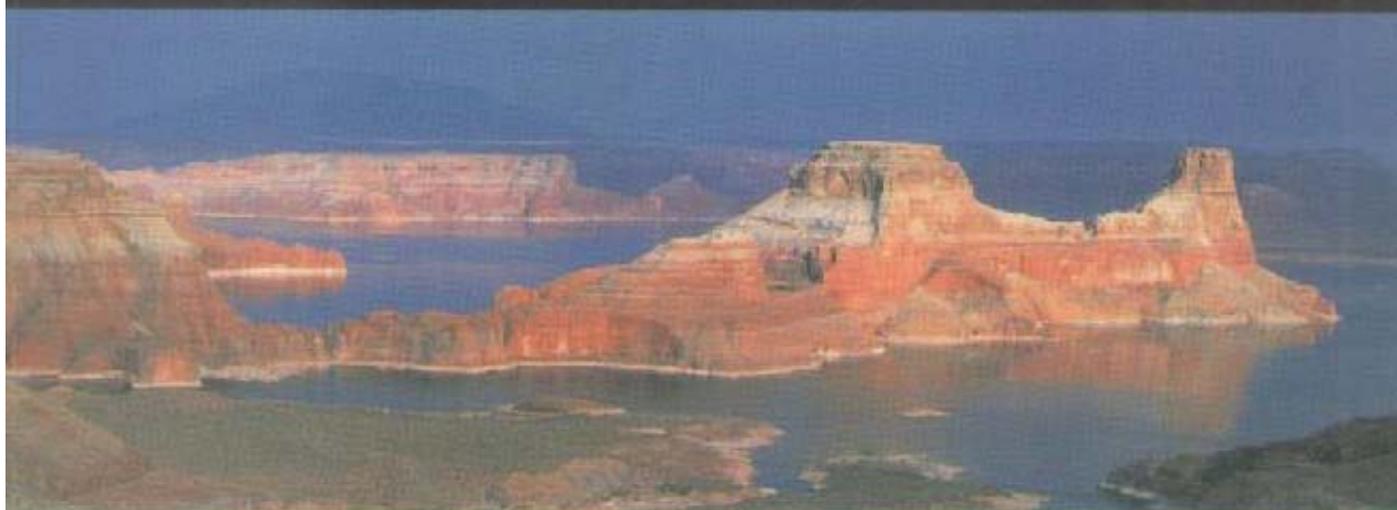


JAVA

LA SYNTHÈSE

Concepts, architectures, frameworks



01
INFORMATIQUE

Gilles Clavel, Valérie Lehman, Nicolas Mirouze,
Emmanuelle Mouthe, Sandrine Munerot,
Emmanuel Pichon, Mohamed Soukal et Simon Tiffanneau

Préface d'Éric Mahé

4^e édition

DUNOD

2-005-624-1

JAVA

LA SYNTHÈSE

**Concepts,
architectures, frameworks**

Gilles Clavel, Valérie Lehman, Nicolas Mirouze,
Emmanuelle Mouthe, Sandrine Munerot,
Emmanuel Pichon, Mohamed Soukal et Simon Tiffanneau

Consultants Java/Technologies Web

Préface d'Éric Mahé

4^e édition



DUNOD

Table des matières

Avant-propos	XIV
Chapitre 1 – Java et les objets	1
1.1 Un langage à part entière	1
1.2 Programmation traditionnelle et programmation objet	2
1.2.1 Programmation classique : données et traitements	2
1.2.2 Programmation objet : l'encapsulation	3
1.3 Le concept de classe, support de l'encapsulation	4
1.3.1 Première version de la classe <code>Personne</code>	4
1.3.2 Définition des variables d'instance <code>nom</code> et <code>societe</code>	6
1.3.3 Définition de la méthode <code>presenteToi</code>	6
1.4 Instancions et utilisons un objet	7
1.5 Implémenter une classe	9
1.5.1 Spécifions les propriétés de la classe <code>Personne</code>	9
1.5.2 Variables d'instance privées	10
1.5.3 Définissons un constructeur	12
1.5.4 Accès à la variable d'instance privée <code>societe</code>	13
1.5.5 Un ou plusieurs constructeurs ?	16
1.5.6 Variables et méthodes de classe	17
1.5.7 La version finale de la classe <code>Personne</code>	20
1.6 Concevoir et utiliser une classe	21
1.6.1 Les deux acteurs de la programmation objet	22
1.6.2 Définir une classe en Java	22
1.7 Qu'est-ce qu'un programme Java ?	24
1.7.1 Une application Java	24
1.7.2 Code compilé et machine virtuelle	25
1.7.3 Applications et applets	29
1.8 Principaux éléments de la syntaxe du langage	30
1.8.1 Types primitifs et définition de variables	30
1.8.2 Les variables	31
1.8.3 Instructions simples et instructions composées	32
1.8.4 Énoncés conditionnels	32
1.8.5 Itérations	33

Chapitre 2 – Des types primitifs à l'héritage	35
2.1 Types primitifs et types objet	35
2.1.1 Les types primitifs numériques	36
2.1.2 Le type primitif char	37
2.1.3 Le type primitif boolean	39
2.2 Représentation et manipulation des variables	40
2.2.1 Valeur d'une variable	40
2.2.2 Variables locales et paramètres	42
2.2.3 Transmission de paramètres	44
2.2.4 Signature d'une méthode	46
2.3 Les opérateurs	47
2.3.1 L'opérateur new	47
2.3.2 L'affectation	48
2.3.3 Les opérations arithmétiques	49
2.3.4 Opérateurs à résultat de type boolean	51
2.3.5 L'opérateur ternaire	51
2.3.6 Opérateurs, types primitifs et types objet	52
2.4 Classes String et StringBuffer	52
2.4.1 La classe String	53
2.4.2 La classe StringBuffer	56
2.4.3 Égalité de chaînes de caractères	57
2.5 À quoi sert l'héritage ?	58
2.5.1 Bien comprendre le mécanisme de la construction par défaut	58
2.5.2 Une classe Ville	60
2.5.3 Une capitale est une ville particulière	62
2.5.4 Un constructeur fonctionnel pour la classe Capitale	63
2.5.5 Accès aux propriétés héritées	65
2.5.6 De qui hérite la classe Ville ?	67
2.6 Héritage, transtypage et polymorphisme	67
2.6.1 L'héritage définit un cast implicite de classe fille vers classe mère	67
2.6.2 Redéfinissons une méthode héritée	69
2.6.3 Les transtypes induits par l'héritage facilitent le polymorphisme	69
2.6.4 Comment rendre polymorphe la méthode decrisToi ?	71
2.6.5 Hériter, redéfinir : récapitulons	72
2.6.6 Encore un exemple de l'utilité du cast induit par l'héritage	73
2.7 Encapsulation des types primitifs	75
2.8 Packages et modificateurs d'accès	76
2.8.1 Organisation des bibliothèques de classes Java	76
2.8.2 Packages et modificateurs d'accès	77
2.8.3 Déclaration de packages	78
2.8.4 Importation de classes et espaces de nommage	79
Chapitre 3 – Les outils avancés du langage	80
3.1 Une classe abstraite n'est pas instanciable	80
3.1.1 Gérer des règlements	80

3.1.2 La classe <code>Reglement</code> est abstraite	82
3.1.3 Classes abstraites et méthodes abstraites	83
3.2 Qu'est-ce qu'une interface ?	84
3.2.1 Héritage simple ou héritage multiple ?	84
3.2.2 Recevoir en paramètre des objets de types différents	84
3.2.3 Une interface pour éviter l'héritage multiple	86
3.2.4 L'héritage d'une interface définit aussi un cast	88
3.2.5 Définir et utiliser des interfaces	91
3.3 Gestion des exceptions	91
3.3.1 Comment gérer les erreurs d'exécution	91
3.3.2 La méthode <code>setNbHabitants</code> lève une exception	92
3.3.3 Propager ou capturer une exception	94
3.3.4 Récapitulons le mécanisme	96
3.3.5 La hiérarchie des classes d'exception	96
3.3.6 Exceptions contrôlées et non contrôlées	98
3.4 Des collections pour gérer des clients	99
3.4.1 Créer une classe <code>Client</code>	99
3.4.2 Construire un tableau	101
3.4.3 Modifier un tableau	103
3.4.4 Un tableau est un objet	104
3.4.5 Exploiter un tableau de chaînes de caractères	104
3.5 Une implémentation dynamique de <code>Client</code>	106
3.5.1 Manipuler la classe <code>Vector</code>	106
3.5.2 Parcourir un vecteur	108
3.5.3 Quelques remarques sur l'implémentation de la classe <code>Client</code>	109
3.6 Classer les clients par société avec un dictionnaire	110
3.6.1 Manipuler un dictionnaire	110
3.6.2 Créer une classe <code>GereClient</code> qui utilise un dictionnaire	111
3.7 Les flux	113
3.7.1 Qu'est-ce qu'un flux ?	113
3.7.2 Les flux standard	114
3.7.3 Encapsulation de fichiers	114
3.7.4 Manipuler un flux en lecture sur un fichier	115
3.7.5 Manipuler un flux en écriture sur un fichier	118
3.7.6 Une vue plus étendue du package <code>java.io</code>	121
3.8 Flux et sérialisation des objets	121
3.8.1 Sérialiser et réinstancier un objet de base	122
3.8.2 Comment rendre un objet sérialisable ?	124
3.8.3 Bien utiliser <code>writeObject</code> et <code>readObject</code>	125
3.8.4 Adapter la sérialisation à des conditions particulières	126
3.9 Quelques particularités des mécanismes objet	127
3.9.1 Le modificateur <code>final</code>	127
3.9.2 La classe <code>Object</code>	128
3.9.3 Attention aux copies !	129
3.9.4 Les problèmes de l'héritage multiple	133
3.9.5 Implémenter une interface par délégation	134

Chapitre 4 – Construire une interface graphique	139
4.1 Une classe support des applications autonomes	140
4.1.1 Une première fenêtre	140
4.1.2 Cahier des charges de notre éditeur graphique	140
4.2 Créer un menu	141
4.2.1 Le menu de l'éditeur graphique	141
4.2.2 Comment insérer un menu dans une fenêtre ?	143
4.3 Structure d'une interface graphique en Java	143
4.3.1 Conteneurs et composants	143
4.3.2 Présentation des contrôles graphiques de Java	144
4.3.3 Un premier composant	145
4.3.4 La classe <code>Panel</code> : composant et conteneur	146
4.3.5 Construction de la barre d'outils	146
4.4 Disposer des composants dans un conteneur	148
4.4.1 Que sont les <code>LayoutManager</code> ?	148
4.4.2 Utilisation de <code>LayoutManager</code> prédéfinis	149
4.4.3 Création de la barre d'état	149
4.4.4 Comment dimensionner les composants ?	152
4.5 Fabriquer une applet	154
4.5.1 Qu'est-ce qu'une applet ?	154
4.5.2 Une applet qui implémente un éditeur graphique	154
4.6 Gérer les événements	156
4.6.1 Comment va fonctionner l'application ?	156
4.6.2 Affichage des coordonnées de la souris	158
4.6.3 Préparer le dessin des segments de droite	159
4.6.4 Récapitulons...	160
4.6.5 Fermer enfin la fenêtre	162
4.6.6 Gérer le menu et la barre d'outils	163
4.6.7 Conclusion	168
4.7 Dessiner avec Java	168
4.7.1 Notion de contexte graphique	168
4.7.2 Dessinons des segments de droites	168
4.7.3 Comment l'affichage est-il géré ?	169
4.7.4 Redessiner les droites disparues	171
4.8 Revenons à notre applet	173
4.8.1 Quitter une applet	173
4.8.2 Mieux gérer la barre d'état	174
4.8.3 Une interface pour décrire une barre d'état	175
4.9 Construire la boîte de dialogue A Propos	177
4.9.1 Le conteneur <code>Dialog</code>	177
4.9.2 Ajouter une image et un bouton	178
4.10 L'évolution du package <code>awt</code> vers les <code>JFC</code>	181
Chapitre 5 – Applets et threads	182
5.1 Deux types d'applications	182

5.1.1 Un exemple d'application autonome	182
5.1.2 Un exemple d'applet	184
5.1.3 Une conception moins rigide	186
5.2 États et transitions, le cycle de vie d'une applet	189
5.3 Environnement HTML d'une applet	191
5.4 Les threads et la synchronisation d'instructions	195
5.4.1 Qu'est-ce qu'un thread ?	195
5.4.2 La classe <code>CompteARebours</code>	197
5.4.3 Deux threads qui partagent une même ressource	198
5.4.4 Synchronisation : principe	200
5.4.5 Synchronisation d'un bloc d'instructions	203
5.4.6 Synchronisation de méthodes d'instance	203
5.5 Utiliser la classe <code>Thread</code>	205
5.5.1 La classe <code>Equipe</code> hérite de <code>Thread</code>	205
5.5.2 La classe <code>Simulation</code> support de l'application	207
5.5.3 Un objet <code>Equipe</code> existe dans plusieurs états	208
5.6 Utiliser l'interface <code>Runnable</code>	208
5.6.1 Première version de la classe <code>HorlogeRunnable</code>	210
5.6.2 Complétons la définition de la classe <code>HorlogeRunnable</code>	212
5.6.3 La version complète de la classe <code>HorlogeRunnable</code>	213
5.6.4 <code>wait</code> , <code>notify</code> et la synchronisation	215
5.7 Manipuler des ressources Web dans une applet	216
5.7.1 Une URL pour référencer une ressource	216
5.7.2 Les droits des applets	218
5.7.3 Les ressources Web et les services de la classe <code>Applet</code>	220
5.8 L'animation	224
5.8.1 Une classe abstraite dérivée d' <code>Applet</code>	225
5.8.2 Une première classe d'animation	226
5.8.3 Un <code>Mediatracker</code> pour synchroniser le chargement des images	229
5.8.4 La technique du double-buffering pour optimiser l'affichage	229
Chapitre 6 – Développer des composants réutilisables avec les JavaBeans	233
6.1 Qu'est-ce qu'un <code>JavaBean</code> ?	233
6.1.1 Un Bean est un composant logiciel	233
6.1.2 Une première application à base de <code>JavaBeans</code>	234
6.1.3 L'interface des <code>JavaBeans</code>	235
6.2 Construire un Bean	239
6.2.1 Un Bean : un ensemble de classes et de ressources	239
6.2.2 Premier pas avec un Bean : la classe d'implémentation <code>SpinEdit</code>	241
6.2.3 Définir une propriété grâce à des accesseurs	244
6.2.4 Communication entre deux Beans grâce à une propriété liée	245
6.2.5 Créer ses propres événements	248
6.2.6 Gérer l'abonnement et propager un événement	252
6.3 Gérer dynamiquement des objets	255
6.3.1 L'introspection	255

6.3.2	Déterminer dynamiquement la classe d'un objet	255
6.3.3	Décrire des objets en utilisant la classe <code>Class</code>	257
6.3.4	Consulter les méthodes d'un objet	259
6.3.5	Exécuter dynamiquement une méthode	260
6.3.6	Manipuler une fenêtre dynamiquement	262
6.3.7	Consulter les variables d'instance et de classe d'un objet	263
6.3.8	Accès aux modificateurs	264
6.4	Packager un Bean	264
6.4.1	Définir l'interface d'un <code>JavaBean</code>	264
6.4.2	L'interface <code>BeanInfo</code>	265
6.4.3	Les fichiers <code>Jar</code>	269
Chapitre 7 – Découvrir l'architecture J2EE		272
7.1	Présentation des nouvelles applications Web	272
7.1.1	L'évolution du Web	272
7.1.2	Les limites des applications HTML classiques	273
7.1.3	Applications à code déployé	275
7.1.4	Une ouverture nécessaire	275
7.2	Le modèle d'applications J2EE	276
7.2.1	Les objectifs	276
7.2.2	Modèle applicatif/architecture technique	276
7.2.3	Vers la réutilisation... ..	280
7.2.4	...et la portabilité	281
7.3	Les composantes de l'architecture J2EE	281
7.3.1	Le conteneur Web	282
7.3.2	Le conteneur d'EJB (<i>Enterprise JavaBeans</i>)	294
7.3.3	Les « Web services » (<i>services Web</i>)	300
7.3.4	L'assemblage et le déploiement d'applications J2EE	304
7.4	Le modèle MVC (<i>Model-View-Controller</i>)	305
7.4.1	Les servlets	305
7.4.2	Les JSP, modèle 1	306
7.4.3	Les JSP, modèle 2 (MVC)	306
7.4.4	MVC 2	308
7.4.5	J2EE et MVC	308
7.5	Les serveurs d'applications	309
7.5.1	Des moteurs de logiciels	309
7.5.2	Les enjeux	310
7.5.3	Les outils complémentaires	313
Chapitre 8 – Java pour toutes les plates-formes		314
8.1	La librairie JDBC	314
8.1.1	JDBC et les architectures client/serveur	314
8.1.2	Typologie des drivers JDBC	316
8.1.3	Les objets de la librairie JDBC	318
8.1.4	Les services avancés de la librairie JDBC	322

8.1.5 Les nouveautés du JDBC 3.0	323
8.2 Java et la mobilité	325
8.2.1 Les technologies du monde mobile	325
8.2.2 Deux catégories d'appareils, deux architectures J2ME	326
8.2.3 Les configurations	329
8.2.4 Les profils	332
8.2.5 Les packages optionnels	335
8.2.6 Les applications s'appuyant sur l'architecture J2ME	335
8.2.7 Les outils	339
8.3 Java, plate-forme universelle ?	340
8.3.1 L'évolution du JDK	341
8.3.2 Java, une technologie	368
Chapitre 9 – Quelques outils actuels	369
9.1 Les outils du marché	370
9.1.1 Les outils de modélisation	370
9.1.2 Les environnements de développement	371
9.1.3 Les serveurs	376
9.2 Les outils Open Source	377
9.2.1 ArgoUML et la modélisation	378
9.2.2 L'environnement de développement Eclipse	378
9.2.3 Tomcat	379
9.2.4 JBoss : conteneur d'EJB	379
9.3 Les frameworks	380
9.3.1 Historique	380
9.3.2 Struts	381
9.3.3 Castor	389
9.2.4 Présentation des JSF : Java Server Faces	396
Index	404