java.lang.Comparable` pour comparer deux objets • 150
- Par l'exemple : gérer l'ordre chronologique d'événements • 150
- Énumérer les éléments d'une collection avec l'interface `java.util.Iterator` • 152
- Par l'exemple : trier les événements d'un agenda dans l'ordre chronologique • 153
- Manipuler les collections avec la classe `java.util.Collections` • 154
- Par l'exemple : quels numéros mettre dans ma grille de loto aujourd'hui ? • 155
- En résumé... • 159
- 8. GESTION DES ERREURS AVEC LES EXCEPTIONS 161**
- La pile d'exécution, organisation et fonctionnement • 162
- Par l'exemple : calculer une factorielle • 162
- Gérer les exceptions • 165
- Même un programme simple peut cacher des erreurs • 165
- Intercepter une exception avec `try catch` • 166
- Par l'exemple : vérifier les erreurs de saisie • 167
- Déclencher une exception avec `throw` • 168
- Par l'exemple : surveiller les cas limites • 168
- Décrire un traitement final avec `finally` • 170
- Par l'exemple : `finally`, demander confirmation pour continuer • 171
- Catégories d'exceptions Java • 172
- Exceptions non contrôlées • 172
- Exceptions contrôlées • 173
- Manipuler une classe à l'exécution avec la réflexion • 174
- Créer une classe d'exception • 179
- En résumé... • 179
- 9. LECTURE ET ÉCRITURE DE FICHIERS..... 181**
- Explorer le système de fichiers (`java.io.File`) • 182
- Par l'exemple : rechercher les fichiers dans un dossier et ses sous-dossiers • 183
- Lire et écrire des données sous forme de flux • 184
- Mode d'accès aux données • 185
- Mode d'accès par flux de données • 185
- Mode d'accès aléatoire • 186
- Lecture avec les flux de données • 186
- Contrôler les erreurs sur un flux de données avec les exceptions • 187
- Par l'exemple : compter le nombre d'occurrences d'un caractère dans un fichier • 189
- Écriture avec les flux de données • 190
- Filtrage des données d'un flux • 191
- Par l'exemple : éliminer les commentaires d'un programme Java • 195
- Par l'exemple : compter les lignes de code d'un ensemble de fichiers Java • 198
- Configurer une application • 200
- Fichiers de traduction • 200
- Fichiers de préférences • 201
- En résumé... • 201
- 10. INTERFACES UTILISATEUR AVEC SWING..... 203**
- Composants d'interface utilisateur • 204
- Mise en page des composants avec les layouts • 205
- Agencer les composants les uns à la suite des autres (`java.awt.FlowLayout`) • 205
- Par l'exemple : afficher des champs de saisie et leurs labels • 206
- Disposer les composants dans une grille (`java.awt.GridLayout`) • 207
- Par l'exemple : interface utilisateur d'un clavier de calculatrice • 207
- Placer les composants aux bords du conteneur (`java.awt.BorderLayout`) • 208
- Par l'exemple : interface utilisateur d'un éditeur de textes • 209
- Mise en page évoluée par combinaison de layouts • 212
- Par l'exemple : panneau de saisie des coordonnées d'un contact • 213
- À chaque système son look and feel • 216
- Interagir avec l'utilisateur grâce aux événements • 217

Événements • 218	Choisir un protocole pour communiquer • 262
Être à l'écoute des événements en implémentant un listener • 218	Adresse IP et port, point de rendez-vous des serveurs Internet • 263
Par l'exemple : quelle heure est-il ? • 219	Requête HTTP vers une URL • 263
Utiliser les classes anonymes pour implémenter un listener • 220	Par l'exemple : afficher le contenu d'une URL
Par l'exemple : calculer des tirages de loto • 220	dans une fenêtre Swing • 264
Par l'exemple : interface utilisateur d'un carnet d'adresses • 222	Programme CGI • 266
Créer vos composants graphiques • 225	Utiliser un formulaire HTML pour paramétrer
Par l'exemple : dessiner le plan d'une maison • 225	un programme CGI • 266
Programmer une applet • 227	Par l'exemple : un formulaire de recherche • 267
Par l'exemple : bienvenue dans le monde des applets ! • 228	Programmation d'une servlet sur le serveur • 268
Créer une interface utilisateur avec une applet • 230	Classe javax.servlet.http.HttpServlet • 268
Par l'exemple : interface utilisateur du calcul de mensualités • 230	Interface javax.servlet.http.HttpServletRequest • 268
En résumé... • 233	Interface javax.servlet.http.HttpServletResponse • 269
11. CONNEXION À LA BASE DE DONNÉES AVEC JDBC 235	Renvoyer du texte HTML avec une servlet • 269
Utilisation d'une base de données en Java • 236	Par l'exemple : Bienvenue dans le monde des servlets ! • 269
Se connecter à une base de données avec un driver JDBC • 237	Installation de Tomcat • 270
Par l'exemple : tester la connexion avec la base de données • 238	Sous Windows • 270
Installation du SGBD MySQL • 239	Sous Linux et Mac OS X • 271
Sous Windows • 239	Lancement de Tomcat • 271
Sous Linux • 239	Sous Windows • 271
Sous Mac OS X • 240	Sous Linux et Mac OS X • 272
Installer le driver JDBC • 240	Organiser les fichiers d'une application web • 272
SQL, le langage des bases de données • 241	Compilation d'une application web • 273
Principaux types de données • 241	Mise en route d'une application web • 273
Mettre à jour les tables et les index • 242	Par l'exemple : exécuter la servlet de bienvenue • 274
Modifier et rechercher les enregistrements d'une table • 242	Cycle d'exécution de la servlet de bienvenue • 274
Programmation SQL avec JDBC • 243	Mise à jour d'une application web • 275
Utiliser une connexion JDBC (java.sql.Connection) • 243	Créer l'interface d'une application web avec les JavaServer Pages • 276
Exécuter des instructions SQL (java.sql.Statement) • 244	Balises JSP pour inclure du contenu dynamique • 277
Exploiter les résultats d'une sélection SQL (java.sql.ResultSet) • 244	Variables JSP prédéfinies • 277
Par l'exemple : enregistrer les factures client • 245	Par l'exemple : bienvenue dans le monde JSP • 278
Obtenir des informations sur la base de données (java.sql.DatabaseMetaData) • 247	Exécuter la page JSP de bienvenue • 279
Forum : gérer la connexion à la base de données • 247	Contrôle des erreurs dans une page JSP • 279
Paramétrer les instructions SQL d'accès à la base du forum	Mise à jour des pages JSP • 280
(java.sql.PreparedStatement) • 250	Utiliser les classes Java dans une page JSP • 280
Forum : stocker utilisateurs et messages dans la base de	Utiliser les composants JavaBeans dans une page JSP • 280
données • 251	Par l'exemple : créer une liste de courses • 282
En résumé... • 258	Faire appel à d'autres pages JSP • 284
12. PROGRAMMATION WEB AVEC LES SERVLETS,	En résumé... • 284
JSP ET JAVABEANS..... 261	13. INTERFACE UTILISATEUR DU FORUM 287
Protocole HTTP et programme CGI • 262	Scénario d'utilisation • 288
Principe de l'architecture client-serveur • 262	Scénario pour un utilisateur non identifié • 288
	Scénario pour un utilisateur identifié • 288

Programmation des pages du forum • 290

Organisation des pages du forum • 290

Utilisation des classes des packages `com.eteks.forum` et `com.eteks.outils` • 290Classe `com.eteks.forum.ConnecteurForum` • 291Classe `com.eteks.forum.UtilisateurForum` • 291Classe `com.eteks.forum.MessageForum` • 292Classe `com.eteks.forum.EnsembleMessagesForum` • 292Classe `com.eteks.outils.OutilsChaîne` • 292Classe `com.eteks.outils.MotDePasse` • 292

Identification de l'utilisateur • 292

Page d'accueil • 296

Inscription d'un utilisateur • 299

Messages d'un sujet • 301

Création de sujet, de message, et modification • 302

Pages de saisie • 302

Pages d'ajout et de modification de message • 305

Quitter l'application • 306

En résumé... • 307

14. ÉCHANGER DES INFORMATIONS AVEC XML..... 309**Premiers contacts avec XML • 310**

Description d'un document XML • 310

Par l'exemple : représenter une facture en XML • 311

Document XML bien formé • 312

Espace de noms • 312

Par l'exemple : associer un espace de noms aux éléments d'une facture • 313

Document XML valide et DTD • 313

Créer une DTD • 314

Par l'exemple : définir la DTD des factures • 315

Utiliser une DTD dans un document XML • 316

Par l'exemple : utiliser la DTD d'une facture dans un document XML • 316

Typier les informations XML avec un schéma XML • 317

Types prédéfinis XML Schema • 317

Déclarer la syntaxe d'un document avec les éléments XML Schema • 317

Par l'exemple : définir le schéma XML des factures • 321

Utiliser un schéma XML dans un document XML • 322

Par l'exemple : utiliser le schéma XML d'une facture dans un document • 322

Transformer un document XML en un autre document • 322

Retrouver des éléments et des attributs avec XPath • 323

Transformer un document XML en un autre document XML avec XSLT • 323

Par l'exemple : extraire la liste des articles d'une facture • 324

Analyser un document XML avec JAXP • 325

Obtenir une instance d'un analyseur • 326

Analyser un document avec SAX • 326

Par l'exemple : rechercher les articles d'une facture • 327

Vérifier la validité d'un document avec SAX • 329

Par l'exemple : rechercher les erreurs dans un document XML • 329

Analyser un document avec DOM • 332

Par l'exemple : rechercher le client d'une facture • 332

Forum : rechercher les utilisateurs ou les messages d'un document XML • 333

Transformer un document XML • 338

Par l'exemple : transformer une facture par programme • 338

Par l'exemple : créer le document XML d'un modèle DOM • 339

Gérer la correspondance entre objets et éléments XML avec JAXB • 340

Définir la syntaxe d'un document XML avec les annotations JAXB • 341

Par l'exemple : définir une société et ses employés avec JAXB • 341

Marshalling et unmarshalling • 344

Par l'exemple : créer le document XML d'une société • 345

Par l'exemple : lire les objets correspondant au document XML d'une société • 345

En résumé... • 346

15. MESSAGERIE INSTANTANÉE AVEC LA PROGRAMMATION MULTITÂCHE 349**Gestion d'animations avec la classe `javax.swing.Timer` • 350**

Par l'exemple : afficher les nouvelles • 350

Programmation d'un thread avec la classe `java.lang.Thread` • 352Implémenter la méthode `run` • 353**Ajout d'un module de chat au forum de discussion • 354**

Interaction entre l'applet de chat et les pages JSP • 355

Composants JavaBeans du serveur pour le chat • 356

Ensemble des messages du chat • 356

Message du chat • 357

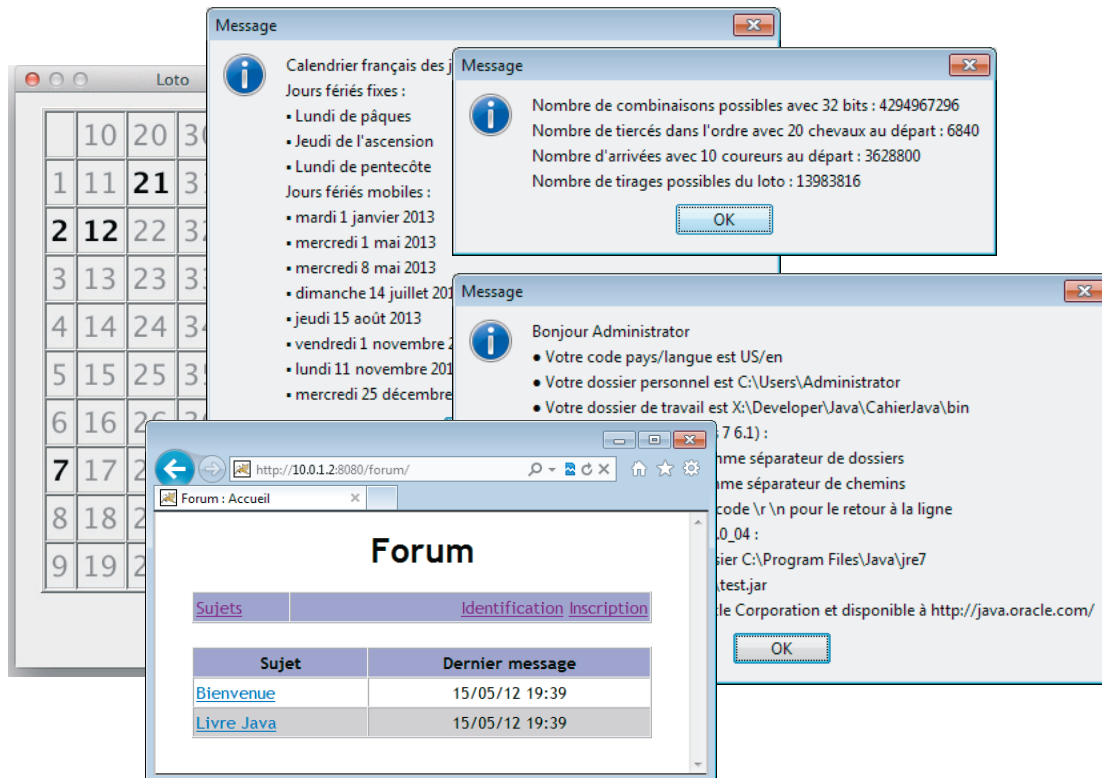
Ensemble des participants au chat • 357

Date de la dernière lecture des messages • 357

Pages JSP de gestion du chat • 357

Arrivée d'un utilisateur dans le chat • 357	Installation • 387
Lecture des participants au chat • 358	Démarrage • 387
Lecture des messages du chat • 359	Création des classes • 387
Ajout d'un message dans le chat • 360	Édition des classes • 388
Départ d'un participant du chat • 361	Compilation et exécution • 388
Interface utilisateur du chat • 361	Eclipse • 388
Threads nécessaires au chat • 366	Installation • 389
Gestion de l'accès aux pages JSP du serveur • 366	Démarrage • 389
Page de lancement de l'applet • 367	Création d'un projet • 389
Intégration du chat au forum de discussion • 368	Création des classes • 389
Synchronisation du module de chat • 369	Édition des classes • 390
États d'un thread • 369	Compilation et exécution • 390
Synchroniser les traitements sur les données partagées • 370	E. Erreurs de compilation les plus fréquentes • 390
De la nécessité de synchroniser... • 370	Symbole introuvable • 391
Synchroniser avec synchronized • 371	Déclaration de classe incorrecte • 392
Chat : synchroniser l'accès à la liste des participants • 372	Déclaration de méthode incorrecte • 392
Synchroniser les traitements dans un ordre déterminé • 375	Modificateur d'accès incorrect • 392
Synchroniser avec wait et notify • 375	Déclaration de variable locale incorrecte • 393
Chat : synchroniser l'envoi des nouveaux messages aux applets • 377	Utilisation de variable incorrecte • 393
En résumé... • 381	Erreur avec return • 393
ANNEXES 383	Erreur dans les conditions des instructions if, for ou while • 394
A. Types de licences logicielles • 383	Équilibre incorrect entre accolades ouvrantes et fermantes • 394
B. Fichiers du forum de discussion • 384	Chaîne littérale non fermée • 394
C. Précisions sur les commentaires javadoc • 386	Commentaire non fermé • 394
D. Mise en route de ConTEXT et d'Eclipse • 386	F. Bibliographie • 395
ConTEXT • 387	G. Glossaire • 396
	INDEX 399

chapitre 1



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

- ▶ Applications isolées
- ▶ Carnet d'adresses
- ▶ Calcul de mensualités d'emprunt
- ▶ Forum de discussion
- ▶ Messagerie instantanée (*chat*)

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...

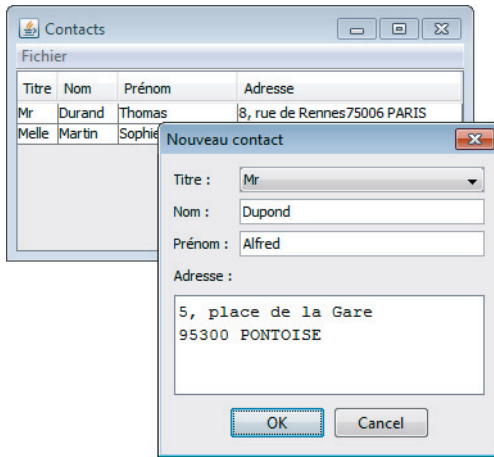


Figure 1-1
Saisie d'un contact dans l'application de carnet d'adresses

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.

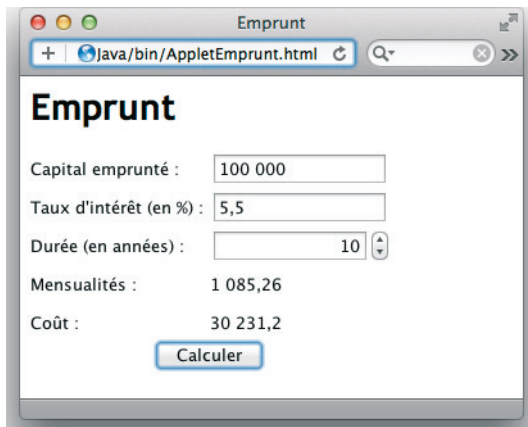


Figure 1-2
Calcul des mensualités d'un emprunt

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 présentés dans cet ouvrage sont lisibles par tous les utilisateurs identifiés ou pas.

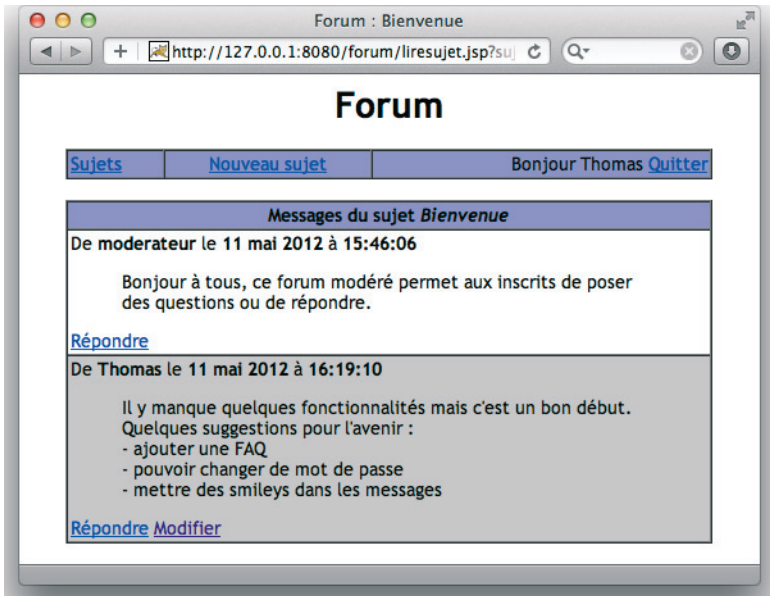


Figure 1-3
Exemple de page du forum affichant
les messages d'un sujet

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 de 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'intègre 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 autres utilisateurs.

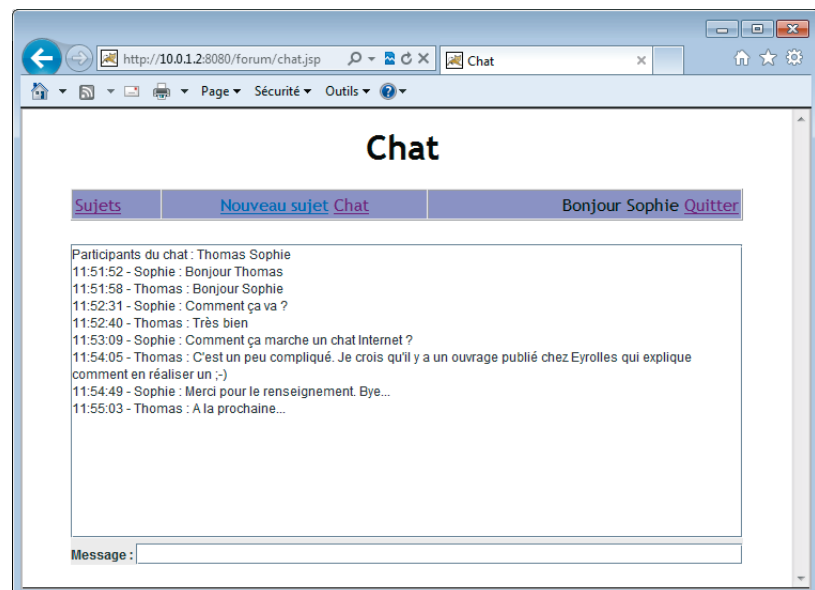


Figure 1-4
Exemple de conversation sur le chat

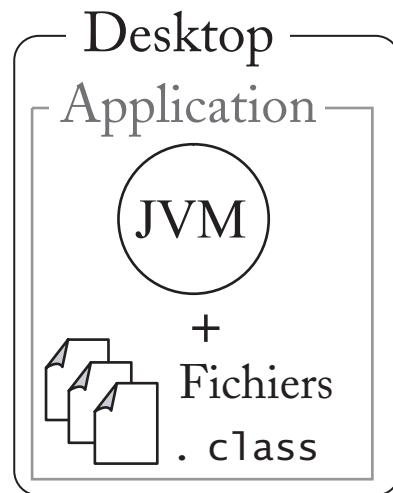
Le chat est développé au chapitre 15, « Messagerie instantanée avec la programmation multitâche » de cet ouvrage.

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

- ▶ Comprendre la démarche objet
- ▶ Vue d'ensemble sur l'architecture Java
- ▶ Installation

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

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.

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.

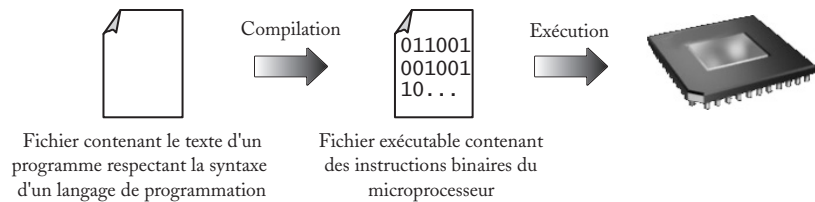


Figure 2-1
Compilation et exécution d'un programme

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 *nombre*s 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.

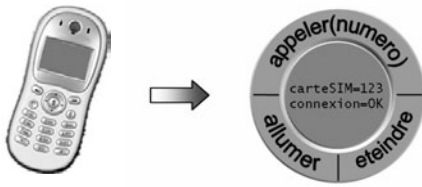


Figure 2-2

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.

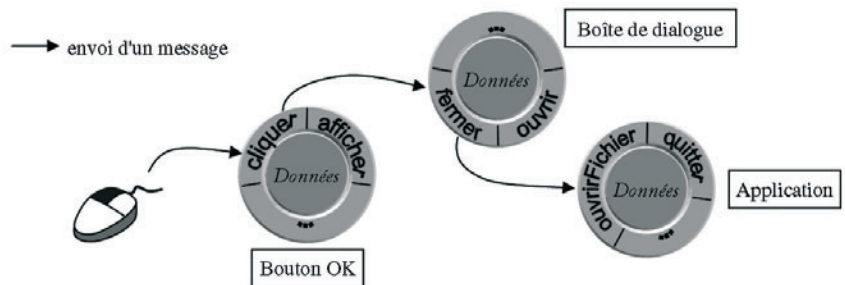
Par exemple, un téléphone portable pourrait être doté d'une interface objet avec les messages suivants :

- 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.

EXEMPLE

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

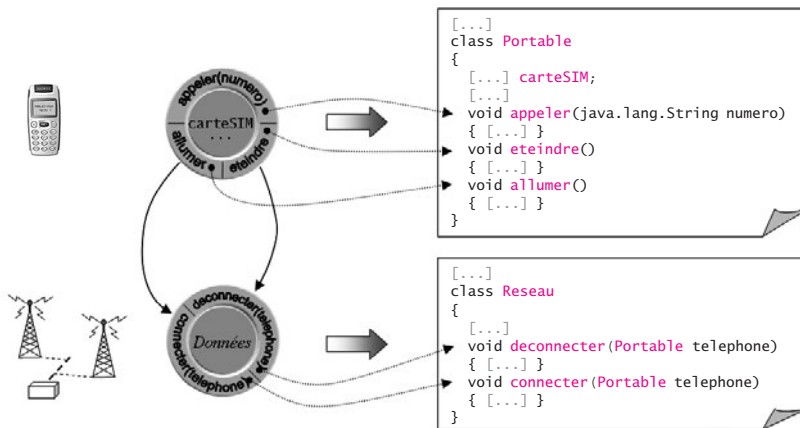


Figure 2-4 Identification des classes correspondant aux objets portable et réseau

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 Terminologie

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

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 !

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).

++ 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.

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.

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.

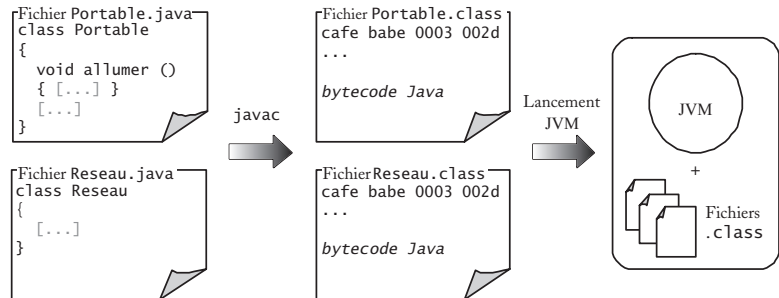


Figure 2-5 Cycle de développement Java

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

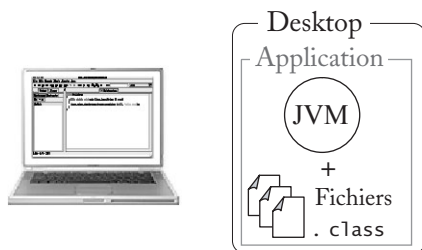


Figure 2-6 Application Java

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
    }
}
```

2 Applet d'un fichier HTML lancée par un navigateur

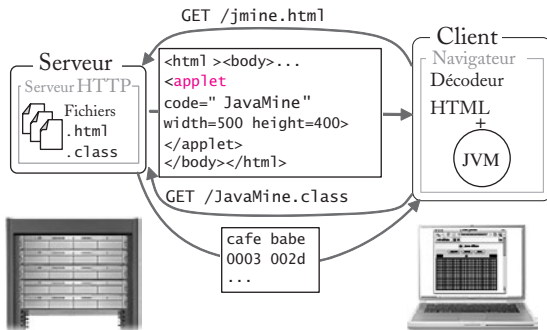


Figure 2-7 Applet Java

Une applet s'exécute dans une page HTML sur une machine cliente raccordée à un serveur web.

La JVM, installée sur la machine cliente, est lancée par le navigateur.

Les fichiers `.class` d'une applet sont installés sur le serveur web et téléchargés par le navigateur.

Le point d'entrée d'une applet est la méthode `init` d'une classe respectant la syntaxe décrite ci-après.

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

3 Servlet lancée par une requête sur un serveur web

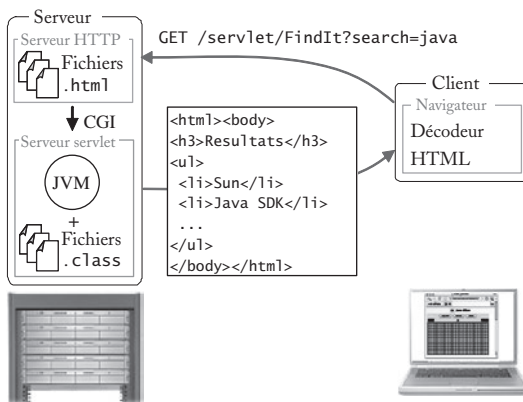


Figure 2-8 Servlet Java

Une servlet s'exécute sur un serveur web pour 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](http://www.oracle.com/technetwork/java) 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.

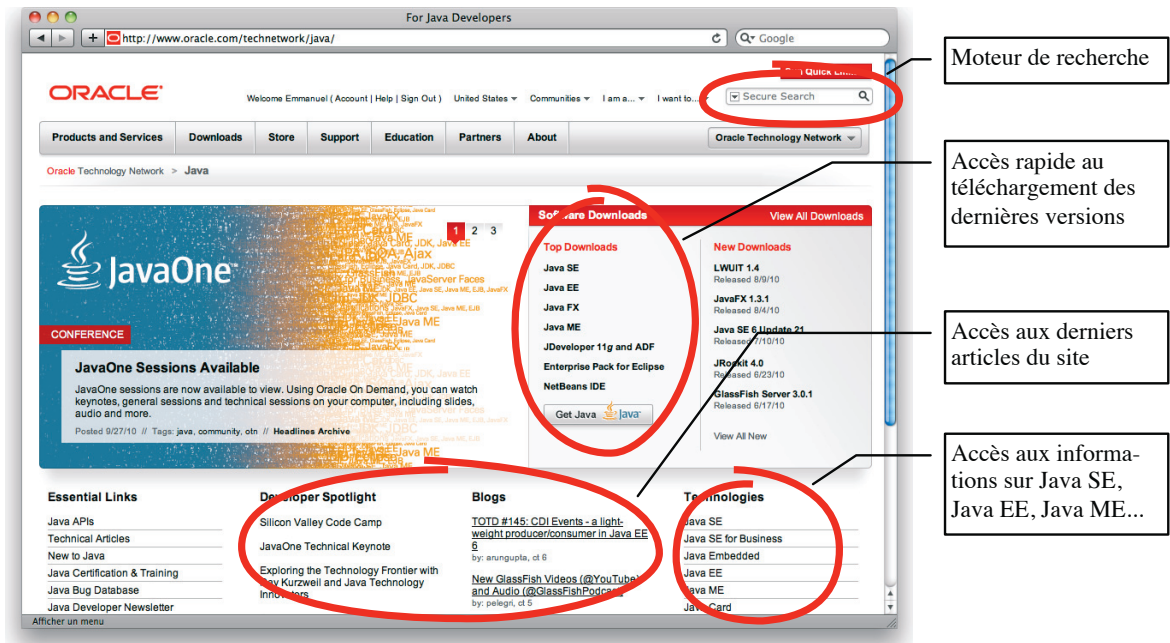


Figure 2-9 Documentation des API Java

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.

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*.

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.

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.

Liste des packages (classement thématique des classes)

Frame principale :

- Au départ, liste détaillée de tous les packages
- Description d'une classe, de ses méthodes et de ses champs après un clic sur une classe
- Hiérarchie des classes
- Index...

- Au départ, liste de toutes les classes
- Liste des classes d'un package après un clic sur un package

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for drawing and performing operations on objects related to two-dimensional geometry.

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) contient(en)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`.

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

l'icônes de la fenêtre de commande

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.

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 débiter 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	<code>dir</code>	<code>ls</code>
Lister les fichiers d'extension <code>.java</code>	<code>dir *.java</code>	<code>ls *.java</code>
Lister les fichiers d'un dossier	<code>dir dossier</code>	<code>ls dossier</code>
Changer de dossier	<code>cd dossier</code>	<code>cd dossier</code>
Copier un fichier	<code>copy fichier1 fichier2</code>	<code>cp fichier1 fichier2</code>
Renommer un fichier ou un dossier	<code>ren fichier1 fichier2</code>	<code>mv fichier1 fichier2</code>
Supprimer un fichier	<code>del fichier</code>	<code>rm fichier</code>
Créer un dossier	<code>md dossier</code>	<code>mkdir dossier</code>
Supprimer un dossier vide	<code>rd dossier</code>	<code>rmdir dossier</code>

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.

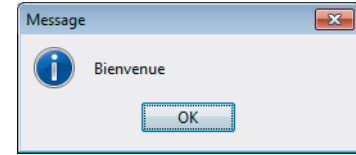


Figure 2–12
Application Bienvenue

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
```

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

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

2 Définition de classe non trouvée (2)

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

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

3 Méthode `main` non trouvée

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

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

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

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 débiter ?

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 *Text-Edit* 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.

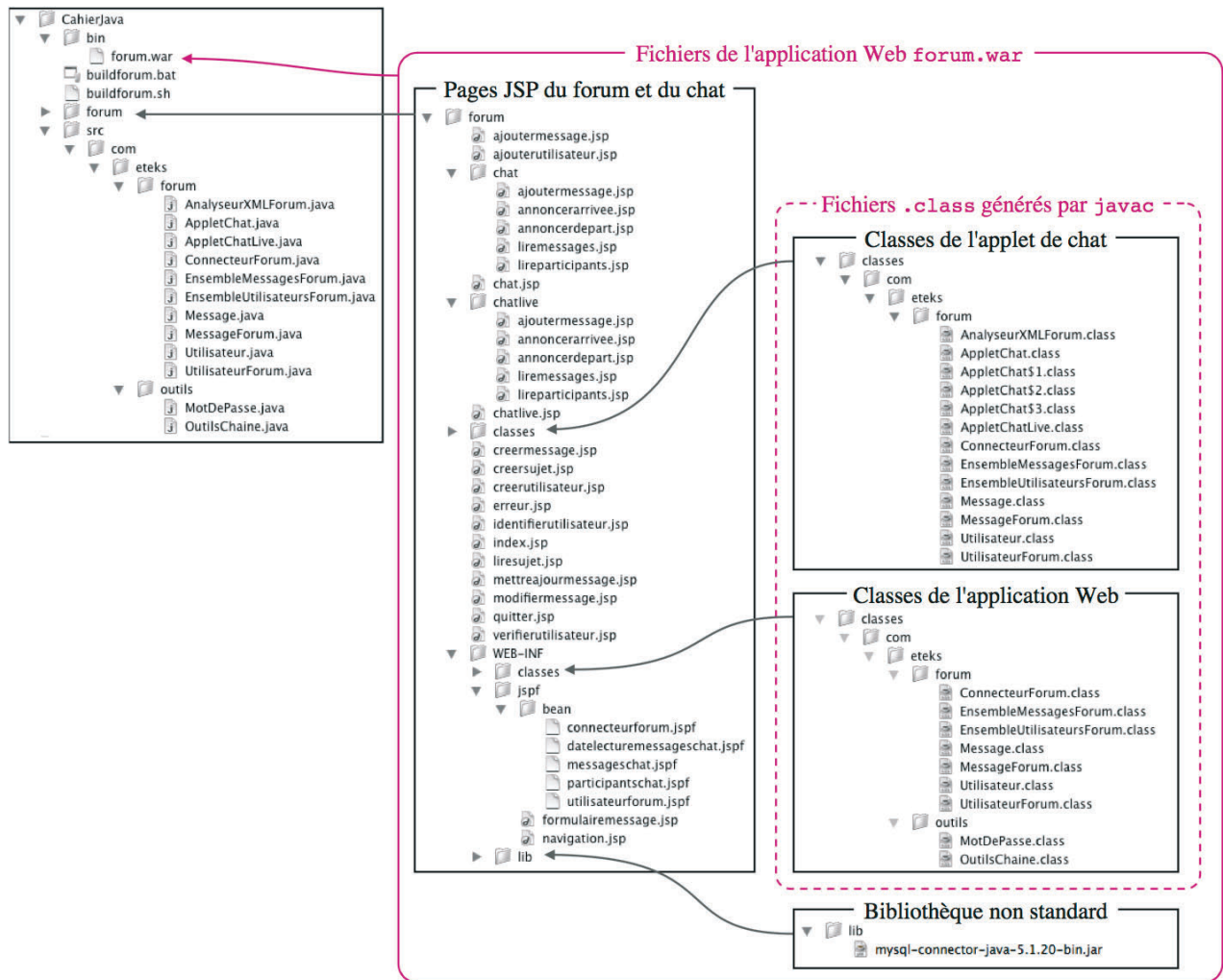
Suite à l'essor des logiciels libres (*free* en anglais à ne pas confondre avec gratuit !), on distingue aujourd'hui quatre grandes catégories de logiciels :

- 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 :

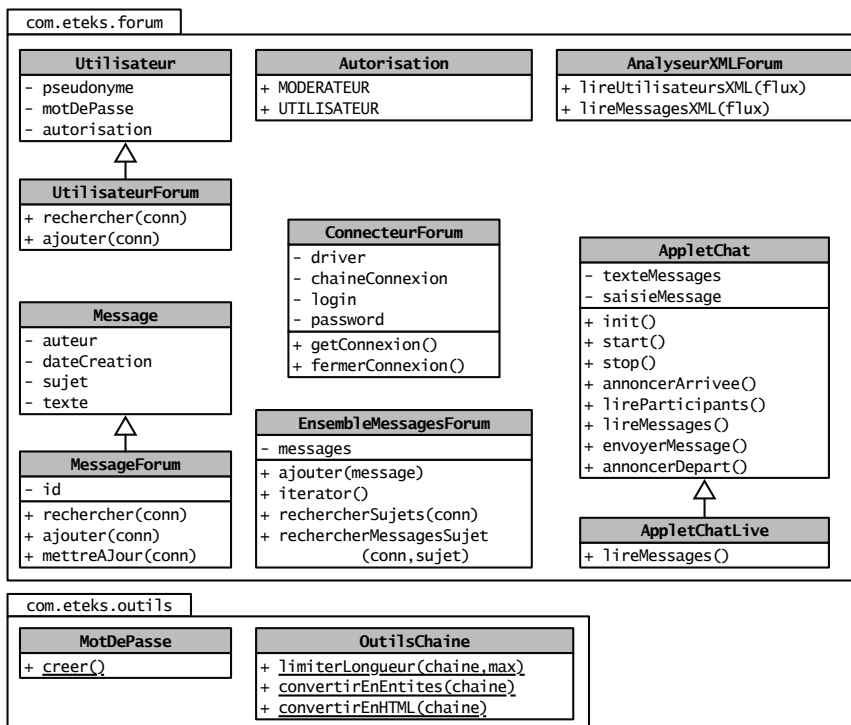
- 1 Compilation des classes nécessaires à l'application web en les rangeant dans le dossier `forum/WEB-INF/classes`.
- 2 Compilation des classes nécessaires à l'applet de chat en les rangeant dans le dossier `forum/classes`.
- 3 Création du fichier d'archive `bin/forum.war` avec le contenu du dossier `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
<code>@author</code> <i>auteur</i>	Décrit l'auteur d'une classe ou d'une interface. Peut être répété pour citer plusieurs auteurs. Exemples : <code>@author Alfred Dupont</code> <code>@author Georges Durand</code>
<code>@version</code> <i>version</i>	Décrit la version d'une classe ou d'une interface. Exemple : <code>@version 1.1.3</code>
<code>@see</code> <i>Classe</i> <code>@see</code> <i>Classe#champ</i> <code>@see</code> <i>Classe#Classe</i> <code>@see</code> <i>Classe#methode</i> <code>@see</code> <i>Classe#methode(typeParam)</i> <code>@see</code> <i>Interface</i> <code>@see</code> <i>Interface#methode</i>	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 : <code>@see com.eteks.outils.Service#Service</code> <code>@see com.eteks.outils.Payant</code> <code>@see com.eteks.outils.Payant#getPrix</code>
<code>@param</code> <i>parametre commentaire</i>	Décrit un paramètre d'une méthode ou d'un constructeur. Exemple : <code>@param prix nouveau prix du produit.</code>
<code>@return</code> <i>commentaire</i>	Décrit la valeur retournée par une méthode. Exemple : <code>@return le prix de ce produit.</code>
<code>@exception</code> <i>ClasseEx commentaire</i>	Décrit les circonstances dans lesquelles une méthode ou un constructeur est susceptible de déclencher l'exception de classe <i>ClasseEx</i> . Exemple : <code>@exception java.lang.IllegalArgumentException si le parametre est negatif ou plus grand que 20.</code>

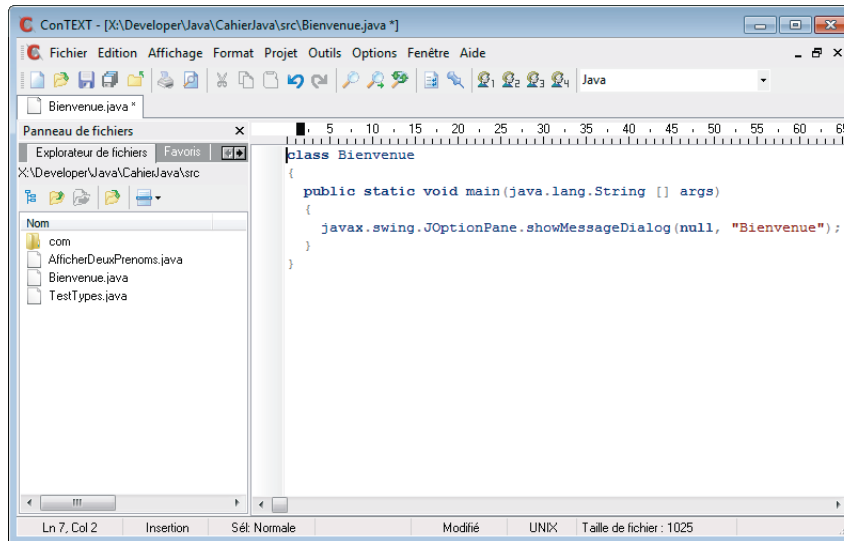
La première phrase d'un commentaire `javadoc` est affichée dans le résumé de la documentation d'une classe.

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 débiter 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

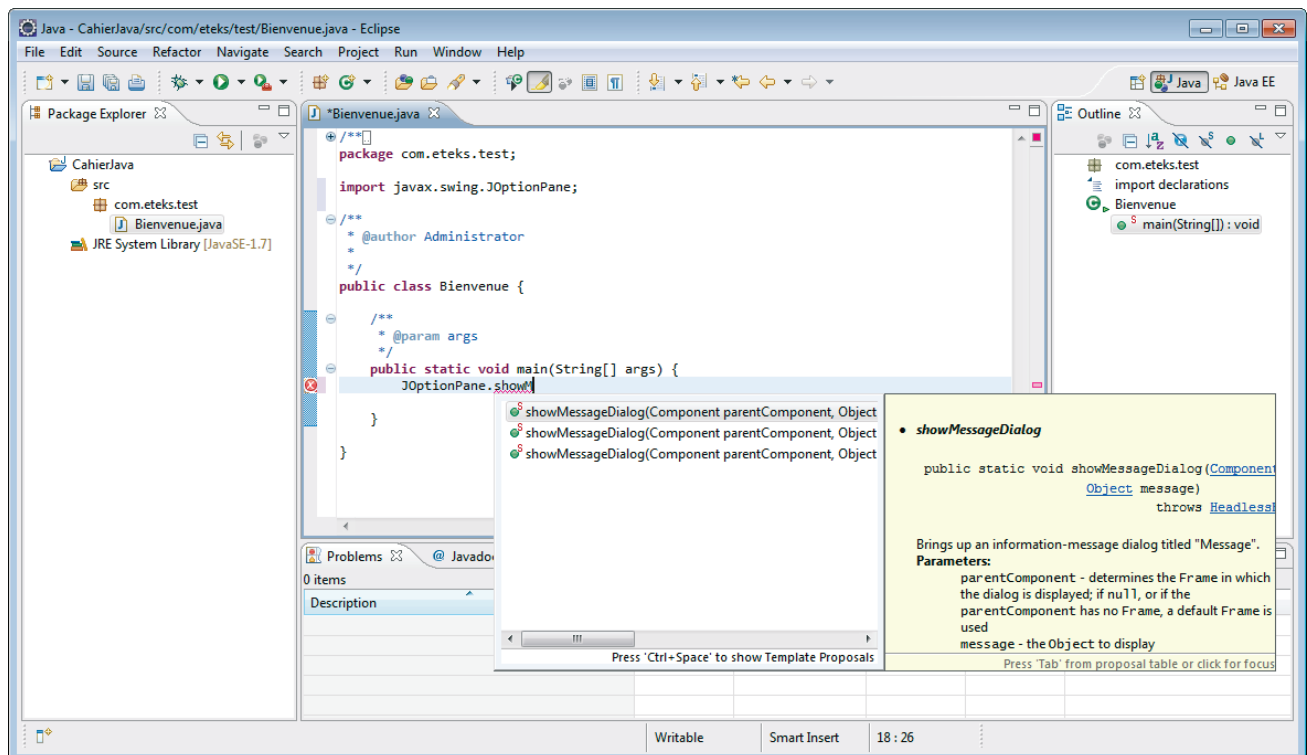
Outre les outils classiques d'un éditeur que vous retrouverez dans le menu *Édition*, quelques outils sont mis à votre disposition dans le menu *Format* pour accélérer l'édition des classes : indentation en bloc de plusieurs lignes, mise en commentaire de code mais aussi l'option *Insérer code depuis 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.

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.

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.

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

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*.

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.

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*.

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.

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`
 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 `import com.eteks.outils.*`; vérifiez par ailleurs que le dossier `com/eteks/outils` contient au moins un fichier `.java`.

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

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

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

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

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

Vérifiez la syntaxe du type primitif (ici `f1oat` à la place de `float`) ou de la classe.

◀ Classe ou type inconnu.

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

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

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

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

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

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

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

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`.

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

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`: <identifiant> **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

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`.

- ◀ 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`.

Déclaration de variable locale incorrecte

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

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

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

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

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

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

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

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

- ◀ 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.

Utilisation de variable incorrecte

ClasseXxxx.java:numLigne: variable x might not have been initialized
 Initialisez la variable `x` avant de l'utiliser.

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

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

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

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

Erreur avec return

ClasseXxxx.java:numLigne: missing return statement

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

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

ClasseXxxx.java:numLigne: unreachable statement

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

- ◀ Instruction impossible à atteindre.

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

Index

: 297
== 56, 61, 102, 104
? 297
61, 63
@Override 86
@WebServlet 270
@XmlType 341

A

abstract 140
accents 31
accès 396
accesseur 40, 116
ActionEvent 219
ActionListener 145, 350–351
actionPerformed 219
addActionListener 219
Adresse (classe) 76
adresse IP 263
affectation 30, 58, 198
opérateur 57
AfficherArticlesFacture 327
AfficherClientFacture 332
AfficherComptes 89
AfficherDeuxPrenoms 36
AfficherEmployesSociete 345
AfficherHeure 219
AfficherMessages 85
AfficherSujets 257
AfficherTitre 93
AfficherUnitesTelecarte 53
AfficherUtilisateursXML 337
Agenda 153

aléatoire 110, 124, 155
AnalyseurXMLForum 333, 362
animation 350
année 115
annotation 86
@Override 86
@WebServlet 270
paramétrage 341
annotations JAXB 341
Ant 51
API, documentation 19
applet 15, 227, 350, 361
init 15
AppletBienvenue 228
AppletChat 362
AppletChatLive 380
AppletEmprunt 230
AppletNouvelles 350
appletviewer (commande JDK) 20, 229
application 14
exécution 23, 52
application web 272
contexte 282
déploiement 275
mise à jour 275, 280
architecture
3 tiers 290
client-serveur 262
ArithmeticException 56, 173
array 121
ArrayIndexOutOfBoundsException 123,
173
ArrayList 130, 155, 255, 283

Arrays 150
arrêt d'un thread 352
ascenseur 209
ASCII (American Standard Code for
Information Interchange) 30
assembleur 11
assert 169
ATTLIST 314
attribut 396
servlet 358
XML 311
Attributes (classe) 327
AUTO INCREMENT 255
autoboxing 111, 158
AutoCloseable 188, 246
Autorisation (énumération) 102
AWT 205

B

balise applet 227, 350, 367
balise XML 311
barre d'outils 209
base de données 236
BeanInfo 281
Bienvenue (classe) 21
bienvenue.jsp 278
BigDecimal 114
bin (dossier des commandes) 49
BLOB 241
bloc d'instructions 58
Boisson (classe) 80, 132, 146, 150
BoissonAlcoolisee 80, 132, 150
BonAnniversaire 120

- boolean 28, 30
- Boolean (classe) 111
- BorderFactory 209
- BorderLayout 208, 213, 230, 362
- boucle 67
- boucle itérative 124, 130, 153, 297, 360
- branchements 63
- break 68
- BufferedInputStream 193
- BufferedReader 199
- build 50
- byte 29
- Byte (classe) 111
- ByteArrayInputStream 187, 380
- ByteArrayOutputStream 191, 380
- bytecode 14

C

- C#
 - abstract 140
 - Array 127
 - base 86
 - cast 84, 147
 - checked 60
 - class (mot-clé) 13
 - CLR 14
 - const 92
 - constructeur 45
 - decimal 114
 - delegate 218
 - enum 94
 - exceptions 168, 174
 - foreach 130, 158
 - héritage 81
 - internal 42
 - is 147
 - liste d'arguments variable 158
 - main 21
 - modificateur d'accès 79
 - MSIL 14
 - namespace 32
 - object 98
 - opérateurs 61
 - out 40, 158
 - override 158
 - polymorphisme 86
 - propriétés 280
 - readonly 91
 - ref 40, 158
 - sealed 91
 - static 90
- string 105
- struct 37
- surcharge des opérateurs 61, 158
- switch 64
- tableau 122, 126
- throw 174
- unsafe 158
- using 52, 158
- virtual 158
- vs Java 158
- while 67

C++

- #include 52
- algorithm 155
- cast 84
- cast automatique 58
- catch 168
- catch (...) 171
- const 91
- constructeur 45
- constructeur par recopie 45
- delete 45
- destructeur 45
- dynamic_cast 84
- édition de liens 14
- enum 94, 158
- exceptions 166, 168, 170, 174
- fonction 13
- for 68
- héritage 81
- héritage multiple 144
- if 63
- inline 42
- listes d'arguments variables 158
- macro 13
- main 21, 128
- méthode virtuelle pure 140, 144
- modificateur d'accès 79
- namespace 32
- new 42, 62
- opérateur new 62
- opérateurs 61
- passage des paramètres 40
- pointeur 29
- pointeur sur fonction 218
- polymorphisme 86
- printf 195
- référence 29, 40
- RTTI 99, 174
- sizeof 28
- string 105
- struct 13
- surcharge des opérateurs 61, 158
- template 158
- this 40
- throw 168, 174
- try 166
- typedef 28
- union 13
- unsigned 28
- using 52
- valeur par défaut des paramètres 47
- variable globale 13
- variable locale 71
- virtual 158
- wchar_t 28
- while 68

calcul

- de mensualité 230
- de mensualités 3

- Calcul (classe) 163
- CalculateurEmprunt 112
- CalculAvecTryCatch 168
- CalculAvecTryCatchFinally 171
- CalculFactorielle 162
- CalculInterets 87
- CalculLignesDeCode 198
- CalculOccurrencesCaractere 189
- CalculPrixTotal 149
- CalculProbabilites 70

caractère

- accentué 30
- char 28
- encodage 297
- spécial 30

- carnet d'adresses 2
- CarnetAdresses 222
- CasierBouteilles 132

casse

- modifier 106

- cast 58, 396
- catch 166, 188, 296
- CaveAVin 132
- CGI (Common Gateway Interface) 266
- chaîne 30, 34, 104
 - accent 31
 - comparaison 105
 - concaténation 61
 - manipulations 105
 - modifier la casse 106
 - opérateur + 61
 - recherche 105

- switch 63
 - vide 30
 - champ 28, 396
 - conventions de nommage 38
 - d'instance 38
 - de classe 89
 - initialisation 38
 - portée 34
 - valeur par défaut 44
 - char 28–29, 104
 - CHAR (SQL) 241, 247
 - Character (classe) 111
 - charAt 105
 - ChargeurRessource 209
 - chat 6, 354
 - synchronisation 372
 - chemin
 - canonique 199
 - relatif 183
 - Class 99, 174, 237
 - class (mot-clé) 39
 - ClassCastException 83–84, 147, 173
 - classe 13, 32, 396
 - abstraite 140
 - ancêtre commun 98
 - anonyme 220
 - charger dynamiquement 174
 - commenter 37
 - d'emballage 111, 150
 - d'exception 179
 - de base 78
 - déclarer 37, 144
 - définir 37
 - explorer 177
 - instanciation 42
 - interne 222
 - IOException 174
 - méthode de classe 89
 - organisation des fichiers 49
 - package 37
 - portée 34
 - sources 34
 - classes (dossier des classes compilées) 49, 272, 292
 - ClassNotFoundException 175, 238, 269
 - classpath 52, 210, 240
 - servlet.jar 273
 - clavier 218
 - ClavierCalculatrice 207
 - clé 134
 - vs indice 134
 - clic 218
 - clone 123
 - code source VI
 - collection 396
 - classe utilitaire 154
 - énumérer les éléments 145, 152
 - interfaces implémentées 155
 - itérateur 131, 152
 - manipulation 131
 - supprimer 131
 - vs tableau 129
 - Collection (interface) 155
 - Collections (classe) 154
 - commande 21
 - option 20, 50
 - commentaire 37–38, 195, 386
 - filtrer 195
 - Comparable 182
 - comparaison 56
 - ComparaisonFigures 141
 - ComparaisonUtilisateurs 103
 - compilation 10, 13, 21, 50, 273, 279, 384
 - erreurs 22, 390
 - vérification des types 58
 - complexType 317
 - composant 204
 - composition 76
 - Compte (classe) 87
 - CompteEpargne 87
 - concaténation 61
 - conception objet 41, 78
 - condition 63, 67
 - ConnecteurForum 248, 291, 295
 - Connection 237, 243
 - Connector/J 239, 273
 - connexion 237, 291
 - console 107, 185
 - constante 92
 - conventions de nommage 92
 - constructeur
 - chaînage 82
 - déclarer 44
 - héritage 82
 - nommage des paramètres 46
 - paramètres 44
 - super-classe 82
 - Constructor (classe) 177
 - Container 205
 - conteneur 204
 - ContentHandler 326
 - ConTEXT 387
 - continue 68
 - contrôle de flux 63
 - conventions VI
 - conventions d'écriture 72
 - conventions de nommage 37–38, 69, 284
 - champ 38
 - classe 37
 - constante 92
 - interface 144
 - méthode 40
 - paramètre 46
 - conventions de programmation 64, 284
 - conversion 58, 396
 - de référence 82, 146
 - en chaîne 61, 106, 232
 - en HTML 105
 - en nombre 111
 - implicite 147
 - types 58
 - ConversionEuro 62
 - ConversionSommeMontantsEuro 62
 - ConversionsReferencesPayant 146
 - ConvertisseurEuro 59
 - cookie 280, 368
 - CREATE 242, 245
 - createStatement 244
 - CreerDocumentSociete 345
 - CreerObjetClasse 176
- ## D
- DatabaseMetaData 247
 - DataInputStream 194, 310
 - DataOutputStream 194, 310
 - date 297
 - comparer 115
 - schéma XML 320, 334, 360
 - Date (classe) 115, 151, 219, 357, 360
 - DATE (SQL) 241
 - DateFormat 115, 120, 151, 219, 297, 362
 - deadlock 378
 - DECIMAL 241
 - DefaultHandler 326
 - DefaultTableModel 222
 - DELETE 242
 - deprecated 115, 297, 366
 - dériver 78, 227, 251, 268
 - design pattern 95
 - DeuxPersonnesUneAdresse 76, 78
 - développement
 - organisation des fichiers 49, 272
 - dictionnaire d'objets 134

- do 67, 189
 - doc (dossier de documentation) 49
 - DOCTYPE 316, 331
 - Document (classe) 332
 - document XML 310
 - bien formé 312
 - valide 313
 - documentation 19, 271
 - DocumentBuilder 326
 - DocumentBuilderFactory 332
 - doGet (point d'entrée de servlet) 15, 268
 - DOM 325, 332
 - générer un document 339
 - vs SAX 325
 - doPost 268
 - dossier
 - chemin 182
 - contenu 182
 - courant 183
 - source 49
 - double 28–29
 - Double (classe) 111
 - DOUBLE (SQL) 241
 - driver 145, 237, 272
 - DriverManager 237
 - DROP 242
 - DTD
 - ATTLIST 314
 - ELEMENT 314
 - ENTITY 315
 - exemple 315, 333
 - syntaxe 315, 323–324
 - utilisation 331
 - vs schéma XML 317
 - durée de vie 89, 396
- E**
- Eclipse 18
 - écouteur
 - Voir aussi* listener 218
 - EditeurTexte 209
 - égalité
 - entre objets 98
 - EJB 258
 - ELEMENT (DTD) 314
 - élément XML 311
 - else 63
 - emballage 111, 155, 397
 - Employe (classe) 343
 - Emprunt (classe) 230
 - encapsulation 33, 396
 - encodage 31
 - encode 297
 - EnsembleMessagesForum 255, 292, 297, 334, 356
 - entité 105, 267, 312
 - enum
 - Voir* Java 5 28
 - énumération 13, 92, 336
 - avantages 94, 148
 - comparer 102
 - déclarer 92
 - switch 63, 93
 - Enumeration (interface) 152
 - EnumerationTitre 93
 - environnements d'exécution 14
 - EOFException 193
 - equals 98, 133, 152
 - vs compareTo 154
 - erreur
 - d'exécution 23
 - dans un document XML 329
 - de compilation 391
 - Error 168
 - Error (classe) 172
 - espace 20
 - espace de noms XML 312
 - ET (opérateur) 57
 - étendre 78, 269
 - études de cas
 - Java 5 VI
 - Java 6 VI
 - événement 218
 - EvenementCalendrier 151
 - Event 218
 - exception 10, 337
 - ArrayIndexOutOfBoundsException 123, 173
 - catégories 172
 - chainage 337
 - ClassCastException 84, 147, 173
 - ClassNotFoundException 175, 238
 - contrôlée 173, 187, 250, 366
 - créer une classe d'exception 179
 - dans une page JSP 277, 295
 - déclencher 168, 250
 - diagnostic 374
 - entrées-sorties 187
 - Error (classe) 172
 - getCause 337
 - IllegalArgumentException 168, 173
 - IllegalMonitorStateException 377
 - IllegalStateException 173
 - initCause 337
 - instructions du bloc try 167
 - intercepter 166, 187, 250
 - interpréter 165
 - InterruptedException 352, 375
 - IOException 188, 269
 - lire le diagnostic d'erreur 165
 - MalformedURLException 265
 - non contrôlée 172
 - NullPointerException 173
 - NumberFormatException 165, 173
 - SecurityException 173
 - ServletException 269
 - SQLException 174, 250
 - stack trace 165
 - StringIndexOutOfBoundsException 173
 - syntaxe 166
 - trace de la pile d'exécution 165
 - UnsupportedOperationException 173
 - Exception (classe) 173
 - executeQuery 244
 - executeUpdate 244
 - exécution 13, 23
 - erreurs 23
 - exit 107
 - expression 58
 - extends 79, 229, 251, 269
- F**
- facture.xml 311
 - fichier
 - chemin 182
 - compilé 49
 - de traduction 200
 - écrire 190
 - exécutable 49
 - lire 186
 - source 37, 49
 - Field (classe) 177
 - Figure (classe) 140
 - File (classe) 182
 - FileFilter 191
 - FileInputStream 187, 337
 - FileOutputStream 191
 - FileReader 186–187
 - FileWriter 191
 - fill 127, 154
 - FilterReader 193

- filtrage de données 191
 - FiltreCommentaires 195
 - final 220
 - finalize 45, 99
 - finally 188, 238, 269, 334
 - float 28–29
 - Float (classe) 45, 111
 - FLOAT (SQL) 241
 - FlowLayout 230
 - flux de données 184
 - for 68, 125, 130, 153, 297
 - for 198
 - for-each 324
 - formulaire HTML 266, 292, 299, 304
 - forName 174, 237–238
 - forum
 - affichage des sujets 257
 - architecture 5
 - barre de navigation 298, 368
 - chat 354
 - création de message 302
 - diagramme de classes 385
 - échange de données 333
 - évolution 300
 - identification 292
 - message 116
 - messages d'un sujet 301
 - modérateur 295
 - modification d'un message 303–304
 - organisation des pages 290, 384
 - page d'accueil 296
 - présentation 4, 288
 - programmation 290
 - scénario 288
 - synchronisation 377
 - utilisateur 99
 - XML 333, 359
 - forward 284, 294
 - fragment 284
- G**
- garbage collector
 - Voir* ramasse-miettes 24
 - GenerateurDocumentXML 339
 - généricité 130, 136, 149
 - getConnection 237
 - getContentPane 207
 - getInt 244
 - getParameter 228, 294
 - getProperty 282
 - getString 244
 - getTables 247
 - getWriter 269
 - global 397
 - Glossaire (classe) 135
 - GregorianCalendar 119, 125
 - GridLayout 207, 213, 220, 230
 - GrilleLoto 156
 - GUI 204
- H**
- hashCode 98, 133, 152
 - HashMap 134
 - HashSet 133, 357
 - HashSet 154
 - héritage 280
 - étendre 227
 - heure 114
 - hexadécimal 11, 29
 - hôte 263
 - HotSpot 24
 - HTML
 - formulaire 266, 274, 292, 299, 304
 - JEditorPane 204, 264
 - JLabel 350
 - JOptionPane 85, 156
 - liens 268, 297, 368
 - servlet 269
 - showMessageDialog 85
 - syntaxe 267
 - tableau 157, 292
 - HTTP
 - GET 268
 - POST 268
 - protocole 262
 - HttpServlet 268
 - HttpServletRequest 268, 278
 - HttpServletResponse 269, 278
- I**
- icône 210
 - standard 211
 - IDE 24
 - IDE (Integrated Development Environment) 31, 212
 - identificateur 28
 - identification 293
 - if 63
 - XSLT 324
 - IHM 204
 - IllegalAccessException 175
 - IllegalArgumentException 168, 173
 - IllegalMonitorStateException 377
 - IllegalStateException 173
 - ImageIcon 210
 - immuabilité 91, 95, 111, 154, 371
 - impasse 378
 - implémentation 12, 40, 396
 - implements 145
 - polymorphisme 82
 - séparer de l'interface 143
 - import 52, 223
 - include 294
 - vs jsp
 - include 284
 - incréméntation 57
 - décrémentation 57
 - indentation 31
 - index 242, 249
 - indice 122
 - vs clé 134
 - init (point d'entrée d'une applet) 15
 - initialisateur static 90
 - initialisation 227
 - constructeur 44
 - valeur par défaut 44
 - inner class 222
 - InputStream 186, 335, 363
 - INSERT 242, 252, 254
 - installation
 - sous Linux 18, 239, 271
 - sous Mac OS X 18, 240, 271
 - sous Windows 17, 239, 270
 - instance 13, 396
 - champ d'instance 38–39
 - méthode d'instance 89
 - instanceof 56, 147
 - InstantiationException 175
 - instruction
 - bloc 58
 - conditionnelle 63
 - de boucle 67
 - de contrôle 63
 - précompilée 250
 - vide 67
 - instruction SQL 241
 - paramétrage 250
 - instructions (bloc) 58
 - int 29
 - INTEGER 241
 - Integer (classe) 53, 111, 150, 155
 - interface 218, 396
 - constantes 148
 - contantes 144

- conventions de nommage 144
 - déclarer 143
 - généricité 149
 - référence 147
 - séparer de l'implémentation 143
 - utilisateur 204, 354
 - internationalisation 115, 200
 - Internet
 - architecture 3 tiers 290
 - client-serveur 262
 - interpréteur 10
 - InterruptedException 352, 375
 - introspection 174
 - IOException 174, 187, 269, 328, 363
 - IP 263
 - ISO-8859-1 297
 - Iterable 257
 - itérateur 133, 152, 244
 - Iterator 360
 - iterator 95, 133, 152–153, 257
- J**
- J2EE (Java 2 Enterprise Edition) 16, 258
 - J2ME (Java 2 Micro Edition) 16
 - J2SE (Java 2 Standard Edition) 16
 - JApplet 227, 230, 350
 - jar (commande JDK) 20, 50, 205, 211
 - Java
 - C# *Voir* C# 10
 - C++ *Voir* C++ 10
 - caractéristiques 10
 - interface 218
 - look and feel 216
 - mise à jour 15
 - mots-clés 28
 - versions 16
 - java (commande JDK) 20
 - Java 5
 - @Override 86
 - annotation 86, 270, 341
 - autoboxing 111, 158
 - boucle itérative 124, 153, 297
 - Class Data Sharing 24
 - enum 28, 63, 92, 102, 148
 - généricité 130, 136, 149
 - import static 90, 148
 - liste d'arguments variable 127
 - Java 7 182
 - AutoCloseable 188, 246
 - catch 176, 328
 - java.nio.file 182
 - notation binaire 29
 - opérateur diamond 130
 - regroupement d'exceptions 176, 328
 - séparateur _ 29
 - switch 63
 - test des chaînes 63
 - try-with-resources 187, 246, 337
 - valeurs littérales 29
 - Java 8 17
 - Java EE 16, 258, 270
 - Java ME 16
 - Java SE 16
 - Java Web Start 232
 - JavaBeans 290
 - portée 281
 - javac (commande JDK) 20
 - javadoc (commande JDK) 20, 50, 386
 - javadoc (documentation) 19
 - JavaMail 300
 - JAXB 318, 340
 - annotations 341
 - marshalling 340, 344
 - unmarshalling 340, 344
 - JAXP 325
 - JButton 207, 219, 230
 - JCheckBox 204, 220
 - JComboBox 213
 - JDBC 145, 236, 273
 - JDK (Java Development Kit) 15
 - JEditorPane 264
 - JFormattedTextField 230
 - JFrame 219, 222
 - JLabel 213, 220, 230, 350
 - JMenu 222
 - JMenuBar 222
 - JMenuItem 222
 - JNLP 232
 - JOptionPane 112, 136, 156, 172, 214, 222
 - HTML 85
 - jour 115
 - JPanel 212, 230
 - JPasswordField 206
 - JRadioButton 204, 220
 - JRE (Java Runtime Environment) 16
 - JScrollPane 209
 - JSP (JavaServer Pages) 276, 284
 - accolades 279
 - balises générales 277
 - classes non standards 280
 - code Java 277
 - erreurs 279, 283
 - errorPage 277, 295
 - exception 278, 295
 - exécution 279
 - forum 288
 - forward 284, 293, 305
 - fragment 284
 - import 277, 296
 - include 284, 291, 293, 298, 305, 357
 - isErrorPage 277
 - out 278
 - page 277, 294, 357
 - param 284, 303
 - request 278
 - response 278, 367
 - servlet 277
 - setProperty 357
 - test 358
 - useBean 281, 291, 295, 356
 - variables prédéfinies 277
 - XML 281
 - JSpinner 230
 - JTable 222
 - JTextArea 213, 361
 - JTextField 213, 361
 - JToolBar 209
 - JVM 14, 107
- K**
- KeyEvent 220
- L**
- langage C#
 - Voir* C# 42
 - langage C++
 - Voir* C++ 42
 - langue de l'utilisateur 200
 - layout
 - BorderLayout 208, 213, 230, 362
 - combinaison 212, 230
 - FlowLayout 230
 - getPreferredSize 205
 - GridBagLayout 212
 - GridLayout 207, 213, 220, 230
 - pack 206
 - suppression 350
 - lettre accentuée 31
 - liaison dynamique 86
 - lib (dossier des bibliothèques non standards) 49, 272, 292
 - licence Apache 270, 383
 - licence GNU GPL 239, 383
 - LinkedList 130

Linux 18, 211, 239, 271
 liste chaînée 130
 liste d'arguments variable 127
 liste des fichiers 183
 liste ordonnée 130
 listecourses.jsp 283
 listener 218, 355
 littéral *Voir* valeur littérale 28
 Locale 200
 logarithme 110
 logs 280
 long 29
 Long (classe) 111
 look and feel 216
 Loto 155

M

Mac OS X 18, 24, 109, 216, 240
 machine virtuelle Java 15, 107
 main (point d'entrée d'une application) 14, 21
 MalformedURLException 265
 manifest 205
 marshallng 340, 344
 Math (classe) 110
 maximum 28, 111
 membre 396
 mémoire 42, 396
 mémoire tampon 192
 message 11, 35
 Message (classe)
 enregistrement 253
 lecture 253
 modification 253
 recherche 253
 XML 333
 MessageForum 253, 280, 292, 334, 357
 enregistrement 305
 lecture 301
 modification 306
 recherche 297, 301
 table 247
 Method (classe) 177
 méthode 397
 conventions de nommage 40
 d'instance 39, 89
 de classe 89
 déclarer 39
 implémentation 40
 paramètres 40
 portée 34

 redéfinir 84
 surcharge 47
 métrique 198
 MIME 269
 minimum 28, 111
 modificateur 396
 modularité 33
 mois 115
 mot-clé 28
 MotDePasse 124, 290, 300
 MouseEvent 220
 multitâche 350
 multithreads 268, 350
 mutateur 40, 116, 281
 MySQL
 connexion 238
 installation 239

N

native2ascii 31
 navigateur
 applet 229
 new 42, 58
 next 152
 NombreEntier (classe) 64
 NombresEnToutesLettres 66
 nommage (conventions) 37
 notify
 vs notifyAll 379
 null 30
 NullPointerException 173
 Number 114
 Number (classe) 111
 NumberFormat 230
 NumberFormatException 165, 173

O

Object 98
 equals 98
 hashCode 98
 toString 99
 ObjectInputStream 194, 310
 ObjectOutputStream 194, 310
 objet 11, 397
 comparer 98, 150
 créer 42
 forme textuelle 99
 initialisation 44, 80
 opérateurs 56
 affectation 57
 arithmétiques 56
 bit à bit 57

 comparaison 56
 diamond 130
 incrémentatoin 57
 logiques 57
 priorité 61
 ternaires 63

Oracle 16
 ORDER BY 243, 255–256
 ordre chronologique 150, 255
 organisation des fichiers 49, 272
 OU (opérateur) 57
 out 107, 278
 OutilsChaine 105, 292
 OutilsFichier 183, 198
 OutputStream 190, 269
 overload 47, 86
 override 84, 86
 OXM 340

P

package 32, 37, 52
 conventions de nommage 37
 panneau à ascenseurs 209
 PanneauContact 213, 222
 paquetage 397
 paramètre
 d'une applet 227, 350, 367
 portée 71
 paramètres d'une annotation 341
 parenthèses 40
 parse 111–112
 parser XML 312, 326
 PATH 17
 Payant (interface) 144, 149
 PDF 278
 Personne (classe) 76
 perte de précision 58
 pile d'exécution 162, 374, 397
 pipeline 185
 polymorphisme 82, 397
 port 263
 portabilité 10, 24, 236, 297
 portée 34, 79, 396
 application 281, 294–295
 page 281, 299, 305
 par défaut 34, 79
 paramètre 71
 request 281, 303
 session 281, 291, 295, 298, 305
 variable locale 71
 prédicat XPath 323

- Preferences 201
 - PreparedStatement 250
 - print 107, 194
 - println 107, 194
 - PrintStream 107, 195
 - PrintWriter 194, 269
 - private 34, 79, 222, 397
 - PrixTotalServices 47
 - Probabilite (classe) 69, 89
 - procédurale (programmation) 11
 - procédure stockée 243
 - processus 350, 397
 - programmation orientée objet 11, 35
 - programme CGI 266
 - prologue XML 310
 - propriété 280
 - propriétés système 107
 - ProprietesJVM 109
 - protected 38, 79, 222, 397
 - protocole 262
 - proxy 368
 - public 34, 79, 222, 397
- R**
- raccourci clavier 222
 - ramasse-miettes 10, 24, 45, 99, 158, 360, 396
 - RandomAccessFile 186
 - read 186, 195
 - Reader 185, 310
 - RechercheFichiers 184
 - récurtivité 162, 184
 - redéfinition 84, 220, 397
 - exception contrôlée 366
 - vs surcharge 86
 - référence 84, 397
 - conversion 82, 146
 - réflexion 174
 - relation
 - a un 76, 95
 - est un 78, 95
 - répertoire
 - Voir* dossier 182
 - request 278, 294
 - ResourceBundle 200
 - response 278
 - ressource 200, 210
 - ResultSet 244, 255
 - retour à la ligne 20, 105
 - return 39, 68
 - réutilisation
 - composition 76
 - héritage 78
 - limiter avec final 91
 - polymorphisme 82
 - run 353, 364
 - Runnable 354
 - RuntimeException 172
- S**
- saisie 112
 - SaisieContact 214
 - SaisiePseudonymeMotDePasse 206
 - SAX 310, 325, 379
 - vs DOM 325
 - SAXException 327–328, 333, 335
 - SAXParser 330
 - SAXParserFactory 330
 - schéma XML 317, 319–320
 - element 317
 - format des dates 320, 334, 360
 - générer 346
 - syntaxe 317
 - types 317
 - vs DTD 317
 - script CGI 266
 - SDK (Software Development Kit) 15
 - sources des classes 34
 - SecurityException 173, 182, 228
 - SELECT 242, 255
 - sérialisation 193
 - serveur
 - client-serveur 262
 - serveur web 15
 - Service (classe) 46, 150
 - servlet 15, 268, 358
 - compilation 273
 - cycle d'exécution 274
 - développement 275
 - généralités 268
 - inconvenients 278
 - initialisation 274
 - ServletBienvenue 269
 - ServletException 269
 - session
 - expiration 283
 - sessionId 367
 - setDefaultCloseOperation 206
 - setProperty 282, 295
 - setSize 212
 - SGBD 236, 290, 295
 - short 29
 - Short (classe) 111
 - showConfirmDialog 172, 214, 222
 - showInputDialog 112
 - showMessageDialog 136, 156
 - shuffle 154
 - signature 84, 397
 - simpleType 317
 - sleep 352
 - Societe (classe) 342
 - sort 127, 133, 154
 - sortie standard 107, 185
 - souris 218
 - sous-classe
 - déclarer 79
 - initialisation 80
 - spécialiser 86
 - SQL 241
 - SQLException 174, 238
 - src (dossier des sources) 49
 - stack
 - Voir* pile d'exécution 162
 - Statement 244
 - Statement (interface) 244
 - static 89, 222, 397
 - StAX 325
 - stream 185
 - String 30, 34, 95, 104, 150
 - StringBuffer 104
 - StringBuilder 104
 - StringIndexOutOfBoundsException 173
 - Sun Microsystems 16
 - super 86
 - super-classe
 - racine de la hiérarchie 98
 - SuppressionCommentairesFichier 197
 - sur fonction 218
 - surcharge 47, 397
 - vs redéfinition 86
 - surcharge des opérateurs 61
 - Swing 24
 - ajout de composant 205
 - création de menus 222
 - gestion événementielle 217
 - interaction utilisateur 217
 - layout 205
 - look and feel 212, 216
 - mise en page 205, 230
 - présentation des composants 204
 - tableau 222
 - SwingSet 205
 - switch 63, 93, 196

synchronisation 99, 268, 369–370
 synchronized 155, 371
 System (classe) 107, 127, 185, 228
 système
 obtention des propriétés 107

T

table 236, 242
 tableau
 boucle itérative 124
 comparer 126
 copier 127
 de type objet 122
 déclarer 122
 énumérer 124
 exception 123
 manipulation 126
 Math 124
 multidimensionnel 126
 organisation 122
 Swing 222
 tri 127
 vs collection 129
 TableModel 222
 tâche 397
 pile d'exécution 162
 tas 162, 396
 Telecarte50 (classe) 40, 52
 TelecarteEmpruntee 42
 template
 XSLT 324
 TestConnexionJDBC 238
 TestTypes 31
 this 40, 86, 232
 thread 397
 chat 366
 états 369
 pile d'exécution 162
 servlet 268
 synchronisation 369
 Thread (classe) 352, 366
 throw 168, 335
 Throwable 278
 throws 173, 248, 269
 TicketDeCaisse 149
 TIME 241
 timeout 283
 Timer 350
 TIMESTAMP 241, 249
 TimeZone 115
 TirageLotoAvecClasseAnonyme 220

TirageLotoSansClasseAnonyme 220
 Tomcat
 configuration 275
 démarrage 271
 installation 270
 manager 275
 startup 271
 webapps 279
 toString 99, 116
 traduction 200
 transaction 243
 transformation XML 338
 transformation XSL 322
 TransformerFacture 338
 transtypage 58, 396
 TreeMap 134
 TreeSet 133, 154
 tri 133, 154
 trigonométrie 110
 TriMots 128
 try 166, 188, 238, 296
 try-with-resources 187, 337
 typepage 28
 type de données
 conversion 58
 objet 29
 primitif 28
 SQL 241
 type MIME 269

U

UML 41, 385
 Unicode 30, 108
 unmarshalling 340, 344
 UnsupportedOperationException 173
 UPDATE 242, 255
 URI 313
 URL (Uniform Resource Locator) 187,
 263, 297, 313
 connexion 365
 encodage 266
 manipuler 264
 relative 274, 367
 URLEncoder 297, 364
 useBean 281, 291, 295, 356
 UTF-8 310
 Utilisateur (classe) 100, 337
 XML 333, 359
 UtilisateurForum 251, 291, 334, 357
 enregistrement 299
 lecture 294

recherche 294
 session 291, 295, 305
 table 247

V

valeur littérale 28–29
 ValeursMathematiques 110
 validation 314, 329
 ValiderDocumentXML 329
 valueOf 336
 VARCHAR 241, 247
 variable
 affecter une valeur 30, 58
 déclarer 28
 locale 28, 34
 nom 28
 opérateurs d'affectation 57
 portée 34
 verrou 375
 void 48

W

W3C 311
 war 279
 warning 136
 web.xml 270
 WEB-INF 272, 368
 WHERE 242, 255
 while 67, 154
 WindowEvent 220
 Windows 17, 24, 109, 201, 216, 239, 270
 wrapper 111, 397
 Write Once Run Anywhere 10
 Writer 310

X

XML
 analyseur 312, 379
 attribut 311
 balise 311
 correspondance objet 340
 DOM *Voir* DOM 325
 DTD *Voir* DTD 313
 DTD vs schéma XML 317
 élément 311
 entité 312
 espaces de noms 312
 générer un document 339
 JAXB 340
 mapping objet 340
 parser 312
 recherche 323

SAX *Voir* SAX 325
schéma 317
schéma XML vs DTD 317
syntaxe 310
transformation 322, 338
type des données 317
XPath 323

XSL 322
XPath 323
XSLT 324
XSLT (Extensible StyleSheet Language
Transformation) 311, 322, 324
éléments 324
instructions de contrôle 324

stylesheet 323
x-www-form-urlencoded 266, 297

Z
ZIP 192

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