

# **Introducción a la programación orientada a objetos**

Cristina Cachero Castro  
Pedro J. Ponce de León Amador  
Estela Saquete Boró

Departamento de lenguajes y sistemas informáticos  
Universidad de Alicante



# Índice general

<b>1. Introducción al paradigma Orientado a Objetos</b>	<b>1</b>
1.1. Objetivos	1
1.2. El progreso de la abstracción	1
1.2.1. Abstracción	1
1.3. Principales Paradigmas de Programación	2
1.3.1. Lenguaje y pensamiento	4
1.3.2. Los lenguajes de programación y los niveles de abstracción	6
1.3.3. Mecanismos de abstracción	9
1.4. El paradigma orientado a objetos	12
1.4.1. Motivación	12
1.4.2. El paradigma orientado a objetos	12
1.4.3. Un nuevo modo de ver el mundo	15
1.4.4. Características básicas de un lenguaje OO	16
1.4.5. Características opcionales	20
1.5. Historia de los lenguajes orientados a objetos	21
1.5.1. Simula	21
1.5.2. Smalltalk	21
1.5.3. Los años 80	21
1.5.4. C++	22
1.5.5. Java	22
1.5.6. Eiffel	22
1.5.7. Presente	22
1.6. Metas del paradigma orientado a objetos	23
1.6.1. Principales parámetros de calidad extrínsecos	23
1.6.2. Principales parámetros intrínsecos	23
1.7. Conclusiones/Resumen	25
<b>2. Fundamentos de la programación orientada a objetos</b>	<b>27</b>
2.1. Objetivos	27
2.2. Clases	27
2.2.1. Partes de la definición de una clase	28
2.2.2. Visibilidad	28
2.2.3. Clase versus Tipo abstracto de dato	29
2.3. Atributos	30
2.3.1. Tipos de atributo	31
2.3.2. Inicialización de atributos	32

## Índice general

2.3.3. Atributos de clase . . . . .	32
2.3.4. Atributos constantes . . . . .	39
2.4. Operaciones . . . . .	40
2.4.1. Tipos de operaciones . . . . .	41
2.4.2. Un poco de UML . . . . .	42
2.4.3. Constructor . . . . .	44
2.4.4. Constructor de copia . . . . .	48
2.4.5. Constructor versus método . . . . .	51
2.4.6. Destructor . . . . .	53
2.4.7. Forma canónica ortodoxa de una Clase . . . . .	55
2.5. El concepto de interfaz . . . . .	56
2.6. El concepto de objeto . . . . .	57
2.6.1. Estado y Comportamiento . . . . .	57
2.6.2. Creación e Inicialización . . . . .	58
2.6.3. Punteros y ubicación en memoria . . . . .	59
2.7. Metaclases . . . . .	59
2.8. El diseño de aplicaciones OO . . . . .	59
2.9. Relaciones entre clases y relaciones entre objetos . . . . .	61
2.9.1. Asociación . . . . .	63
2.9.2. Relaciones todo-parte . . . . .	70
2.9.3. Relación de Uso (Dependencia) . . . . .	78
2.10. Conclusiones . . . . .	79
2.11. Ejercicios resueltos . . . . .	80
2.12. Ejercicios propuestos . . . . .	88
<b>3. HERENCIA</b> . . . . .	<b>89</b>
3.1. Introducción a la Herencia . . . . .	89
3.1.1. Objetivos . . . . .	89
3.1.2. Motivación . . . . .	90
3.1.3. Definición de herencia . . . . .	91
3.1.4. Herencia como implementación de la generalización . . . . .	91
3.1.5. Tipos de herencia . . . . .	92
3.1.6. Caracterización de la herencia . . . . .	93
3.1.7. Herencia en C++ . . . . .	94
3.1.8. Sintaxis . . . . .	95
3.2. Herencia Simple . . . . .	99
3.2.1. Introducción . . . . .	99
3.2.2. El constructor y destructor en Herencia Simple . . . . .	100
3.2.3. Orden de las llamadas del constructor y destructor . . . . .	100
3.2.4. Ejemplo de Herencia Simple . . . . .	101
3.2.5. Particularidades de la Herencia . . . . .	108
3.2.6. Ejercicios . . . . .	109
3.3. Herencia Múltiple . . . . .	112
3.3.1. Introducción . . . . .	112

3.3.2. Uso de ámbito en Herencia Múltiple . . . . .	114
3.3.3. Uso de virtual en Herencia Múltiple . . . . .	114
3.3.4. Ejercicio . . . . .	115
3.4. Herencia de Interfaz . . . . .	116
3.4.1. Clases Abstractas . . . . .	118
3.4.2. Ejercicios . . . . .	119
3.5. Herencia de Implementación . . . . .	123
3.5.1. Uso seguro de la Herencia de Implementación . . . . .	123
3.5.2. Uso inseguro de la Herencia de Implementación . . . . .	124
3.5.3. Herencia de construcción en C++ (Herencia de implementación pura) . . . . .	125
3.6. Beneficios y costes de la herencia . . . . .	125
3.6.1. Beneficios . . . . .	125
3.6.2. Costes . . . . .	126
3.7. Elección de la técnica de reuso . . . . .	127
3.7.1. Introducción . . . . .	127
3.7.2. Uso de composición (Layering) . . . . .	128
3.7.3. Uso de herencia . . . . .	129
3.7.4. Composición vs. Herencia. Ejemplo . . . . .	129
3.7.5. Ejercicios . . . . .	131
3.8. Conclusiones . . . . .	134
3.9. Ejercicios propuestos . . . . .	135
<b>4. POLIMORFISMO . . . . .</b>	<b>141</b>
4.1. Polimorfismo y reuso . . . . .	141
4.1.1. Objetivos . . . . .	141
4.1.2. Motivación . . . . .	142
4.1.3. Tiempo de enlace en los lenguajes de programación . . . . .	147
4.2. Sobrecarga . . . . .	150
4.2.1. Conceptos previos: signatura y ámbito . . . . .	150
4.2.2. Sobrecarga basada en ámbito . . . . .	152
4.2.3. Sobrecarga basada en signaturas de tipo . . . . .	152
4.2.4. Alternativas a la sobrecarga . . . . .	162
4.3. Polimorfismo en jerarquías de herencia . . . . .	168
4.3.1. Redefinición . . . . .	168
4.3.2. Shadowing . . . . .	170
4.3.3. Sobrescritura . . . . .	171
4.4. Variables Polimórficas . . . . .	187
4.4.1. Variables polimórficas simples . . . . .	188
4.4.2. La variable receptora . . . . .	190
4.4.3. Downcasting . . . . .	192
4.4.4. Polimorfismo Puro . . . . .	195
4.5. Genericidad . . . . .	196
4.5.1. Funciones genéricas en C++ . . . . .	196
4.5.2. Plantillas de Clase en C++ . . . . .	200

## Índice general

4.5.3. Constantes en plantillas en C++ . . . . .	203
4.5.4. Herencia en clases genéricas . . . . .	204
4.5.5. Relaciones entre instancias de plantillas . . . . .	205
4.5.6. Diferencia entre herencia y genericidad . . . . .	206
4.6. Caso de estudio . . . . .	206
4.7. Conclusiones . . . . .	209
4.8. Ejercicios Resueltos . . . . .	212
4.9. Ejercicios Propuestos . . . . .	232
4.10. Ejercicios de entrega optativa . . . . .	237
<b>5. GESTION DE ERRORES Y OTRAS CARACTERISTICAS</b>	<b>239</b>
5.1. Gestión de errores . . . . .	239
5.1.1. Motivación . . . . .	239
5.1.2. Concepto de excepción . . . . .	240
5.1.3. El mecanismo de excepciones en C++ y Java . . . . .	240
5.1.4. Lanzamiento de excepciones . . . . .	241
5.1.5. Ejemplo: Excepciones definidas por el programador . . . . .	242
5.1.6. Ejemplo: Excepciones en apertura de ficheros . . . . .	243
5.1.7. Excepciones predefinidas . . . . .	244
5.1.8. Ventajas del uso de excepciones . . . . .	245
5.1.9. Excepciones en Java . . . . .	245
5.2. Otras características . . . . .	246
5.2.1. Persistencia . . . . .	246
5.2.2. Concurrencia . . . . .	246
5.2.3. Aserciones . . . . .	246
5.2.4. Recogida de basura . . . . .	246
5.2.5. Reflexión . . . . .	247
5.3. Conclusiones . . . . .	248
5.4. Ejercicios propuestos . . . . .	248

# Índice de cuadros

1.1. Mecanismos de abstracción . . . . .	8
1.2. Código a diferentes niveles de abstracción . . . . .	10
2.1. Constructores versus otros métodos . . . . .	52
2.2. Comportamiento de una cuenta bancaria . . . . .	58
2.3. Ejemplos de multiplicidad de una relación . . . . .	64
3.1. Visibilidades para clases derivadas en función del tipo de herencia . . . . .	96
4.1. Operadores sobrecargables en C++ . . . . .	156
4.2. Programas de depuración de la sobrecarga del operador [] . . . . .	159
4.3. Resultados de ejecutar los programas de la tabla 4.2 con cada una de las cinco implementaciones del operador [] . . . . .	160
4.4. Diferencias entre redefinición, <i>shadowing</i> y sobrescritura . . . . .	180





# Índice de figuras

1.1. Ejemplo de programación funcional con Scheme . . . . .	3
1.2. El clásico 'Hola mundo' escrito en ensamblador para procesadores x86 . . . . .	6
1.3. Distintos lenguajes de programación abstraen conceptos diferentes . . . . .	9
1.4. Ocultación de información versus encapsulación. . . . .	11
1.5. Funcionamiento de un motor representado por un par de variables . . . . .	14
1.6. Ejemplo de métodos en CLOS. . . . .	17
1.7. ejemplo de métodos en Python . . . . .	17
1.8. Dos jerarquías de clases . . . . .	19
2.1. Esquema de una clase en UML . . . . .	28
2.2. clase Empleado . . . . .	34
2.3. clase Empleado con variable estática pública . . . . .	36
2.4. Ejemplo de clase en UML . . . . .	43
2.5. Efecto en memoria dinámica del constructor de copia de oficio . . . . .	50
2.6. Efecto de un constructor de copia definido por el programador . . . . .	52
2.7. Ejemplo de interfaz en UML . . . . .	57
2.8. Representación en UML de un objeto . . . . .	57
2.9. Esquema de la memoria de un proceso . . . . .	60
2.10. Ejemplo de metaclasses . . . . .	60
2.11. Relaciones entre clases y objetos . . . . .	62
2.12. Diversas asociaciones entre objetos . . . . .	63
2.13. Asociación entre una pieza y una casilla en el juego de los barcos . . . . .	64
2.14. Asociación entre un barco y una casilla . . . . .	65
2.15. Asociación entre trabajadores y proyectos . . . . .	68
2.16. multiplicidades en una relación todo-parte entre objetos . . . . .	72
2.17. Multiplicidad en una composición . . . . .	72
2.18. Relación todo-parte: taller de bicicletas . . . . .	74
2.19. Interpretación de la realidad en función del tipo de relación todo-parte . . . . .	76
2.20. Relación de uso entre Coche y Gasolinera . . . . .	79
2.21. Clase Empleado . . . . .	81
2.22. Clase Asignatura . . . . .	82
3.1. Ejemplo de Herencia Simple . . . . .	92
3.2. Ejemplo de Herencia Múltiple . . . . .	92
3.3. Ejemplo combinación atributos en Herencia . . . . .	93
3.4. Ejemplo combinación atributos en Herencia . . . . .	94

## Índice de figuras

3.5. Clase Base . . . . .	95
3.6. Clase Base y Derivada . . . . .	97
3.7. UML ejemplo . . . . .	98
3.8. UML ejemplo Herencia Simple . . . . .	99
3.9. UML ejemplo . . . . .	101
3.10.UML ejemplo Herencia simple . . . . .	104
3.11.UML Herencia cuenta empresarial . . . . .	107
3.12.UML Ejercicio 1 . . . . .	109
3.13.UML Ejercicio 2 . . . . .	110
3.14.Relaciones de Alejandro . . . . .	112
3.15.Representacion UML herencia múltiple . . . . .	112
3.16.Representacion UML herencia múltiple para cuentas . . . . .	113
3.17.Representacion UML herencia múltiple en 3 niveles . . . . .	115
3.18.Representacion UML ejercicio herencia múltiple . . . . .	116
3.19.Representacion UML . . . . .	117
3.20.Representacion UML herencia interfaz . . . . .	119
3.21.Representacion UML ejercicio herencia interfaz . . . . .	119
3.22.Representacion UML ejercicio herencia interfaz (II) . . . . .	121
3.23.Representacion UML ejercicio herencia implementación . . . . .	123
3.24.Representacion UML herencia restricción . . . . .	124
3.25.Representacion UML herencia de generalización . . . . .	124
3.26.Representacion UML ejercicio herencia conveniencia . . . . .	125
3.27.Representacion UML HERENCIA . . . . .	131
3.28.Representacion UML COMPOSICION . . . . .	132
3.29.Solución con herencia . . . . .	132
3.30.Solución con composición . . . . .	133
3.31.Solución con herencia y composición . . . . .	133
3.32.Solución varias copias misma película . . . . .	133
3.33.Solución ejercicio . . . . .	138
4.1. Diagrama de clases: clasificación personas de la UA . . . . .	172
4.2. Diagrama de clases: sistema de gestión de mascotas . . . . .	177
4.3. Diagrama de clases simplificado: juego Hundir la Flota . . . . .	188
4.4. Procedimiento para identificar la sobrecarga . . . . .	210
4.5. Procedimiento para identificar sobrescritura, redefinición y <i>shadowing</i> . . . . .	211
4.6. Jerarquía de herencia de Persona . . . . .	212
4.7. Jerarquía de herencia de TCuenta . . . . .	219
4.8. Diagrama clases: jerarquía de herencia de Empleado . . . . .	222
4.9. Diagrama de clases: jerarquía genérica de herencia . . . . .	226
4.10.Clase Tiempo . . . . .	233
4.11.Clase Cadena . . . . .	233
4.12.Clase Punto . . . . .	234
4.13.Diagrama de clases: TablaAsociativa . . . . .	234
4.14.Clase Vector . . . . .	235