

Hugues Bersini

6<sup>e</sup> édition

# La programmation orientée objet

**Cours et exercices en UML 2**  
avec Java, C#, C++, Python, PHP et LINQ

EYROLLES

# Table des matières

## **Avant-propos ..... V**

L'orientation objet (OO) en quelques mots	VI
Les grands acteurs de l'orienté objet	... VIII
Objectifs de l'ouvrage	IX
Plan de l'ouvrage	XI
À qui s'adresse ce livre ?	XII

## CHAPITRE 1

### **Principes de base : quel objet pour l'informatique ? ..... 1**

Le trio <entité, attribut, valeur>	2
Stockage des objets en mémoire	2
Types primitifs	4
Le référent d'un objet	5
Plusieurs référents pour un même objet	6
L'objet dans sa version passive	7
L'objet et ses constituants	7
Objet composite	8
Dépendance sans composition	9
L'objet dans sa version active	9
Activité des objets	9
Les différents états d'un objet	9
Les changements d'état : qui en est la cause ?	10
Comment relier les opérations et les attributs ?	10
Introduction à la notion de classe	11
Méthodes et classes	11
Sur quel objet précis s'exécute la méthode ?	13
Des objets en interaction	14
Comment les objets communiquent	14
Envoi de messages	15
Identification des destinataires de message	15

Des objets soumis à une hiérarchie	16
Du plus général au plus spécifique	16
Dépendance contextuelle du bon niveau taxonomique	17
Polymorphisme	18
Héritage bien reçu	19
Exercices	19

## CHAPITRE 2

### **Un objet sans classe... n'a pas de classe ..... 21**

Constitution d'une classe d'objets	22
Définition d'une méthode de la classe :	
avec ou sans retour	23
Identification et surcharge des méthodes par leur signature	24
La classe comme module fonctionnel	26
Différenciation des objets par la valeur des attributs	26
Le constructeur	26
Mémoire dynamique, mémoire statique	28
La classe comme garante de son bon usage	29
La classe comme module opérationnel	30
Mémoire de la classe et mémoire des objets	30
Méthodes de la classe et des instances	32
Un premier petit programme complet dans les cinq langages	32
En Java	33
EN C#	35
En C++	37
En Python	40

En PHP 5 .....	42
<b>La classe et la logistique de développement</b>	44
Classes et développement de sous-ensembles logiciels .....	44
Classes, fichiers et répertoires .....	45
<b>Exercices</b> .....	46

### CHAPITRE 3

#### **Du faire savoir au savoir-faire... du procédural à l'OO .....**

<b>Objectif objet : les aventures de l'OO</b> .....	52
Argumentation pour l'objet .....	52
Transition vers l'objet .....	53
<b>Mise en pratique</b> .....	53
Simulation d'un écosystème .....	53
<b>Analyse</b> .....	54
Analyse procédurale .....	54
Fonctions principales .....	54
<b>Conception</b> .....	55
Conception procédurale .....	55
Conception objet .....	56
<b>Conséquences de l'orientation objet</b> .....	57
Les acteurs du scénario .....	57
Indépendance de développement et dépendance fonctionnelle .....	57
Petite allusion (anticipée) à l'héritage .....	58
La collaboration des classes deux à deux ..	58

### CHAPITRE 4

#### **Ici Londres : les objets parlent aux objets.....**

Envois de messages .....	60
Association de classes .....	61
Dépendance de classes .....	64
Réaction en chaîne de messages .....	66
<b>Exercices</b> .....	67

### CHAPITRE 5

#### **Collaboration entre classes ..**

Pour en finir avec la lutte des classes .....	70
---	----

<b>La compilation Java : effet domino</b> .....	71
<b>En C#, Python, PHP 5 ou C++</b> .....	72
<b>De l'association unidirectionnelle à l'association bidirectionnelle</b> .....	74
<b>Auto-association</b> .....	77
<b>Paquets et espaces de noms</b> .....	79
<b>Exercices</b> .....	82

### CHAPITRE 6

#### **Méthodes ou messages ?.....**

<b>Passage d'arguments prédéfinis   dans les messages</b> .....	84
En Java .....	84
<i>Résultat</i> .....	84
En C# .....	85
<i>Résultat</i> .....	86
En C++ .....	86
<i>Résultat</i> .....	87
En Python .....	87
<i>Résultat</i> .....	88
En PHP 5 .....	88
<b>Passage d'argument objet dans les messages</b> ..	91
En Java .....	91
<i>Résultat</i> .....	91
En C# .....	92
<i>Résultat</i> .....	94
En PHP 5 .....	94
En C++ .....	96
<i>Résultat passage par valeur</i> .....	97
<i>Résultat passage par référent</i> .....	97
En Python .....	97
<i>Résultat</i> .....	98
<b>Une méthode est-elle d'office un message ?</b> ..	98
Même message, plusieurs méthodes .....	98
Nature et rôle, type, classe et interface :	
quelques préalables .....	99
Interface : liste de signatures de méthodes disponibles .....	100
Des méthodes strictement internes .....	101
<b>La mondialisation des messages</b> .....	101

Message sur Internet .....	101	CHAPITRE 9	
L'informatique distribuée .....	102	<b>Vie et mort des objets .....</b>	<b>135</b>
Exercices .....	103	Question de mémoire .....	136
<b>CHAPITRE 7</b>		Un rappel sur la mémoire RAM .....	136
<b>L'encapsulation</b>		L'OO coûte cher en mémoire .....	137
<b>des attributs .....</b>	<b>107</b>	Qui se ressemble s'assemble :	
Accès aux attributs d'un objet .....	108	le principe de localité .....	137
Accès externe aux attributs .....	108	Les objets intermédiaires .....	138
Cachez ces attributs que je ne saurais voir	109	Mémoire pile .....	138
Encapsulation des attributs .....	110	<i>En C++</i> .....	140
<i>En Java</i> .....	110	<i>En C#</i> .....	141
<i>En C++</i> .....	111	Disparaître de la mémoire comme	
<i>En C#</i> .....	111	de la vie réelle .....	144
<i>En PHP 5</i> .....	112	Mémoire tas .....	145
<i>En Python</i> .....	113	<b>C++ : le programmeur est le seul maître</b>	
Encapsulation : pour quoi faire ? .....	114	à bord .....	145
Pour préserver l'intégrité des objets .....	114	La mémoire a des fuites .....	146
La gestion d'exception .....	115	<b>En Java, C#, Python et PHP 5 :</b>	
Pour cloisonner leur traitement .....	117	<b>la chasse au gaspi .....</b>	<b>148</b>
Pour pouvoir faire évoluer leur traitement		<i>En Java</i> .....	149
en douceur .....	117	<i>En C#</i> .....	150
La classe : enceinte de confinement .....	119	<i>En PHP 5</i> .....	150
Exercices .....	119	Le ramasse-miettes (ou garbage collector)	151
<b>CHAPITRE 8</b>		Des objets qui se mordent la queue .....	152
<b>Les classes et leur jardin</b>		<i>En Python</i> .....	153
<b>secret .....</b>	<b>121</b>	Exercices .....	155
Encapsulation des méthodes .....	122	<b>CHAPITRE 10</b>	
Interface et implémentation .....	122	<b>UML 2.....</b>	<b>159</b>
Toujours un souci de stabilité .....	123	Diagrammes UML 2 .....	161
Signature d'une classe : son interface .....	125	Représentation graphique standardisée ..	162
Les niveaux intermédiaires		Du tableau noir à l'ordinateur .....	163
d'encapsulation .....	126	Programmer par cycles courts	
<i>Classes amies</i> .....	126	en superposant les diagrammes .....	164
<i>Une classe dans une autre</i> .....	127	Diagramme de classe et diagramme	
<i>Utilisation des paquets</i> .....	128	de séquence .....	165
Afin d'éviter l'effet papillon .....	130	Diagramme de classe .....	166
Exercices .....	132	Une classe .....	166
		<i>En Java : UML1.java</i> .....	166
		<i>En C# : UML1.cs</i> .....	167

<i>En C++ : UML1.cpp</i> . . . . .	168	Diagramme d'états-transitions . . . . .	211
<i>En Python : UML1.py</i> . . . . .	169	Exercices . . . . .	214
<i>En PHP 5 : UML1.php</i> . . . . .	169		
Similitudes et différences entre les langages .	170	<b>CHAPITRE 11</b>	
Association entre classes . . . . .	170	<b>Héritage</b> . . . . .	<b>221</b>
<i>En Java : UML 2.java</i> . . . . .	171	Comment regrouper les classes	
<i>En C# : UML 2.cs</i> . . . . .	171	dans des superclasses ? . . . . .	222
<i>En C++ : UML 2.cpp</i> . . . . .	172	Héritage des attributs . . . . .	223
<i>En Python : UML 2.py</i> . . . . .	173	Pourquoi l'addition de propriétés ? . . . . .	226
<i>En PHP 5 : UML2.php</i> . . . . .	174	L'héritage : du cognitif aux taxonomies . . . . .	226
Similitudes et différences entre les langages .	174	Interprétation ensembliste de l'héritage . . . . .	227
Pas d'association sans message . . . . .	175	Qui peut le plus peut le moins . . . . .	228
Rôles et cardinalité . . . . .	176	Héritage ou composition ? . . . . .	229
Dépendance entre classes . . . . .	184	Économiser en ajoutant des classes ? . . . . .	230
Composition . . . . .	185	Héritage des méthodes . . . . .	230
En Java . . . . .	186	Code Java . . . . .	233
<i>UML3.java</i> . . . . .	187	Code C# . . . . .	234
<i>UML3bis.java</i> . . . . .	189	Code C++ . . . . .	235
En C# . . . . .	190	Code Python . . . . .	237
<i>UML3.cs</i> . . . . .	191	Code PHP 5 . . . . .	238
<i>UML3bis.cs</i> . . . . .	192	La recherche des méthodes	
En C++ . . . . .	193	dans la hiérarchie . . . . .	239
<i>UML3.cpp</i> . . . . .	194	Encapsulation protected . . . . .	240
<i>UML3bis.cpp</i> . . . . .	196	Héritage et constructeurs . . . . .	241
En Python . . . . .	197	Premier code Java . . . . .	241
<i>UML3.py</i> . . . . .	197	Deuxième code Java . . . . .	242
<i>UML3bis.py</i> . . . . .	198	Troisième code Java : le plus logique	
En PHP 5 . . . . .	199	et le bon . . . . .	243
Classe d'association . . . . .	200	En C# . . . . .	244
Les paquets . . . . .	201	En C++ . . . . .	245
<b>Les bienfaits d'UML</b> . . . . .	202	En Python . . . . .	246
Un premier diagramme de classe		En PHP 5 . . . . .	246
de l'écosystème . . . . .	202	Héritage public en C++ . . . . .	247
Des joueurs de football qui font		<b>Le multihéritage</b> . . . . .	248
leurs classes . . . . .	202	Ramifications descendantes et ascendantes	248
Les avantages des diagrammes de classes .	202	Multihéritage en C++ et Python . . . . .	249
Un diagramme de classe simple à faire, mais		<i>Code C++ illustrant le multihéritage</i> . . . . .	250
qui décrit une réalité complexe à exécuter .	204	<i>Code Python illustrant le multihéritage</i> . . . . .	251
Procéder de manière modulaire		Des méthodes et attributs portant un même	
et incrémentale . . . . .	205	nom dans des superclasses distinctes . . . . .	252
<b>Diagramme de séquence</b> . . . . .	205		

Code C++ illustrant un premier problème lié au multihéritage	252
En Python	254
Plusieurs chemins vers une même superclasse	255
Code C++ : illustrant un deuxième problème lié au multihéritage	256
L'héritage virtuel	258
Exercices	259

## CHAPITRE 12

### Redéfinition des méthodes 265

La redéfinition des méthodes	266
Beaucoup de verbiage mais peu d'actes véritables	267
Un match de football polymorphique	268
La classe Balle	269
En Java	269
En C++	270
En C#	270
En Python	270
En PHP 5	270
La classe Joueur	270
En Java	270
En C++	271
En C#	272
En Python	273
En PHP 5	274
Précisons la nature des joueurs	274
En Java	275
En C++	276
En C#	277
En Python	278
En PHP 5	279
Passons à l'entraîneur	280
En Java	280
En C++	281
En C#	281
En Python	282
En PHP 5	282
Passons maintenant au bouquet final	282

En Java	282
Un même ordre mais une exécution différente	284
C++ : un comportement surprenant	285
Polymorphisme : uniquement possible dans la mémoire tas	289
En C#	289
En Python	291
En PHP 5	292
Quand la sous-classe doit se démarquer pour marquer	293
Les attaquants participent à un casting	294
Éviter les « mauvais castings »	295
En C++	296
En C#	296
Le casting a mauvaise presse	297
Redéfinition et encapsulation	298
Exercices	299

## CHAPITRE 13

### Abstraite, cette classe est sans objet..... 311

De Canaletto à Turner	312
Des classes sans objet	312
Du principe de l'abstraction à l'abstraction syntaxique	313
Classe abstraite	315
new et abstract incompatibles	316
Abstraite de père en fils	316
Un petit exemple dans quatre langages de programmation	317
En Java	317
En C#	318
En PHP 5	319
En C++	320
L'abstraction en Python	321
Un petit supplément de polymorphisme	322
Les enfants de la balle	322
Cliquez frénétiquement	322
Le Paris-Dakar	324
Le polymorphisme en UML	325

Exercices ..... 326  
*Exercice 13.11* ..... 335  
*Exercice 13.12* ..... 335

CHAPITRE 14

**Clonage, comparaison et affectation d'objets ..... 339**

Introduction à la classe Object ..... 340  
 Une classe à compétence universelle ..... 341  
 Code Java illustrant l'utilisation de la classe Vector et innovation de Java 5 341  
*Nouvelle version du code* ..... 342  
 Décortiquons la classe Object ..... 343  
 Test d'égalité de deux objets ..... 345  
 Code Java pour expérimenter la méthode equals(Object o) ..... 345  
 Égalité en profondeur ..... 348  
 Le clonage d'objets ..... 350  
 Code Java pour expérimenter la méthode clone() ..... 350  
 Égalité et clonage d'objets en Python ..... 354  
 Code Python pour expérimenter l'égalité et le clonage ..... 354  
 Égalité et clonage d'objets en PHP 5 ..... 355  
 Code PHP 5 pour expérimenter l'égalité et le clonage ..... 355  
 Égalité, clonage et affectation d'objets en C++ ..... 357  
 Code C++ illustrant la duplication, la comparaison et l'affectation d'objets ... 357  
 Traitons d'abord la mémoire tas ..... 361  
 Surcharge de l'opérateur d'affectation ... 363  
 Comparaisons d'objets ..... 363  
 La mémoire pile ..... 364  
 Surcharge de l'opérateur de comparaison . 364  
 Dernière étape ..... 365  
 Code C++ de la classe O1 créé automatiquement par Rational Rose ..... 366  
 En C#, un cocktail de Java et de C++ ..... 368  
 Pour les structures ..... 368

Pour les classes ..... 368  
 Code C# ..... 368  
 Exercices ..... 374

CHAPITRE 15

**Interfaces ..... 375**

Interfaces : favoriser la décomposition et la stabilité ..... 376  
 Java, C# et PHP 5 : interface et héritage .. 377  
 Les trois raisons d'être des interfaces .... 378  
 Forcer la redéfinition ..... 378  
*Code Java illustrant l'interface Comparable* . 379  
*Code Java illustrant l'interface ActionListener* ..... 381  
*Code Java illustrant l'interface KeyListener* . 382  
 Permettre le multihéritage ..... 384  
 La carte de visite de l'objet ..... 385  
*Code Java* ..... 386  
*Code C#* ..... 387  
*Code PHP 5* ..... 391  
 Les interfaces dans UML 2 ..... 392  
 En C++ : fichiers .h et fichiers .cpp ..... 393  
 Interfaces : du local à Internet ..... 397  
 Exercices ..... 397

CHAPITRE 16

**Distribution gratuite d'objets : pour services rendus sur le réseau..... 401**

Objets distribués sur le réseau : pourquoi? 402  
 Faire d'Internet un ordinateur géant .... 402  
 Répartition des données ..... 403  
 Répartition des utilisateurs et des responsables ..... 403  
 Peer-to-peer ..... 404  
 L'informatique ubiquitaire ..... 405  
 Robustesse ..... 406  
 RMI (Remote Method Invocation) ..... 406  
 Côté serveur ..... 407  
 Côté client ..... 408

RMIC : stub et skeleton . . . . .	410	L'effet du multithreading sur les diagrammes de séquence UML . . . . .	450
Lancement du registre . . . . .	411	Du multithreading aux applications distribuées . . . . .	451
<b>Corba (Common Object Request Broker Architecture)</b> . . . . .	412	<b>Des threads équirépartis</b> . . . . .	452
Un standard : ça compte . . . . .	413	En Java . . . . .	452
IDL . . . . .	413	En C# . . . . .	452
Compilateur IDL vers Java . . . . .	414	En Python . . . . .	453
Côté client . . . . .	415	<b>Synchroniser les threads</b> . . . . .	454
Côté serveur . . . . .	417	En Java . . . . .	454
Exécutons l'application Corba . . . . .	418	En C# . . . . .	456
Corba n'est pas polymorphe . . . . .	419	En Python . . . . .	459
<b>Ajoutons un peu de flexibilité à tout cela</b> .	420	<b>Exercices</b> . . . . .	462
Corba : invocation dynamique versus invocation statique . . . . .	421		
Jini . . . . .	421		
XML : pour une dénomination universelle des services . . . . .	422		
<b>Les services web sur .Net</b> . . . . .	425		
Code C# du service . . . . .	425		
WDSL . . . . .	426		
Création du proxy . . . . .	427		
Code C# du client . . . . .	428		
Soap (Simple Object Access Protocol) . .	428		
Invocation dynamique sous .Net . . . . .	429		
Invocation asynchrone en .Net . . . . .	430		
Mais où sont passés les objets ? . . . . .	432		
Un annuaire des services XML universel : UDDI . . . . .	435		
Services web versus RMI et Corba . . . . .	435		
Services web versus Windows Communication Foundation (WCF) . . .	436		
Exercices . . . . .	436		
<b>CHAPITRE 17</b>		<b>CHAPITRE 18</b>	
<b>Multithreading</b> . . . . .	<b>439</b>	<b>Programmation événementielle</b> . . . . .	<b>465</b>
Informatique séquentielle . . . . .	441	Des objets qui s'observent . . . . .	466
Multithreading . . . . .	443	En Java . . . . .	467
Implémentation en Java . . . . .	444	La plante . . . . .	467
Implémentation en C# . . . . .	446	Du côté du prédateur . . . . .	468
Implémentation en Python . . . . .	449	Du côté de la proie . . . . .	469
		Finalement, du côté de la Jungle . . . . .	469
		Résultat . . . . .	470
		<b>En C# : les délégués</b> . . . . .	470
		Généralités sur les délégués dans .Net . . .	470
		Retour aux observateurs et observables . .	474
		<i>Tout d'abord, la plante</i> . . . . .	474
		<i>Du côté du prédateur</i> . . . . .	475
		<i>Du côté de la proie</i> . . . . .	476
		<i>Finalement, du côté de la Jungle</i> . . . . .	476
		<b>En Python : tout reste à faire</b> . . . . .	479
		<b>Un feu de signalisation plus réaliste</b> . . . . .	481
		En Java . . . . .	482
		Exercices . . . . .	484
		<b>CHAPITRE 19</b>	
		<b>Persistance d'objets</b> . . . . .	<b>485</b>
		Sauvegarder l'état entre deux exécutions . .	486
		Et que dure le disque dur . . . . .	486

Quatre manières d'assurer la persistance des objets . . . . .	486
<b>Simple sauvegarde sur fichier . . . . .</b>	<b>487</b>
Utilisation des streams ou flux . . . . .	487
Qui sauve quoi ? . . . . .	488
En Java . . . . .	488
En C# . . . . .	490
En C++ . . . . .	491
En Python . . . . .	492
En PHP 5 . . . . .	493
<b>Sauvegarder les objets sans les dénaturer :</b>	
<b>la sérialisation . . . . .</b>	<b>494</b>
En Java . . . . .	495
En C# . . . . .	497
En Python . . . . .	498
Contenu des fichiers de sérialisation :	
illisible . . . . .	499
<b>Les bases de données relationnelles . . . . .</b>	<b>499</b>
SQL . . . . .	500
Une table, une classe . . . . .	500
Comment interfacer Java et C#	
aux bases de données . . . . .	502
<i>En Java</i> . . . . .	503
<i>En C#</i> . . . . .	505
Relations entre tables et associations	
entre classes . . . . .	507
<i>Relation 1-n</i> . . . . .	507
<i>Relation n-n</i> . . . . .	510
<i>Dernier problème : l'héritage</i> . . . . .	511
<b>Réservation de places de spectacles . . . . .</b>	<b>512</b>
<b>Les bases de données relationnelles-objet . . . . .</b>	<b>517</b>
SQL3 . . . . .	519
<b>Les bases de données orientées objet . . . . .</b>	<b>521</b>
OQL . . . . .	522
<b>Django et Python . . . . .</b>	<b>523</b>
<b>LINQ . . . . .</b>	<b>524</b>
Premier exemple de LINQ agissant sur une collection d'objets . . . . .	525
Second exemple de LINQ agissant sur une base de données relationnelle . . . . .	527
<b>Exercices . . . . .</b>	<b>531</b>
<b>CHAPITRE 20</b>	
<b>Et si on faisait</b>	
<b>un petit flipper ? . . . . .</b>	<b>533</b>
Généralités sur le flipper et les GUI . . . . .	534
Une petite animation en C# . . . . .	540
Retour au Flipper . . . . .	545
Code Java du Flipper . . . . .	548
Un petit tennis . . . . .	558
<b>CHAPITRE 21</b>	
<b>Les graphes . . . . .</b>	<b>565</b>
Le monde regorge de réseaux . . . . .	566
Tout d'abord : juste un ensemble d'objets . . . . .	568
Liste liée . . . . .	570
En Java . . . . .	572
En C++ . . . . .	574
La généricité en C++ . . . . .	576
La généricité en Java et C# . . . . .	580
Passons aux graphes . . . . .	586
Exercices . . . . .	590
<b>CHAPITRE 22</b>	
<b>Petites chimie, biologie</b>	
<b>et économie OO amusantes . . . . .</b>	<b>595</b>
Pourquoi de la chimie OO ? . . . . .	596
Chimie computationnelle . . . . .	596
Chimie comme plate-forme didactique . . . . .	596
Une aide à la modélisation chimique . . . . .	596
<b>Les diagrammes de classes du réacteur</b>	
<b>chimique . . . . .</b>	<b>597</b>
La classe Composant_Chimique . . . . .	598
<i>Les classes Composant_Neutre</i> <i>et Composant_Charge</i> . . . . .	599
<i>Les trois sous-classes de composants neutres</i> . . . . .	600
<i>Les trois sous-classes de composants chargés</i> . . . . .	602
La classe NœudAtomique . . . . .	603
La classe NœudMoléculaire . . . . .	603
La classe Liaison . . . . .	603
Le graphe moléculaire . . . . .	604
<i>Les règles de canonisation</i> . . . . .	606

