



Ressourcesinformatiques

JAVA 8

Les fondamentaux du langage Java

Avec exercices pratiques et corrigés

Thierry GROUSSARD

Téléchargement
www.editions-eni.fr



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RI8JAV** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1 Présentation

1. Historique	11
1.1 Pourquoi Java ?	11
1.2 Objectifs de la conception de Java	12
1.3 Essor de Java	13
2. Caractéristiques de Java	14
2.1 Le langage de programmation Java	15
2.1.1 Simple	15
2.1.2 Orienté objet	16
2.1.3 Distribué	17
2.1.4 Interprété	17
2.1.5 Robuste	17
2.1.6 Sécurisé	18
2.1.7 Indépendant des architectures	19
2.1.8 Portable	19
2.1.9 Performant	19
2.1.10 Multitâche	20
2.1.11 Dynamique	20
2.2 La plate-forme Java	20
2.2.1 La machine virtuelle Java (JVM)	21
2.2.2 L'API Java	23
2.2.3 Les outils de déploiement des applications	26
2.2.4 Les outils d'aide au développement	26
2.3 Cycle de conception d'un programme Java	27

3.	Installation du SDK version Win32 pour l'environnement Windows	28
3.1	Téléchargement	28
3.2	Installation	29
3.3	Configuration	30
3.4	Test de la configuration du SDK	31
3.5	Installation de la documentation du SDK et des API standard	32
4.	Les différentes étapes de création d'un programme Java	34
4.1	Création des fichiers source	34
4.2	Compiler un fichier source	35
4.3	Exécuter une application	37
5.	Votre première application Java	38
5.1	Squelette d'une application	38
5.2	Arguments en ligne de commande	40
5.2.1	Principes et utilisation	40
5.2.2	Passage d'arguments à une application Java au moment de l'exécution	41

Chapitre 2

Bases du langage

1.	Les variables, constantes et énumérations	43
1.1	Les variables	43
1.1.1	Nom des variables	44
1.1.2	Type des variables	44
1.1.3	Valeurs par défaut	48
1.1.4	Valeurs littérales	49
1.1.5	Conversions de types	50
1.1.6	Déclaration des variables	53
1.1.7	Portée des variables	54
1.1.8	Niveau d'accès des variables	55
1.1.9	Durée de vie des variables	55

1.2	Les constantes	56
1.3	Les énumérations	57
1.4	Les tableaux.	60
1.5	Les chaînes de caractères	65
1.6	Date et heure.	72
2.	Les opérateurs	75
2.1	Les opérateurs unaires	76
2.2	Les opérateurs d'affectation	77
2.3	Les opérateurs arithmétiques.	77
2.4	Les opérateurs bit à bit	78
2.5	Les opérateurs de comparaison	78
2.6	L'opérateur de concaténation.	79
2.7	Les opérateurs logiques.	80
2.8	Ordre d'évaluation des opérateurs.	81
3.	Les structures de contrôle	82
3.1	Structures de décision	82
3.1.1	Structure if	82
3.1.2	Structure switch	84
3.2	Les structures de boucle	86
3.2.1	Structure while	86
3.2.2	Structure do ... while	87
3.2.3	Structure for	87
3.2.4	Interruption d'une structure de boucle	90
4.	Exercices	93
5.	Corrections	94

Chapitre 3

Programmation objet

1. Introduction	101
2. Mise en œuvre avec Java	104
2.1 Création d'une classe	105
2.1.1 Déclaration de la classe	105
2.1.2 Création des champs	106
2.1.3 Création de méthodes	107
2.1.4 Les accesseurs	112
2.1.5 Constructeurs et destructeurs	113
2.1.6 Champs et méthodes statiques	115
2.1.7 Les annotations.	117
2.2 Utilisation d'une classe.	120
2.2.1 Création d'une instance	120
2.2.2 Initialisation d'une instance.	121
2.2.3 Destruction d'une instance	123
2.3 Héritage.	128
2.3.1 this et super	130
2.3.2 Classes abstraites	135
2.3.3 Classes finales.	136
2.3.4 Conversion de type.	137
2.3.5 La classe Object	143
2.4 Interfaces.	150
2.4.1 Création d'une interface.	150
2.4.2 Utilisation d'une interface	151
2.4.3 Méthodes par défaut	155
2.5 Classes imbriquées	161
2.5.1 Classes imbriquées statiques	161
2.5.2 Classes internes	162
2.5.3 Classes anonymes.	164
2.6 Expression lambda	169

2.7	Référence de méthode	176
2.8	Les génériques	178
2.8.1	Classes génériques	179
2.8.2	Méthodes génériques	187
2.8.3	Les génériques et l'héritage	188
2.8.4	Limitation des génériques	194
2.9	Les packages	197
2.9.1	Création d'un package	197
2.9.2	Utilisation et importation d'un package	199
3.	Gestion des exceptions	201
3.1	Les erreurs de syntaxe	201
3.2	Les erreurs d'exécution	203
3.3	Les erreurs de logique	203
3.3.1	Les exceptions	204
3.3.2	Récupération d'exceptions	205
3.3.3	Exceptions associées à des ressources	209
3.3.4	Création et déclenchement d'exceptions	213
4.	Les collections	215
4.1	La classe ArrayList	216
4.2	La classe HashSet	220
4.3	La classe LinkedList	229
4.4	streams et pipelines	230
5.	Exercices	233
6.	Corrections	234

Chapitre 4

Applications graphiques

1. Introduction	257
1.1 Les bibliothèques graphiques	258
1.1.1 La bibliothèque AWT	258
1.1.2 La bibliothèque Swing	258
1.2 Constitution de l'interface graphique d'une application	259
2. Conception d'une interface graphique	260
2.1 Les fenêtres	260
2.2 La gestion des événements	265
2.3 Aspect des composants	295
2.4 Le positionnement des composants	296
2.4.1 FlowLayout	297
2.4.2 BorderLayout	299
2.4.3 GridLayout	305
2.4.4 BoxLayout	307
2.4.5 GridBagLayout	311
2.4.6 Sans gestionnaire de mise en page	315
2.5 Les composants graphiques	318
2.5.1 La classe JComponent	319
2.5.2 Affichage d'informations	322
2.5.3 Les composants d'édition de texte	327
2.5.4 Les composants de déclenchement d'actions	333
2.5.5 Les composants de sélection	340
2.6 Les boîtes de dialogue	348
2.6.1 La boîte de saisie	349
2.6.2 La boîte de message	352
2.6.3 La boîte de confirmation	352

Chapitre 5
Les applets

- 1. Principe de fonctionnement 355
- 2. Création d'une applet 356
 - 2.1 Cycle de vie d'une applet 357
 - 2.1.1 Méthodes liées au cycle de vie de l'applet 357
 - 2.1.2 Méthodes de gestion de l'aspect graphique de l'applet . 358
 - 2.2 Construire l'interface utilisateur d'une applet 361
 - 2.2.1 Création d'une police de caractères 362
 - 2.2.2 Obtenir les dimensions de l'applet. 362
 - 2.2.3 Dessiner les caractères 363
 - 2.2.4 Déterminer les dimensions d'une chaîne. 363
 - 2.3 Les images dans les applets 368
 - 2.3.1 Chargement d'une image 369
 - 2.3.2 Traitement de l'image 371
 - 2.3.3 Traçage de l'image. 371
 - 2.4 Les threads dans les applets 374
 - 2.4.1 Création d'un nouveau thread 376
 - 2.4.2 Définir le traitement à effectuer 376
 - 2.4.3 Démarrer et arrêter un thread 379
 - 2.5 Les sons dans les applets 381
- 3. Déployer une applet 383
 - 3.1 La balise <applet> 383
 - 3.2 Paramétrage d'une applet 385
 - 3.2.1 Définir les paramètres 385
 - 3.2.2 Récupération des paramètres dans l'applet. 386
 - 3.3 Sécurité dans une applet 387
 - 3.4 Communication entre applets. 388

3.5	Interaction avec le navigateur et le système	390
3.5.1	Affichage sur la console	390
3.5.2	Utilisation de la barre d'état du navigateur	392
3.5.3	Affichage d'une page html	393
3.5.4	Obtenir certaines propriétés du système	394

Chapitre 6

Accès aux bases de données

1.	Principe de fonctionnement d'une base de données	397
1.1	Terminologie	397
1.2	Le langage SQL	398
1.2.1	Recherche d'informations	399
1.2.2	Ajout d'informations	401
1.2.3	Mise à jour d'informations	401
1.2.4	Suppression d'informations	402
2.	Accès à une base de données à partir de Java	402
2.1	Présentation de jdbc	404
2.2	Chargement du pilote	405
2.3	Établir et manipuler la connexion	406
2.3.1	Établir la connexion	406
2.3.2	Manipuler la connexion	407
2.4	Exécution d'instructions SQL	412
2.4.1	Exécution d'instructions de base avec l'objet Statement	412
2.4.2	Exécution d'instructions paramétrées avec l'objet PreparedStatement	420
2.4.3	Exécution de procédures stockées avec l'objet CallableStatement	424
2.5	Utilisation des jeux d'enregistrements avec l'interface ResultSet	427
2.5.1	Positionnement dans un ResultSet	429
2.5.2	Lecture des données dans un ResultSet	431

- 2.5.3 Modification des données dans un ResultSet 434
- 2.5.4 Suppression de données dans un ResultSet 437
- 2.5.5 Ajout de données dans un ResultSet 438
- 2.6 Gestion des transactions 439
 - 2.6.1 Mise en œuvre des transactions 441
 - 2.6.2 Points de sauvegarde 442
 - 2.6.3 Niveaux d'isolement 442

Chapitre 7
Déploiement d'applications

- 1. Archives Java 445
 - 1.1 Présentation 445
 - 1.2 Manipulation d'une archive 446
 - 1.2.1 Création d'une archive 446
 - 1.2.2 Visualisation du contenu 447
 - 1.2.3 Extraction 448
 - 1.2.4 Mise à jour 448
 - 1.2.5 Exécution 449
 - 1.3 Le manifest 449
 - 1.3.1 Présentation 450
 - 1.3.2 Création 450
 - 1.4 Scellement et signature d'une archive 451
 - 1.4.1 Scellement 451
 - 1.4.2 Signature 455
- 2. Java Web Start 460
 - 2.1 Présentation 460
 - 2.2 Exécution d'une application 461
 - 2.2.1 À partir d'un navigateur 461
 - 2.2.2 À partir du cache local 461

2.3	Déploiement d'une application	464
2.3.1	Configuration du serveur Web	464
2.3.2	Création du fichier JNLP	467
2.3.3	Déployer l'application sur le serveur	471
2.3.4	Création de la page Web d'accueil	473
	Index	475

Chapitre 4

Applications graphiques

1. Introduction

Jusqu'à présent, tous les exemples de code que nous avons réalisés fonctionnent exclusivement en mode caractères. Les informations sont affichées dans une console et également saisies à partir de celle-ci. La simplicité de ce mode de fonctionnement est un atout indéniable pour l'apprentissage d'un langage. Par contre, la plupart des utilisateurs de vos futures applications s'attendent certainement à avoir une interface un petit peu moins austère qu'un écran en mode caractères. Nous allons donc étudier dans ce chapitre comment fonctionnent les interfaces graphiques avec Java. Vous allez rapidement vous apercevoir que la conception d'interfaces graphiques en Java n'est pas très simple et nécessite l'écriture de nombreuses lignes de code. Dans la pratique, de nombreux outils de développement sont capables de prendre en charge la génération d'une grande partie de ce code en fonction de la conception graphique de l'application que vous dessinez. Il est cependant important de bien comprendre les principes de fonctionnement de ce code pour éventuellement intervenir dessus et l'optimiser. Nous n'utiliserons pas dans ce chapitre d'outil spécifique mais nous conserverons notre trio éditeur de texte, compilateur, machine virtuelle.

1.1 Les bibliothèques graphiques

Le langage Java propose deux bibliothèques dédiées à la conception d'interfaces graphiques. La bibliothèque AWT et la bibliothèque SWING. Les principes d'utilisation sont quasiment identiques pour ces deux bibliothèques. L'utilisation simultanée de ces deux bibliothèques dans une même application peut provoquer des problèmes de fonctionnement et doit être évitée.

1.1.1 La bibliothèque AWT

Cette bibliothèque est la toute première disponible pour le développement d'interfaces graphiques. Elle contient une multitude de classes et interfaces permettant la définition et la gestion d'interfaces graphiques. Cette bibliothèque utilise en fait les fonctionnalités graphiques du système d'exploitation. Ce n'est donc pas le code présent dans cette bibliothèque qui assure le rendu graphique des différents composants. Ce code sert uniquement d'intermédiaire avec le système d'exploitation. Son utilisation est de ce fait relativement économe en ressources. Par contre elle souffre de plusieurs inconvénients.

- L'aspect visuel de chaque composant étant lié à la représentation qu'en fait le système d'exploitation, il est parfois délicat de développer une application ayant une apparence cohérente sur tous les systèmes. La taille et la position des différents composants étant les deux éléments principalement affectés par ce problème.
- Pour que cette bibliothèque soit compatible avec tous les systèmes d'exploitation, les composants qu'elle contient sont donc limités aux plus courants (boutons, zone de texte, listes...).

1.1.2 La bibliothèque Swing

Cette bibliothèque a été conçue pour pallier les principales insuffisances de la bibliothèque AWT. Cette amélioration a été obtenue en écrivant entièrement cette bibliothèque en Java sans pratiquement faire appel aux services du système d'exploitation. Seuls quelques éléments graphiques (fenêtres et boîtes de dialogue) sont encore en relation avec le système d'exploitation. Pour les autres composants, c'est le code de la bibliothèque Swing qui détermine entièrement leurs aspects et comportements.

La bibliothèque Swing contient donc une quantité impressionnante de classes servant à redéfinir les composants graphiques. Il ne faut cependant pas penser que la bibliothèque Swing rend complètement obsolète la bibliothèque AWT. Beaucoup d'éléments de la bibliothèque AWT sont d'ailleurs repris dans la bibliothèque Swing. Nous utiliserons principalement cette bibliothèque dans le reste de ce chapitre.

1.2 Constitution de l'interface graphique d'une application

La conception de l'interface graphique d'une application consiste essentiellement à créer des instances des classes représentant les différents éléments nécessaires, modifier les caractéristiques de ces instances de classe, les assembler et prévoir le code de gestion des différents événements pouvant intervenir au cours du fonctionnement de l'application. Une application graphique est donc constituée d'une multitude d'éléments superposés ou imbriqués. Parmi ces éléments, l'un d'entre eux joue un rôle prépondérant dans l'application. Il est souvent appelé conteneur de premier niveau. C'est lui qui va interagir avec le système d'exploitation et contenir tous les autres éléments. En général ce conteneur de premier niveau ne contient pas directement les composants graphiques mais d'autres conteneurs sur lesquels sont placés les composants graphiques. Pour faciliter la disposition de ces éléments les uns par rapport aux autres, nous pouvons utiliser l'aide d'un gestionnaire de mise en page. Cette superposition d'éléments peut être assimilée à une arborescence au sommet de laquelle nous avons le conteneur de premier niveau et dont les différentes branches sont constituées d'autres conteneurs. Les feuilles de l'arborescence correspondant aux composants graphiques.

Le conteneur de premier niveau étant l'élément indispensable de toute application graphique, nous allons donc commencer par étudier en détail ces caractéristiques et son utilisation puis nous étudierons les principaux composants graphiques.

2. Conception d'une interface graphique

Nous avons vu un petit peu plus haut que toute application graphique est au moins constituée d'un conteneur de premier niveau. La bibliothèque Swing dispose de trois classes permettant de jouer ce rôle :

`JApplet` : représente une fenêtre graphique embarquée à l'intérieur d'une page html pour être prise en charge par un navigateur. Cet élément est étudié en détail dans le chapitre qui lui est consacré.

`JWindow` : représente une fenêtre graphique la plus rudimentaire qui soit. Celle-ci ne dispose pas de barre de titre, de menu système, pas de bordure, c'est en fait un simple rectangle sur l'écran. Cette classe est rarement utilisée sauf pour l'affichage d'un écran d'accueil lors du démarrage d'une application (*splash screen*).

`JFrame` : représente une fenêtre graphique complète et pleinement fonctionnelle. Elle dispose d'une barre de titre, d'un menu système, d'une bordure, elle peut facilement accueillir un menu, c'est bien sûr cet élément que nous allons utiliser dans la très grande majorité des cas.

2.1 Les fenêtres

La classe `JFrame` est l'élément indispensable de toute application graphique. Comme pour une classe normale nous devons créer une instance, modifier éventuellement les propriétés et utiliser les méthodes. Voici donc le code de la première application graphique.

```
package fr.eni;
import javax.swing.JFrame;
public class Principale {
    public static void main(String[] args)
    {
        JFrame fenetre;
        // création de l'instance de la classe JFrame
        fenetre=new JFrame();
        // modification de la position et de la
        // taille de la fenêtre
        fenetre.setBounds(0,0,300,400);
        // modification du titre de la fenêtre
```

```
fenetre.setTitle("première fenêtre en JAVA");  
// affichage de la fenêtre  
fenetre.setVisible(true);  
}
```

Et le résultat de son exécution :



C'est simple d'utilisation et très efficace. C'est d'ailleurs tellement efficace que vous ne pouvez pas arrêter l'application. En effet même si la fenêtre est fermée par l'utilisateur, cette fermeture ne provoque pas la suppression de l'instance de la `JFrame` de la mémoire. La seule solution pour arrêter l'application est de stopper la machine virtuelle Java avec la combinaison de touches [Ctrl] **C**. Il faut bien sûr prévoir une autre solution pour terminer plus facilement l'exécution de l'application et faire en sorte que celle-ci s'arrête à la fermeture de la fenêtre.

La première solution consiste à gérer les événements se produisant lors de la fermeture de la fenêtre et dans un de ces événements provoquer l'arrêt de l'application. Cette solution sera étudiée dans le paragraphe consacré à la gestion des événements.

La deuxième solution utilise des comportements prédéfinis pour la fermeture de la fenêtre. Ces comportements sont déterminés par la méthode `setDefaultCloseOperation`. Plusieurs constantes sont définies pour déterminer l'action entreprise à la fermeture de la fenêtre.

`DISPOSE_ON_CLOSE` : cette option provoque l'arrêt de l'application lors de la fermeture de la dernière fenêtre prise en charge par la machine virtuelle.

`DO_NOTHING_ON_CLOSE` : avec cette option, il ne se passe rien lorsque l'utilisateur demande la fermeture de la fenêtre. Il est dans ce cas obligatoire de gérer les événements pour que l'action de l'utilisateur provoque un effet sur la fenêtre ou l'application.

`EXIT_ON_CLOSE` : cette option provoque l'arrêt de l'application même si d'autres fenêtres sont encore visibles.

`HIDE_ON_CLOSE` : avec cette option, la fenêtre est simplement masquée par un appel à sa méthode `setVisible(false)`.

La classe `JFrame` se trouve située à la fin d'une hiérarchie de classes assez importante et implémente de nombreuses interfaces. De ce fait elle possède donc de nombreuses méthodes et attributs.

Class JFrame

```
java.lang.Object
├── java.awt.Component
│   ├── java.awt.Container
│   │   ├── java.awt.Window
│   │   │   ├── java.awt.Frame
│   │   │   └── javax.swing.JFrame
```

All Implemented Interfaces:

[ImageObserver](#), [MenuContainer](#), [Serializable](#), [Accessible](#), [RootPaneContainer](#), [WindowConstants](#)

Le but de cet ouvrage n'est pas de reprendre entièrement la documentation du JDK, aussi il ne présente pas toutes les méthodes disponibles mais simplement les plus couramment utilisées au fur et à mesure des besoins. Il peut être cependant intéressant de parcourir la documentation avant de se lancer dans la conception d'une méthode pour déterminer si ce que l'on souhaite réaliser n'a pas déjà été prévu par les concepteurs de Java.

Maintenant que nous sommes capables d'afficher une fenêtre, le plus gros de notre travail va consister à ajouter un contenu à la fenêtre. Avant de pouvoir ajouter quelque chose sur une fenêtre, il faut bien comprendre sa structure qui est relativement complexe. Un objet `JFrame` est composé de plusieurs éléments superposés jouant chacun un rôle bien spécifique dans la gestion de la fenêtre.