

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.

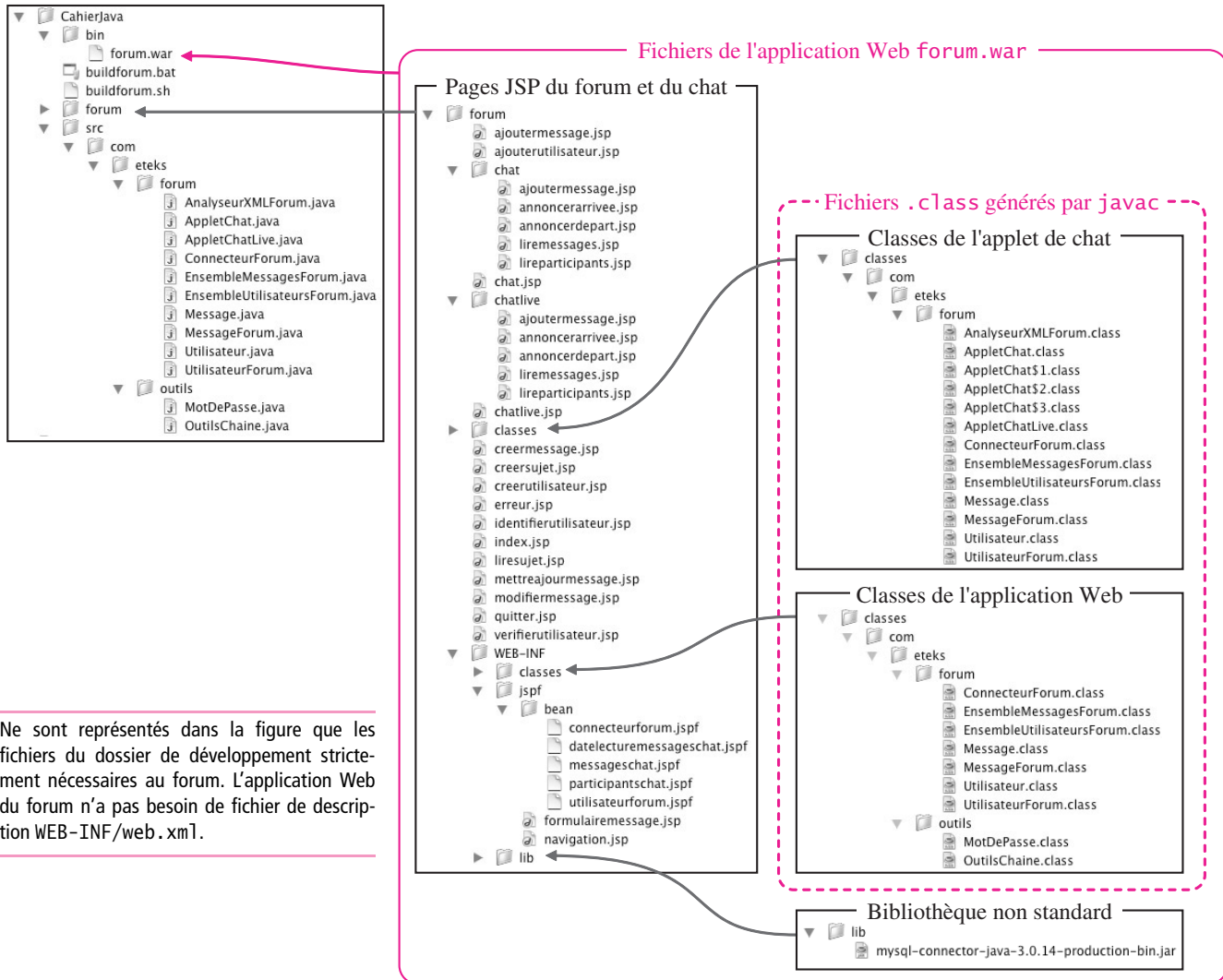
Suite à l'essor des logiciels libres (*free* en anglais à ne pas confondre avec gratuit !), on distingue aujourd'hui quatre grandes catégories de logiciels :

- Les logiciels du domaine public. Ces logiciels 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. Ces logiciels 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 logiciels 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.



Ne sont représentés dans la figure que les fichiers du dossier de développement strictement nécessaires au forum. L'application Web du forum n'a pas besoin de fichier de description WEB-INF/web.xml.

Mise en route du forum

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

La génération 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.

FORUM buildforum.sh

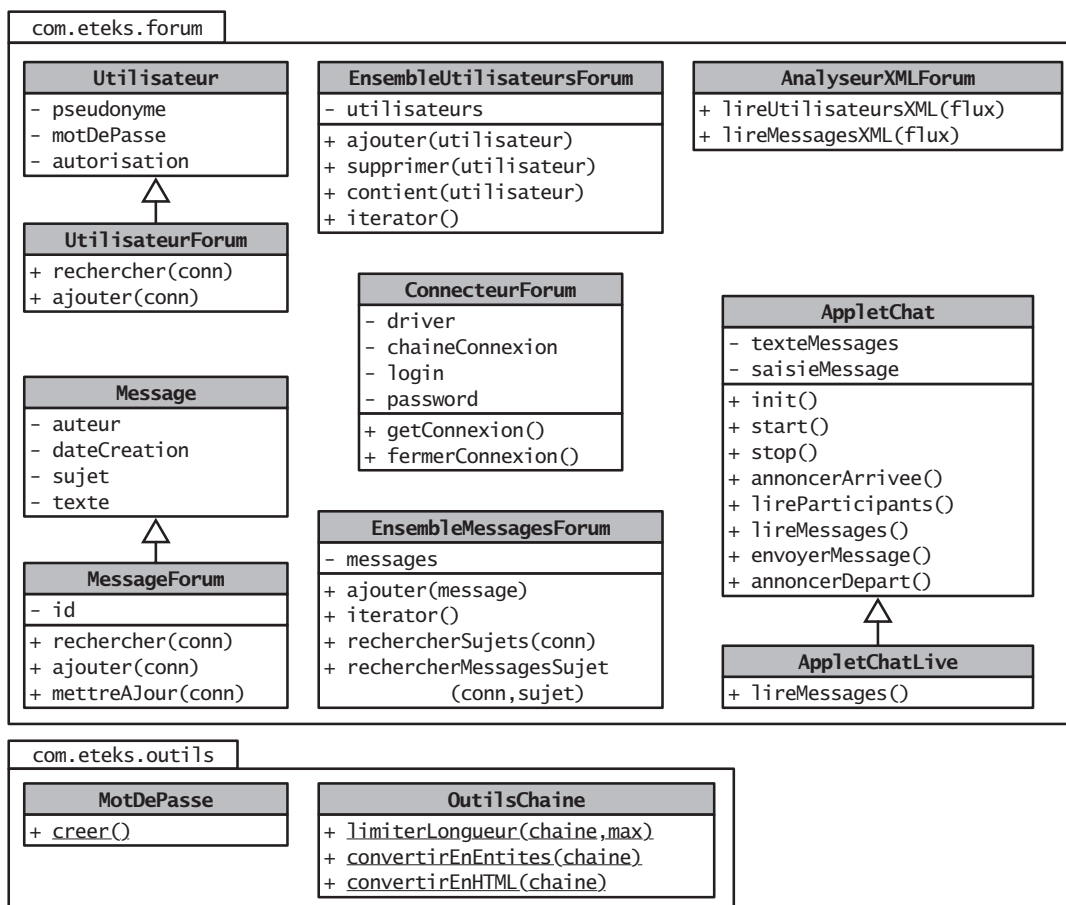
```
javac -sourcepath ./src -d ./forum/WEB-INF/classes
➔ ./src/com/eteks/forum/UtilisateurForum.java
➔ ./src/com/eteks/forum/EnsembleUtilisateursForum.java
➔ ./src/com/eteks/forum/EnsembleMessagesForum.java
➔ ./src/com/eteks/outils/MotDePasse.java
➔ ./src/com/eteks/outils/OutilsChaine.java

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

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

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

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



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 générée.

► <http://java.sun.com/j2se/javadoc/>

Balise javadoc	Usage
<code>@author <i>auteur</i></code>	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 <i>version</i></code>	Décrit la version d'une classe ou d'une interface. Exemple : <code>@version 1.1.3</code>
<code>@see <i>Classe</i></code> <code>@see <i>Classe#champ</i></code> <code>@see <i>Classe#Classe</i></code> <code>@see <i>Classe#methode</i></code> <code>@see <i>Classe#methode(typeParam)</i></code> <code>@see <i>Interface</i></code> <code>@see <i>Interface#methode</i></code>	Crée dans la documentation générée 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 <i>parametre commentaire</i></code>	Décrit un paramètre d'une méthode ou d'un constructeur. Exemple : <code>@param prix nouveau prix du produit.</code>
<code>@return <i>commentaire</i></code>	Décrit la valeur retournée par une méthode. Exemple : <code>@return le prix de ce produit.</code>
<code>@exception <i>ClasseEx commentaire</i></code>	Décrit les circonstances dans lesquelles une méthode ou un constructeur est susceptible de déclencher l'exception de classe <code>ClasseEx</code> . 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.

JAVA 5.0 Métadonnées ou annotations

Les commentaires javadoc sont exploitées par des applications Java spéciales appelées des doclets, dont le générateur de documentation au format HTML du JDK est le représentant le plus connu. Un outil comme XDoclet se sert de la possibilité de définir des balises personnalisées afin de simplifier la génération de fichiers de configuration Hibernate, J2EE... De façon similaire, les annotations introduites dans Java 5.0 permettent d'ajouter des informations déclaratives avant la déclaration d'une classe, d'un champ, d'une méthode ou d'un constructeur grâce à des balises qui débutent aussi par le symbole `@` (comme l'annotation `@Override` placée avant une méthode redéfinie qui force `javac` à vérifier sa signature).

► <http://xdoclet.sourceforge.net/>

► <http://adiguba.developpez.com/tutoriels/java/tiger/annotations/>

D. Contenu du CD-Rom d'accompagnement

Le CD-Rom d'accompagnement contient les études de cas présentées dans cet ouvrage et les outils nécessaires à leur exécution, c'est-à-dire le JDK, MySQL et Tomcat. Ce CD-Rom contient aussi ConTEXT, un éditeur de textes pour écrire vos premiers programmes Java sous Windows, ainsi que JBuilder 2005 Foundation et Eclipse 3.1 deux des IDE les plus puissants du marché.

Études de cas de l'ouvrage

Le dossier CAHIER contient un fichier compressé des études de cas de cet ouvrage à destination de chacun des systèmes sur lesquels elles ont été testés :

- CahierJava.zip pour Windows à décompresser avec l'outil de votre choix ou avec la commande `jar xf CahierJava.zip` (contient les fichiers de commandes .bat des applications) ;
- CahierJava.tar.gz pour Linux à décompresser avec la commande `tar xzf CahierJava.tar.gz` (contient les fichiers de commandes .sh des applications) ;
- CahierJava.dmg pour Mac OS X à installer en lançant l'application Installation Cahier Java après avoir double-cliqué sur cette image de disque (contient les fichiers de commandes .sh et les dossiers .app des applications).

JDK

Le dossier JDK contient les programmes d'installation du JDK version 1.5.0_06 pour Windows 98/ME/2000/XP et Linux x86, ainsi que la documentation des API de la version 1.5 du J2SE. L'installation du JDK et de sa documentation est décrite au chapitre 2.

Il est rappelé que le JDK est préinstallé sous Mac OS X.

MySQL

Le dossier MYSQL contient les programmes d'installation du SGBD MySQL version 5.0.18 pour Windows 95/98/ME/2000/XP, Linux x86 et Mac OS X, ainsi que le fichier `mysql-connector-java-3.1.12.zip` qui contient le driver JDBC MySQL/Connector J version 3.1.12. L'installation de MySQL et de son driver JDBC est décrite au chapitre 11 « Connexion à la base de données avec JDBC ».

Tomcat

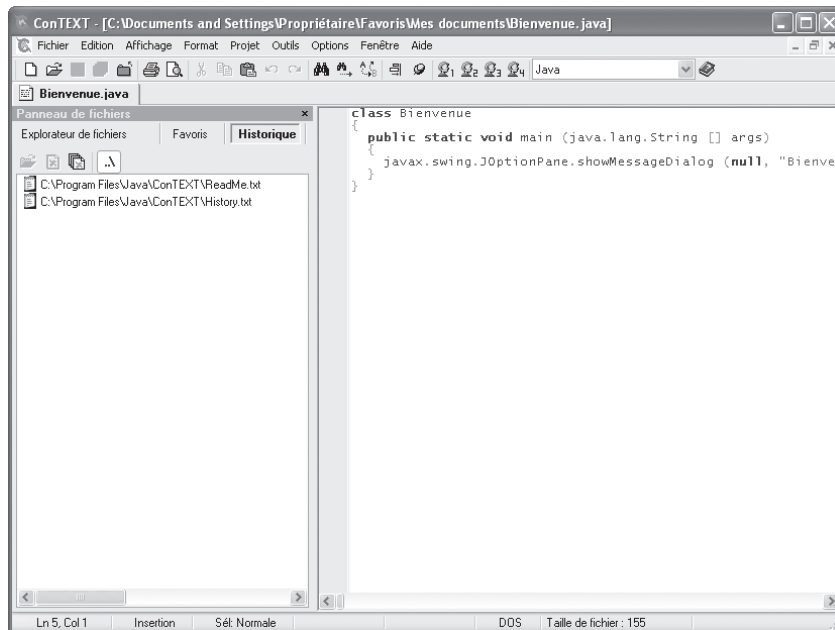
Le dossier TOMCAT contient le fichier `jakarta-tomcat-4.1.31.zip` d'installation de Tomcat pour tout système. L'installation et le lancement de Tomcat est décrite au chapitre 12 « Programmation Web avec les servlets, JSP et JavaBeans ».

ConTEXT est diffusé sur le CD-Rom d'accompagnement avec l'autorisation expresse de son auteur Eden Kirin.

<http://www.context.cx/>

ConTEXT

Le dossier CONTEXT contient le programme d'installation CONTEXTsetup.exe de l'éditeur ConTEXT version 0.98.3 pour Windows. ConTEXT est un éditeur gratuit suffisant pour débiter la programmation en Java ou pour éditer des programmes sur une configuration matérielle ancienne.



Installation

Lancez ensuite le programme d'installation CONTEXTsetup.exe puis laissez-vous guider. 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 Editeur 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.

Édition des classes

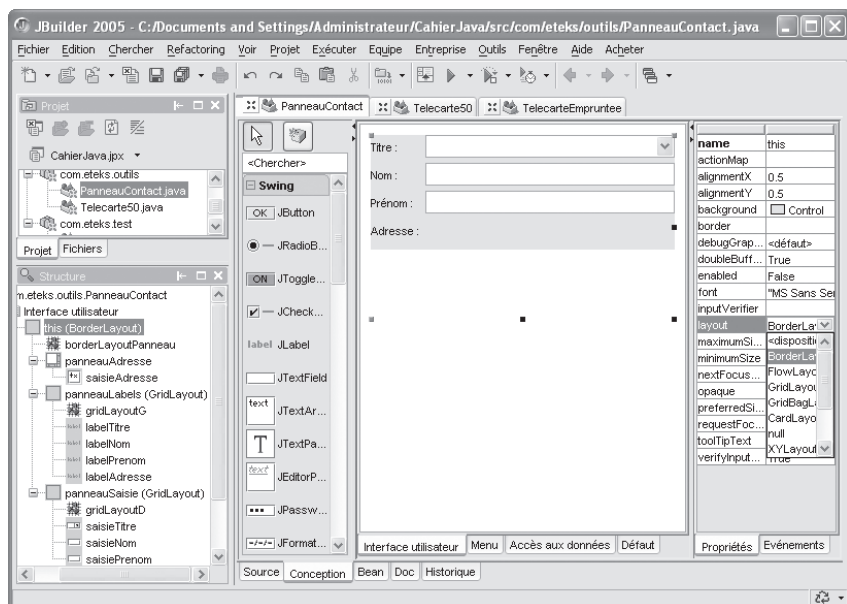
Outre les outils classiques d'un éditeur que vous retrouverez dans le menu Edition, quelques outils sont mis à votre disposition dans le menu Format pour accélérer l'édition des classes : indentation en bloc de plusieurs lignes, mise en commentaire de code mais aussi l'option Insérer code depuis modèle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

Borland JBuilder 2005 Foundation

Le dossier JBUILDER contient les programmes d'installation de l'IDE JBuilder 2005 Foundation de Borland pour Windows, Linux et Mac OS X. JBuilder 2005 Foundation est un IDE gratuit qui vous permet de développer des applications Java à caractère non commercial.



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.

POUR ALLER PLUS LOIN

JBuilder 2005 Developer et Enterprise

JBuilder 2005 Developer et JBuilder 2005 Enterprise, versions plus riches en fonctionnalités, sont disponibles en version d'évaluation sur le site de Borland. Leurs possibilités apparaissent en grisé dans les menus de la version JBuilder 2005 Foundation.

Les documentations annexes à JBuilder ne sont pas incluses sur le CD-Rom d'accompagnement.

SOUS MAC OS X Installation sous Mac OS 10.4

Si le programme d'installation de JBuilder ne fonctionne pas sous Mac OS 10.4, configuré avec le JDK 1.5 par défaut, il vous faudra effectuer les ajustements suivants : dans l'application Java Preferences du dossier Applications/Utilitaires/Java/J2SE 5.0, positionnez l'élément J2SE 1.4.2 en tête de la liste Réglages de moteur d'exécution de l'application Java. Ensuite, dans une fenêtre de Terminal, modifier le JDK par défaut en lançant les commandes suivantes :

```
cd /System/Library/Frameworks/
    ➤ JavaVM.framework/Versions/
sudo rm CurrentJDK
sudo ln -s 1.4.2 CurrentJDK
Pour revenir par défaut au JDK 1.5, remplacer la
dernière ligne par :
sudo ln -s 1.5.0 CurrentJDK
```

B.A.-BA Notion de projet au sein d'un IDE

La notion de projet est très importante dans un IDE : elle correspond en fait à un dossier de développement dans lequel vous créez et compilez les fichiers source d'une application (ou de plusieurs) et regroupe toutes les options de compilation et d'exécution des commandes javac et java, qui sont lancées indirectement par les menus d'un IDE.

POUR ALLER PLUS LOIN

Autres options d'un projet

D'autres options sont disponibles sur un projet, comme celles relatives au formatage du code pour la position des accolades, l'indentation, la gestion des retours à la ligne... Vous retrouverez ces options dans la boîte de dialogue affichée en sélectionnant l'élément Propriétés du projet... du menu Projet.

Systèmes supportés

- Windows NT/2000/XP
- Linux, notamment Red Hat Linux 7.3 ou Red Hat Enterprise Linux 2.1
- Mac OS X, version 10.3

Installation

Sous Windows et Mac OS X, décompressez le fichier `jb2005_fnb_systeme.zip` avec l'outil de votre choix ou avec la commande (si vous avez installé le JDK) :

```
jar xf jb2005_fnd_systeme.zip
```

Sous Linux, décompressez le fichier `jb2005_fnb_linux.tar.gz` avec l'outil de votre choix ou avec la commande :

```
tar xzf jb2005_fnd_linux.tar.gz
```

Sous Windows, Linux et Mac OS X, lancez ensuite le programme d'installation `install` situé dans le dossier créé puis laissez-vous guider. JBuilder 2005 s'installe dans le dossier de votre choix avec les outils de développements Java.

Démarrage

Lancez le programme Borland JBuilder 2005 Foundation. Au premier lancement, JBuilder vous demande si vous voulez associer à cet IDE les fichiers d'extension `.java` `.class` `.jpr` et `.jpx` (les deux derniers sont relatifs à des fichiers propres à JBuilder).

Des informations présentant JBuilder 2005 et les fichiers du projet `Welcome` créé par défaut sont finalement affichés : vous pouvez les utiliser pour tester JBuilder ou simplement les ignorer.

Création d'un projet

Pour créer un projet, choisissez l'élément `Nouveau projet...` dans le menu Fichier. L'Expert projet (*Project Wizard* en anglais) qui s'affiche vous propose alors les trois étapes suivantes.

- 1 Choisir le nom de votre projet et le dossier dans lequel il sera enregistré.
- 2 Sélectionner les chemins où seront rangés les fichiers source `.java`, les fichiers `.class`, les fichiers générés par javadoc, ainsi que les bibliothèques nécessaires à votre projet et le répertoire de travail dans lequel sera lancée la JVM pour exécuter votre application.
- 3 Choisir l'encodage des fichiers du projet et les valeurs de certains libellés javadoc.

JAVA 5.0 Utilisation de Java 5.0 dans JBuilder

Pour utiliser les nouvelles fonctionnalités de Java 5.0 dans un projet JBuilder, il faut installer aussi le JDK 1.5 car JBuilder est fourni avec Java 1.4. Sélectionnez ensuite l'élément Propriétés du projet... du menu Projet, et affichez la section Chemins (Paths) dans la boîte de dialogue qui s'affiche. Cliquez sur le bouton ... en regard du JDK puis dans la boîte de dialogue Sélection d'un JDK, cliquez sur le bouton Nouveau... pour spécifier le chemin d'installation du JDK 1.5. Dans la section Construction > Java des propriétés du projet, il faudra finalement choisir l'élément Java 2 SDK v 5.0 dans les listes déroulantes Fonctionnalités du langage et VM cible.

Création des classes

La création de classe s'effectue grâce à des experts qui vous guident en fonction du type de classe que vous désirez (classe simple, application, applet...). Les classes simples se créent grâce à l'élément Nouvelle classe... du menu Fichier qui lance une boîte de dialogue Expert classe dans laquelle vous renseignez l'identificateur de la nouvelle classe, son paquetage, 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 paquetage sont créés automatiquement si nécessaire.

Édition des classes

Outre les outils classiques d'un éditeur que vous retrouverez dans les menus Edition et Chercher, trois fonctionnalités sont particulièrement utiles pendant l'édition de vos classes :

- la vérification syntaxique à la volée qui vous avertit des erreurs que vous avez faites ;
- la complétion automatique qui vous propose les packages à importer, les méthodes disponibles sur un objet ou une classe... La complétion se déclenche soit volontairement grâce au raccourci clavier Ctrl + Espace, soit automatiquement dans certaines situations. Observez bien son comportement car c'est très probablement l'outil qui améliorera le plus votre productivité ;
- les modèles de code Java (*templates*) qui permettent de générer des portions de code en ne tapant que quelques lettres, par exemple pour créer une boucle d'itération à partir du mot `iter`. Cette fonctionnalité s'obtient grâce au raccourci clavier Ctrl + J après avoir saisi l'un des mots de la liste des modèles disponibles. Cette liste est visible dans la section Editeur / Modèles / Java de la boîte de dialogue Préférences lancée par l'élément Préférences... du menu Outils.

En cliquant sur l'onglet Bean d'une classe `public`, vous accéderez à la fonctionnalité de l'outil BeansExpress. L'onglet Propriétés de cet outil permet de

POUR ALLER PLUS LOIN Menu Refactoring

Les éléments du menu Refactoring vous permettent de renommer la classe, la méthode ou la variable en cours de sélection dans l'ensemble des fichiers `.java` d'un projet, mais aussi d'englober des instructions avec `try catch`.

Voir figure page 347.

ATTENTION Dimensions des composants

La taille du conteneur affichée par l'outil de conception n'est pas forcément celle que vous obtiendrez au final. Si vous appliquez par exemple la méthode `pack` au conteneur, sa taille dépendra du layout choisi et des dimensions préférées des composants ajoutés au conteneur.

ATTENTION Style du code généré

Pour que le modificateur d'accès des champs générés soit `private` et que la structure générée pour les listeners d'événements utilise des classes anonymes, il faut modifier les options proposées par l'onglet Généré de la section Formatage des Propriétés du projet.

générer automatiquement les accesseurs `get` et les mutateurs `set` des propriétés de composants JavaBeans en cliquant sur les boîtes à cocher des colonnes `getter` et `setter`.

Compilation et exécution

Il suffit d'utiliser les éléments du menu `Projet` pour compiler les fichiers `.java` d'un projet, et les éléments du menu `Exécuter` (`Run` en anglais) pour lancer une application ou la déboguer.

Conception de l'interface utilisateur

JBuilder 2005 contient un outil de conception qui permet d'éditer interactivement l'interface utilisateur d'une application. Pour y accéder, cliquez sur l'onglet `Conception` d'une sous-classe de conteneur Swing (comme `javax.swing.JPanel` ou `javax.swing.JFrame`) de votre projet. S'affichent alors la liste des composants que vous pouvez ajouter à un conteneur, la structure hiérarchique des composants visualisés et la liste des propriétés disponibles sur le composant en cours de sélection, comme son identificateur, son layout, ses dimensions...

Ajout de composants

Pour construire une interface, sélectionnez le layout de votre conteneur grâce à sa propriété `Layout`, puis ajoutez-y les composants ou les sous-conteneurs de votre choix en cliquant sur leur icône et en cliquant soit dans leur conteneur à l'écran, soit sur l'identificateur de leur conteneur dans la structure hiérarchique des composants. La position d'un composant (`WEST`, `CENTER`...) est gérée grâce à la propriété `Constraints`, si besoin est.

Le code Java qui correspond à l'interface est généré par l'outil de conception dans la classe en cours d'édition sous forme de champs et d'instructions incluses dans la méthode `jbInit` appelée par le constructeur de la classe. JBuilder vous autorise à éditer manuellement cette méthode si nécessaire.

Ajout de listeners

JBuilder facilite aussi la programmation événementielle en créant automatiquement la structure des listeners dont vous avez besoin : cliquez sur le composant sur lequel vous voulez ajouter un traitement événementiel, sélectionnez l'onglet `Evénements` à côté de l'onglet `Propriétés`, saisissez l'identificateur d'une méthode en regard de l'événement qui vous intéresse puis tapez sur `Entrée`. L'outil de conception vous positionne alors directement dans le bloc de la méthode que vous avez saisie et vous n'avez plus qu'à programmer les instructions du traitement !

REGARD DU DÉVELOPPEUR JBuilder 2005 vs Eclipse

JBuilder 2005 est un IDE simple à installer et à appréhender pour les débutants : il inclut le JDK 1.4.2, les versions francisées des erreurs de compilation javac et de nombreux outils faciles d'accès. Certaines fonctionnalités comme la création d'applications Web ou d'EJB sont par contre disponibles uniquement dans les versions payantes.

Eclipse est un IDE Open Source, offrant des fonctionnalités diverses grâce à de nombreux plug-ins inclus en standard. Il est de plus en plus utilisé en entreprise et sert de base à l'IDE WebSphere Studio d'IBM.

JBuilder 2005 a un outil de conception d'interface utilisateur Swing que ne propose pas Eclipse en standard. Ces deux IDE incluent la gestion de CVS pour gérer l'archivage des fichiers d'une équipe.

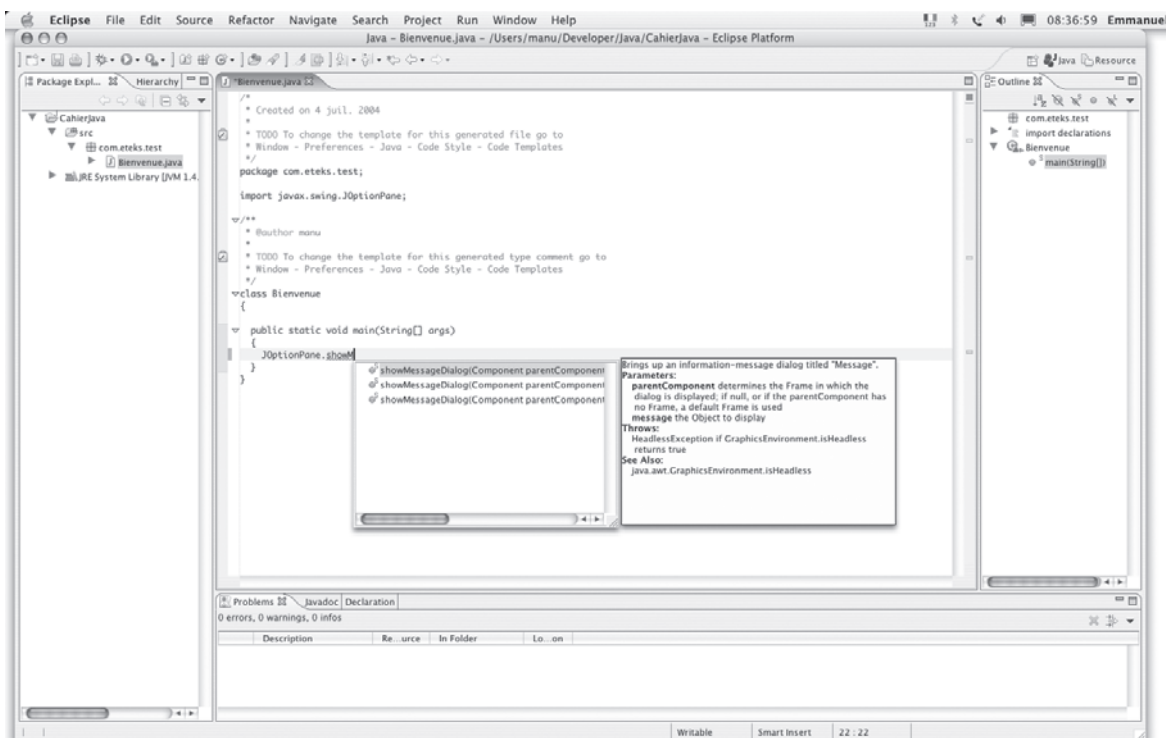
Pour information, JBuilder 2005 et Eclipse sont eux-mêmes des applications Java ; l'interface utilisateur de JBuilder 2005 est réalisée avec Swing, tandis que celle d'Eclipse est réalisée avec SWT, une bibliothèque de composants Java créée pour cet IDE.

▶ <http://www.borland.fr/jbuilder/>

▶ <http://www.eclipse.org/>

Eclipse 3.1

Le dossier ECLIPSE contient les programmes d'installation de l'IDE Eclipse version 3.1.1 pour Windows, Linux et Mac OS X. Eclipse est un IDE Open Source sur lequel est basé l'IDE WebSphere Studio d'IBM.



Pour des raisons de capacité, le CD-Rom contient une distribution spéciale d'Eclipse qui regroupe en un seul fichier compressé les versions d'Eclipse pour Windows, Linux et Mac OS X.

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

Systèmes supportés

- Windows : Windows 98/ME/2000/XP
- Linux : version GTK 2 pour x86
- Mac OS X : Mac OS 10.3

Installation

Installez le JDK comme indiqué dans le chapitre 2, puis décompressez le fichier `eclipse-SDK-3.1.1.zip` avec l'outil de votre choix ou avec la commande :

```
| jar xf eclipse-SDK-3.1.1.zip
```

Sous Linux, rendez la commande `eclipse` du dossier `eclipse` exécutable grâce à la commande :

```
| chmod +x eclipse/eclipse
```

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 (5.0 ou autre) avec lequel il est compatible, ainsi que les sous-dossiers où seront rangés les fichiers source `.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.

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

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

Création des classes

La création de classes s'effectue en sélectionnant l'élément Class... dans le sous-menu New du menu File (si l'élément Class... n'apparaît pas, sélectionnez l'élément Other... puis Class dans la liste qui s'affiche). L'assistant New Java Class qui s'affiche vous permet de renseigner l'identificateur de la nouvelle classe, son paquetage, 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 paquetage sont créés automatiquement si nécessaire. Si Eclipse vous propose de passer en perspective Java, répondez par l'affirmative pour voir apparaître votre nouvelle classe dans la liste des classes de l'écran.

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 permettent de générer 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 par exemple, l'élément Generate getters and setters... du menu Source qui permet de générer automatiquement les accesseurs get et les mutateurs set d'une classe, ou l'élément Rename... du menu Refactoring qui permet de renommer la classe, la méthode ou la variable en cours de sélection dans l'ensemble des fichiers .java d'un projet.

Compilation et exécution

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

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

B.A.-BA 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.

B.A.-BA Warning

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

E. Erreurs de compilation les plus fréquentes

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

Symbole introuvable

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

- ▶ `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.

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

- ▶ `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 ou type inconnu.

- ▶ `ClasseXxxx.java:numLigne: cannot resolve symbol`
symbol : `class flot`
location: class `ClasseXxxx`
Vérifiez la syntaxe du type primitif (ici `flot` à la place de `float`) ou de la classe.

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

- ▶ `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.

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

- ▶ `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.

Le constructeur de la classe `com.eteks.test.ClasseYyyy` sans paramètre 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`.

Déclaration de classe incorrecte

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.

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

Supprimez le nom de la classe à la fin de la déclaration du package.

ClasseXxxx.java:numLigne: **duplicate class**: class *ClasseXxxx*

Vérifiez si vous n'avez pas cité plusieurs fois le fichier *ClasseXxxx.java* dans la commande `javac` ou si vos fichiers sources ne contiennent pas deux fois la déclaration de la classe *ClasseXxxx*.

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

◀ Conflit entre le nom du package et de la classe.

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

Déclaration de méthode incorrecte

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

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

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.

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

◀ 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.

Modificateur d'accès incorrect

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`.

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*.

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

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

◀ 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`, `friendly`, `protected` et `public`.

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

- ▶ `ClasseXxxx.java:numLigne: illegal start of expression`
`private int somme;`
 Supprimez `private` si `somme` est une variable locale.

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

- ▶ `ClasseXxxx.java:numLigne: texte is already defined in main(java.lang.String[])`
 Supprimez le `type` devant `texte` si vous voulez utiliser la variable `texte` ou changez d'identifiant si vous voulez déclarer une autre variable locale.

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.

- ▶ `ClasseXxxx.java:numLigne: not a statement`
`java.lang.String message;`
 Déclarez votre variable avant l'instruction ou incluez-la dans un bloc. Vérifiez que vous n'avez pas oublié des accolades.

Utilisation de variable incorrecte

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

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

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

- ▶ `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`.

Erreur avec return

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

- ▶ `ClasseXxxx.java:numLigne: missing return statement`
 Ajoutez l'instruction `return` suivie du résultat de la méthode.

Instruction impossible à atteindre.

- ▶ `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.

Erreur dans les conditions des instructions if, for ou while

ClasseXxxx.java:numLigne: unexpected type
 required: variable
 found : value

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

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

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

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

◀ Types incompatibles. Cette erreur survient parfois quand on utilise = au lieu de == dans une clause de l'instruction if.

Équilibre incorrect entre accolades ouvrantes et fermantes

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

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

ClasseXxxx.java:numLigne: illegal start of expression
 }

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

ClasseXxxx.java:numLigne: illegal start of type

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

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

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

◀ 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.

Chaîne littérale non fermée

ClasseXxxx.java:numLigne: unclosed string literal

Vérifiez que vous n'avez 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.

◀ Chaîne de caractères littérale non fermée.

Commentaire non fermé

ClasseXxxx.java:numLigne: unclosed comment

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

◀ Commentaire non fermé.

F. 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.
<code>Cast</code>	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> .
<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>friendly</i>	<i>Package access</i>	Modificateur d'accès d'un champ ou d'une méthode limitant leur portée aux classes du même package que leur classe.
<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>	Extend, 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.
<i>JVM</i>	<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.

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).
<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 leur portée à leur 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 leur portée aux sous-classes de leur classe et aux classes du même package que leur classe.
<i>public</i>	Public	Modificateur d'accès d'un champ ou d'une méthode ayant la même portée que leur 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 sont utilisables.
<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 empilées 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..

G. Bibliographie

- [1] The Objective-C Programming Language – Apple
(<http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/>)
Objective-C est un autre langage de programmation objet dérivé du C comme le C++. L'introduction sur la programmation objet de cet ouvrage est très intéressante (vous y reconnaîtrez sûrement la notation qui lui a été empruntée pour symboliser les objets dans l'ouvrage que vous avez dans les mains).
- [2] 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 !
- [3] Cahier du programmeur UML – Modéliser un site e-commerce, Pascal Roques, Eyrolles 2002.
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.
- [4] The Java Tutorial – Mary Campione, Kathy Walrath,... – Sun Microsystems (<http://java.sun.com/docs/books/tutorial/>, existe aussi en version papier aux éditions Addison Wesley)
De très bonnes documentations pour démarrer en Java et pour utiliser sa bibliothèque.
- [5] Thinking in Java – Bruce Eckel – Mind View (<http://www.mindview.net/> & <http://penserenjava.free.fr/>, existe aussi en version papier aux éditions Prentice Hall)
S'appuyant sur de nombreux exemples, les premiers chapitres de cet ouvrage de plus de 1000 pages traitent du noyau du langage Java avec une approche originale très efficace pour les personnes ayant déjà programmé. Son plus gros défaut est son manque d'illustrations (vous ne trouverez pas une seule capture d'écran dans le chapitre consacré à Swing !).
- [6] Java Look and Feel Design Guidelines – Sun Microsystems (<http://java.sun.com/products/jlfd2/guidelines.html>, existe aussi en version papier aux éditions Addison Wesley)
Ce bel ouvrage explique comment concevoir une interface utilisateur avec les composants Swing.
- [7] 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.
- [8] Java efficace – Joshua Bosch – 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.