

PROGRAMACIÓN ORIENTADA A OBJETOS EN JAVA

Buenas prácticas

Miguel Hernández Bejarano
Luis Eduardo Baquero Rey

edú

Informática

PROGRAMACIÓN ORIENTADA A OBJETOS EN JAVA

Buenas prácticas

Miguel Hernández Bejarano
Luis Eduardo Baquero Rey

edü[®]

Conocimiento a su alcance

Bogotá - México, D.F.

Hernández Bejarano, Miguel, *et al.*

Programación orientada a objetos en Java/ Miguel Hernández Bejarano y Luis Eduardo Baquero Rey -- 1a. Edición. Bogotá : Ediciones de la U, 2023.

316 p. ; 24 cm.

ISBN 978-958-792-463-3

e-ISBN 978-958-792-464-0

1. Informática 2. Lenguajes de programación 3. Programación orientada a objetos en Java I. Tít.

621.39 ed.

Área: Informática

Primera edición: Bogotá, Colombia, enero de 2023

ISBN. 978-958-792-463-3

© Miguel Hernández Bejarano

© Luis Eduardo Baquero Rey

© Ediciones de la U - Carrera 27 # 27-43 - Tel. (+57-1) 3203510 - 3203499

www.edicionesdelau.com - E-mail: editor@edicionesdelau.com

Bogotá, Colombia

Ediciones de la U es una empresa editorial que, con una visión moderna y estratégica de las tecnologías, desarrolla, promueve, distribuye y comercializa contenidos, herramientas de formación, libros técnicos y profesionales, e-books, e-learning o aprendizaje en línea, realizados por autores con amplia experiencia en las diferentes áreas profesionales e investigativas, para brindar a nuestros usuarios soluciones útiles y prácticas que contribuyan al dominio de sus campos de trabajo y a su mejor desempeño en un mundo global, cambiante y cada vez más competitivo.

Coordinación editorial: Adriana Gutiérrez M.

Carátula: Ediciones de la U

Impresión: DGP Editores SAS

Calle 63 No. 70 D - 34, Pbx. (571) 7217756

Impreso y hecho en Colombia

Printed and made in Colombia

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro y otros medios, sin el permiso previo y por escrito de los titulares del Copyright.

Dedicatoria

A Dios, a mi esposa Flor Ángela,
a mis hijos Jefferson, Julieth y Óscar
y a mi nieto Alejandro
Miguel

A Cristal, Isabela y Rocío
Eduardo

Contenido

Prólogo.....	17
--------------	----

Capítulo 1. Fundamentos de la Programación Orientada a Objetos (POO).....	21
--	-----------

1.1 Temática a desarrollar	21
1.2 Introducción.....	21
1.3 Programación orientada a objetos	23
1.4 Principios de la programación orientada a objetos.....	23
1.5 Ventajas del uso de la programación orientada a objetos	24
1.6 Desventajas del uso de la programación orientada a objetos.....	24
1.7 Lenguajes de programación orientada a objetos	24
1.8 Clases y objetos.....	25
1.8.1 Clases.....	25
1.8.2 Objeto	29
1.9 Visibilidad en atributos y métodos	34
1.10 Paquetes.....	35
1.11 Lecturas recomendadas	36
1.12 Preguntas revisión de conceptos	37
1.13 Ejercicios	37
1.14 Bibliografía	38

Capítulo 2. Componentes de un programa	39
---	-----------

2.1 Temática a desarrollar	39
2.2 Introducción.....	39
2.3 Características de un programa en Java	40
2.4 Estructura de un programa en Java.....	40
2.4.1 Declaración de importaciones (import)	40
2.4.2 Definición de clases	41

2.5 Presentación del programa	43
2.6 Construcción de un programa en Java.....	43
2.7 Palabras reservadas	44
2.8 Los identificadores.....	45
2.9 Tipos de datos.....	45
2.10 Variables	47
2.11 Constantes.....	48
2.12 Comentarios	48
2.13 Definición de variables	50
2.14 Definición de constantes	50
2.15 Operadores aritméticos.....	51
2.16 Los separadores.....	52
2.17 Lecturas recomendadas	52
2.18 Preguntas de revisión de conceptos.....	52
2.19 Ejercicios	53
2.20 Referencias bibliográficas	53

Capítulo 3. Herramientas para el desarrollo de la programación orientada a objetos 55

3.1 Temática a desarrollar.....	55
3.2 Introducción.....	55
3.3 Entornos Integrados de Desarrollo (IDE)	56
3.3.1 Eclipse.....	57
3.3.2 NetBeans	57
3.3.3 BlueJ.....	58
3.3.4 GreenFoot	59
3.3.5 IntelliJ IDEA.....	59
3.4 Herramientas de modelamiento	61
3.4.1 StarUML	61
3.4.2 Dia.....	61
3.4.3 ArgoUML	61
3.4.4 Rational Software Modeler	62
3.5 Java como lenguaje de programación.....	62
3.6 Lecturas recomendadas.....	64
3.7 Preguntas de revisión de conceptos	64
3.8 Ejercicios.....	65

3.9 Referencias bibliográficas.....	65
-------------------------------------	----

Capítulo 4. Métodos constructores, set y get..... 67

4.1 Temática a desarrollar	67
4.2 Introducción.....	67
4.3 Métodos.....	68
4.3.1 Método constructor.....	68
4.4 Modificadores de acceso	72
4.5 Métodos set y get.....	78
4.6 Métodos de instancias.....	83
4.7 Estructura de un método	84
4.8 Prueba de escritorio	85
4.9 Métodos que no retornan valor	88
4.10 Requerimientos	88
4.11 Un desarrollo completo.....	89
4.12 Entrada y salida de datos	95
4.13 Entrada y salida estándar	96
4.14 Lecturas recomendadas	106
4.15 Preguntas de revisión de conceptos.....	106
4.16 Ejercicios	107
4.17 Bibliografía	107

Capítulo 5. Condicionales 109

5.1 Temática a desarrollar	109
5.2 Introducción.....	109
5.3 Estructuras de control condicionales.....	110
5.4 Operadores relacionales	110
5.5 Operadores lógicos.....	112
5.6 Tipos de condicionales	114
5.6.1 Condicionales simples	114
5.6.2 Condicionales compuestas	119
5.6.3 Condicionales anidadas.....	122
5.7 Selección múltiple.....	131
5.8 Operador condicional (?)	135
5.9 Preguntas de revisión de conceptos	138

5.10 Lecturas recomendadas	139
5.11 Ejercicios	139
5.12 Bibliografía	142
Capítulo 6. Ciclos	143
6.1 Temática a desarrollar	143
6.2 Introducción	143
6.3 Estructuras de control repetitivas o ciclos.....	145
6.3.1 Estructura de repetición while (mientras).....	145
6.3.2 Estructura de repetición do-while	152
6.3.3 Estructura de repetición for	156
6.4 Comparación de los ciclos while, do-while, for	160
6.5 Ciclo for-each (for mejorado)	161
6.6 Uso de los ciclos repetitivos	162
6.7 Lecturas recomendadas.....	162
6.8 Preguntas y ejercicios de revisión de conceptos	163
6.9 Ejercicios	163
6.10 Bibliografía	163
Capítulo 7. Cadenas	165
7.1 Temática a desarrollar	165
7.2 Introducción	165
7.3 Clase String	167
7.4 Obtener cadenas desde las primitivas.....	174
7.5 Obtener primitivas desde las cadenas.....	174
7.6 Lecturas recomendadas.....	184
7.7 Preguntas de revisión de conceptos	184
7.8 Ejercicios	185
7.9 Referencias bibliográficas.....	186
Capítulo 8. Relaciones entre clases	189
8.1 Temática a desarrollar	189
8.2 Introducción.....	189
8.3 Elementos entre las relaciones de clases.....	191

8.3.1 Cardinalidad	191
8.3.2 Asociación	192
8.3.3 Agregación	194
8.3.4 La composición	196
8.4 La relación de especialización/generalización	197
8.5 Dependencia.....	198
8.6 Lecturas recomendadas.....	199
8.7 Preguntas de evaluación	199
8.8 Ejercicios.....	199
8.9 Bibliografía.....	202
Capítulo 9. Abstracción y encapsulamiento.....	203
9.1 Temática a desarrollar	203
9.2 Introducción.....	203
9.3 Abstracción.....	203
9.4 Encapsulamiento.....	206
9.5 Lecturas recomendadas.....	209
9.6 Preguntas de revisión de conceptos	209
9.7 Ejercicios.....	210
9.8 Bibliografía.....	211
Capítulo 10. Herencia	213
10.1 Temática a desarrollar.....	213
10.2 Introducción	213
10.3 Herencia	215
10.4 Representación de la herencia.....	219
10.5 Sentencia super.....	225
10.6 Lecturas recomendadas	228
10.7 Preguntas de revisión de conceptos.....	228
10.8 Ejercicios	229
10.9 Referencias bibliográficas	231
Capítulo 11. Polimorfismo	233
11.1 Temática a desarrollar.....	233

11.2 Introducción	233
11.3 Polimorfismo	234
11.4 Sobrecarga de métodos	234
11.5 Sobrecarga de constructores.....	238
11.6 Sobreescritura de métodos.....	241
11.7 Enlace dinámico	247
11.8 Lecturas recomendadas	250
11.9 Preguntas de revisión de conceptos.....	251
11.10 Ejercicios	251
11.11 Bibliografía.....	252

Capítulo 12. Interfaz y clases abstractas 253

12.1 Temática a desarrollar.....	253
12.2 Introducción	253
12.3 Interfaz.....	254
12.4 Clases abstractas	258
12.5 Preguntas de revisión de conceptos.....	260
12.6 Ejercicios	261
12.7 Bibliografía	265

Capítulo 13. Modelado orientado a objetos y aplicaciones de software 267

13.1 Temática a desarrollar.....	267
13.2 Introducción	267
13.3 Análisis orientado a objetos.....	268
13.4 Escenarios	269
13.5 Prototipo	270
13.6 Diagramas de casos de uso.....	270
13.7 Diseño orientado a objetos.....	277
13.8 Diagramas UML (Unified Modeling Language)	280
13.9 Aplicaciones de software	281
13.10 Lecturas recomendadas.....	299
13.11 Preguntas de revisión de conceptos	299
13.12 Ejercicios	299
13.13 Bibliografía.....	301

Capítulo 14. Documentación	303
14.1 Temática a desarrollar.....	303
14.2 Introducción	303
14.3 Javadoc	303
14.4 Javadoc en los IDE	309
14.5 API de Java	313
14.6 Lecturas recomendadas	313
14.7 Preguntas de revisión de conceptos.....	313
14.8 Ejercicios	314
14.9 Bibliografía	315

Índice de figuras

Figura 1. Mapa mental de programación orientada a objetos	22
Figura 2. Mapa mental de clase.....	26
Figura 3. Partes del diagrama de clase	27
Figura 4. Mapa mental de objeto	30
Figura 5. Ejemplos del objeto reloj	32
Figura 6. Diagrama de paquete.....	36
Figura 7. Diagrama de paquetes con dos clases comunes.....	36
Figura 8. Mapa mental de tipos de datos	46
Figura 9. Pantalla de carga de Eclipse LUNA.....	57
Figura 10. Pantalla inicio en NetBeans.....	58
Figura 11. Pantalla de inicio de BlueJ	58
Figura 12. Pantalla inicio de GreenFoot.....	59
Figura 13. Pantalla IntelliJ IDEA	60
Figura 14. Concepto de Java	63
Figura 15. Concepto plataformas de Java	64
Figura 16. Mapa mental del concepto de métodos.....	69
Figura 17. Mapa mental de método constructor.....	70
Figura 18. Mapa mental de los métodos set y get.....	79
Figura 19. Concepto de las estructuras condicionales	111
Figura 21. Mapa mental de ciclos.....	144
Figura 20. Mapa mental de la clase String.....	166
Figura 22. Mapa mental de relaciones entre clases	190
Figura 23. Mapa mental de abstracción	205
Figura 24. Mapa mental de encapsulamiento	207
Figura 25. Mapa mental de herencia.....	214
Figura 26. Mapa mental de polimorfismo.....	235
Figura 27. Mapa mental del modelado orientado a objetos	268
Figura 27a. Mapa mental de aplicaciones de software	283
Figura 28. Documentación en BlueJ.....	309
Figura 29. Generación Javadoc en Eclipse	310
Figura 30. Documentación Javadoc en Eclipse	311
Figura 31. Generación Javadoc en Netbeans	312
Figura 32. Documentación Javadoc en Netbeans.....	312
Figura 33. API Java 8.....	314

Índice de tablas

Tabla 1. Ejemplos de objetos asociados a su correspondiente clase.....	25
Tabla 2. Ejemplo de objeto	29
Tabla 3. Ejemplo estructura interna de un objeto	34
Tabla 4. Palabras reservadas en Java	44
Tabla 5. Tipos de datos de Java.....	47
Tabla 6. Operadores aritméticos	51
Tabla 7. Operadores relacionales.....	112
Tabla 8. Operadores lógicos	113
Tabla 9. Resultado operadores lógicos	113
Tabla 10. Apoyo implementación del método	117
Tabla 11. Apoyo parte de análisis	126
Tabla 12. Ejemplo de cadenas.....	167
Tabla 13. Métodos básicos	168
Tabla 14. Tipos de cardinalidad	191
Tabla 15. Comparación entre abstracción y encapsulamiento.....	209

Prólogo

En el quehacer diario de la docencia, y más concretamente en el área de la programación, surgen muchos temas, talleres y ejercicios para trabajar con los estudiantes. En tal sentido, este libro es el resultado de varios años de experiencias donde se retroalimentan carencias y éxitos en busca de buenas prácticas de la Programación Orientada a Objetos (POO); para tal efecto, se utiliza Java como lenguaje de programación de alto nivel. Está diseñado para cualquier estudiante que desea hacer inmersión en la programación de computadores, independientemente del área o disciplina del conocimiento, pero, eso sí, cambiando el paradigma de la programación estructurada a la de los objetos, como en la vida real.

Se tienen en cuenta elementos de buenas prácticas de la programación con la finalidad de hacer fácil la lectura del código, así como su mantenimiento, facilitando la escalabilidad del mismo, la reutilización y la integración de manera homogénea, estandarizando el desarrollo, basado en convenios y reglas sencillas, y aportando higiene en la codificación de los programas fuente.

Con el propósito de lograr lo anterior, esta obra se divide en catorce capítulos, cada uno distribuido de la siguiente manera: relación general de la temática a desarrollar, una introducción al tema central donde se complementa con un mapa mental, el desarrollo de la temática respectiva incluyendo ejemplos, se recomiendan unas lecturas que posibilitan ampliar el tema, unas preguntas de revisión de conceptos, ejercicios propuestos y, finalmente, se propone una bibliografía complementaria.

En el capítulo 1, se tratan temas fundamentales de este paradigma, como el de la programación orientada a objetos, los principios de la POO y los conceptos de clase, objeto, atributo y método.

En el capítulo 2, se estudian los componentes de un programa, donde se involucran identificadores, tipos de datos, palabras reservadas, manejo de comentarios, variables, constantes, las expresiones, la estructura de un programa de Java, la entrada y la salida de datos y los tipos de operadores.

En el capítulo 3, se estudian las herramientas necesarias para el desarrollo de la programación orientada a objetos, como los entornos integrados de desarrollo, herramientas de modelamiento y el lenguaje de programación Java.

En el capítulo 4, se hace referencia a los métodos, qué son los métodos constructores, los métodos set y get y los métodos modificadores.

En el capítulo 5, se estudia todo lo relacionado con los condicionales, en su orden, las estructuras de control, los condicionales simple, compuesto y anidado, los operadores lógicos, selección múltiple y el operador condicional ?.

En el capítulo 6, se estudia el tema relacionado con el manejo de cadenas en Java, incluyendo los métodos asociados a la clase String.

En el capítulo 7, se hace referencia a los ciclos o instrucciones repetitivas que se pueden trabajar en un programa Java.

En el capítulo 8, se estudian las diferentes relaciones entre clases, como la cardinalidad, asociación, agregación, composición, generalización y dependencia.

En el capítulo 9, se involucran los temas de abstracción y encapsulamiento, como mecanismos de la programación orientada a objetos.

En el capítulo 10, se estudia el concepto de herencia y la terminología asociada a este concepto, como lo que es una superclase, subclase, protected, extends y super.

En el capítulo 11, se hace referencia al mecanismo de la programación orientada a objetos denominado polimorfismo, incluyendo polimorfismo de subtipo, sobrecarga de métodos y sobrescritura de métodos.

En el capítulo 12, se continúa con el concepto y manejo de interfaz y clases abstractas en Java.

En el capítulo 13, se estudian los aspectos básicos a tener en cuenta a la hora de desarrollar un proyecto o aplicación de software de comienzo a fin, desde el análisis orientado a objetos, el diseño, el modelamiento y la aplicación de software final.

En el capítulo 14, por último, se presentan opciones desde el lenguaje de programación Java para la respectiva documentación de los programas y

aplicaciones, con herramientas con Javadoc, soporte Javadoc para los IDE y la API de Java.

El estudiante interesado en el tema podría orientarlo también a cualquier otro lenguaje de programación de alto nivel, diseñado para trabajar con el paradigma de la programación orientada a objetos.

Capítulo 1

Fundamentos de la Programación Orientada a Objetos (POO)

1.1 Temática a desarrollar

- Programación orientada a objetos
- Principios de la POO
- Clase
- Objeto
- Atributo
- Método

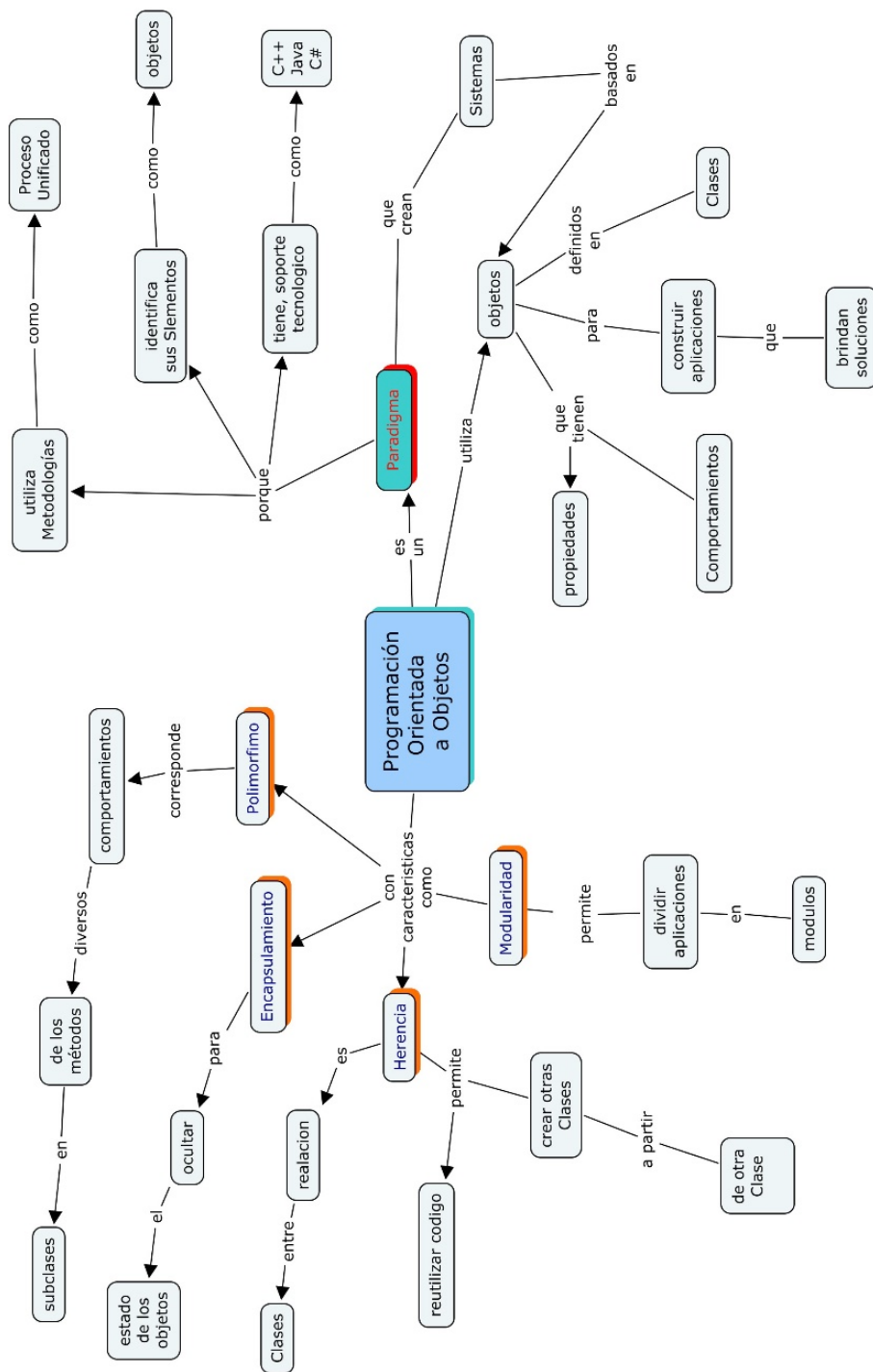
1.2 Introducción

La concepción del mundo en términos de Programación Orientada a Objetos (POO) está integrada por un conjunto de entidades u objetos relacionados entre sí, los cuales pueden ser: personas, árboles, automóviles, casas, teléfonos y computadores; en general, todo elemento que tiene atributos y que pueden ser palpables a través de los sentidos.

El ser humano tiene una apreciación de su ambiente en términos de objetos, por lo cual le resulta simple pensar de la misma manera cuando diseña un modelo para la solución de un problema, puesto que se tiene el concepto o pensamiento de objetos de manera intrínseca.

La Programación Orientada a Objetos (*Object Oriented Programming*) tiene como ejes centrales las clases y los objetos, donde una clase corresponde a la agrupación de datos (atributos) y acciones (métodos). Una clase describe un conjunto de objetos con propiedades y comportamientos similares. La figura 1 muestra, mediante un concepto de mapa mental, este concepto y todo lo que le rodea.

Figura 1
Mapa mental de programación orientada a objetos



El buen uso de este paradigma de programación se convierte en una buena práctica que arrojaría como resultados la construcción de programas de calidad, facilitar su mantenimiento y la reutilización de programas, entre otros aspectos. El tomar objetos de un problema del mundo real y extraer sus características y comportamientos, y representarlos formalmente en diagramas UML, mediante los diagramas de clases y diagramas de objetos, son de las primeras actividades inmersas en este estilo de programación.

1.3 Programación orientada a objetos

La Programación Orientada a Objetos (POO) es un paradigma o *modelo de programación*; esto indica que no es un lenguaje de programación específico o una tecnología, sino una *forma de programar* donde lo que se intenta es llevar o modelar el mundo real a un conjunto de entidades u objetos que están relacionados y se comunican entre ellos como elementos constitutivos del software, dando solución a un problema en términos de programación.

En síntesis, el elemento básico de este paradigma es el objeto, el cual contiene toda la información necesaria que se abstrae del mundo del problema, los datos que describen su estado y las operaciones que pueden modificar dicho estado, determinando las capacidades del objeto.

1.4 Principios de la programación orientada a objetos

Abstracción: es la capacidad de determinar los detalles fundamentales de un objeto mientras se ignora el entorno; por ejemplo, la información de un cliente en una factura requiere el documento, el nombre, la dirección y el teléfono.

Encapsulamiento: es la capacidad de agrupar en una sola unidad datos y métodos. Por ejemplo, para calcular el área de un triángulo, se deben tener la base y la altura; o cuando se ve la televisión, no hay preocupación por el modo cómo este funciona, lo único que se realiza es manipular el control y disfrutar del programa, película, entre otros ejemplos.

Herencia: es el proceso en el que un objeto adquiere las propiedades de otro a partir de una clase base, de la cual se heredan todas sus propiedades, métodos y eventos; estos, a su vez, pueden o no ser implementados y/o modificados;

por ejemplo, la herencia de bienes que se transmite de padre a hijo, pero el padre tiene la potestad de desheredar.

Polimorfismo: es una propiedad que permite enviar el mismo mensaje a objetos de diferentes clases de forma que cada uno de ellos responde al mismo mensaje de diferente modo; por ejemplo, el acto de comer para proveer nutrientes al organismo es diferente en los siguientes animales: el león (carnívoro), el canario (alpiste) y el perro (todo tipo de alimento).

Modularidad: es la propiedad que permite dividir una aplicación de software en unidades o partes más pequeñas, denominadas módulos, donde cada una de ellas puede ser independiente de la aplicación y de los restantes módulos. Por ejemplo, el software institucional de una empresa que puede ser dividido en subprogramas como contabilidad, presupuesto, facturación, entre otros.

1.5 Ventajas del uso de la programación orientada a objetos

- Simula el sistema del mundo real.
- Facilita el mantenimiento del software.
- Facilita el trabajo en equipo.
- Modela procesos de la realidad.
- Genera independencia e interoperabilidad de la tecnología.
- Incrementa la reutilización del software.

1.6 Desventajas del uso de la programación orientada a objetos

- Complejidad para adaptarse.
- Mayor cantidad de código (aunque se tiene la posibilidad de volver, de ser reutilizable).

1.7 Lenguajes de programación orientada a objetos

Permite el desarrollo de aplicativos de software que interactúan con los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones.

**Ejemplo**

- C++
- ADA
- Java
- SmallTalk
- C#
- Php versión 5.0 en adelante

1.8 Clases y objetos

Las palabras “clase” y “objeto” son utilizadas en la programación orientada a objetos, la cual planea simular u organizar datos en conjuntos modulares de elementos de información del mundo real.

1.8.1 Clases

Asumida como un molde o plantilla empleada para crear objetos, lo que permite considerar que las clases agrupan a un conjunto de objetos similares.

**Ejemplo**

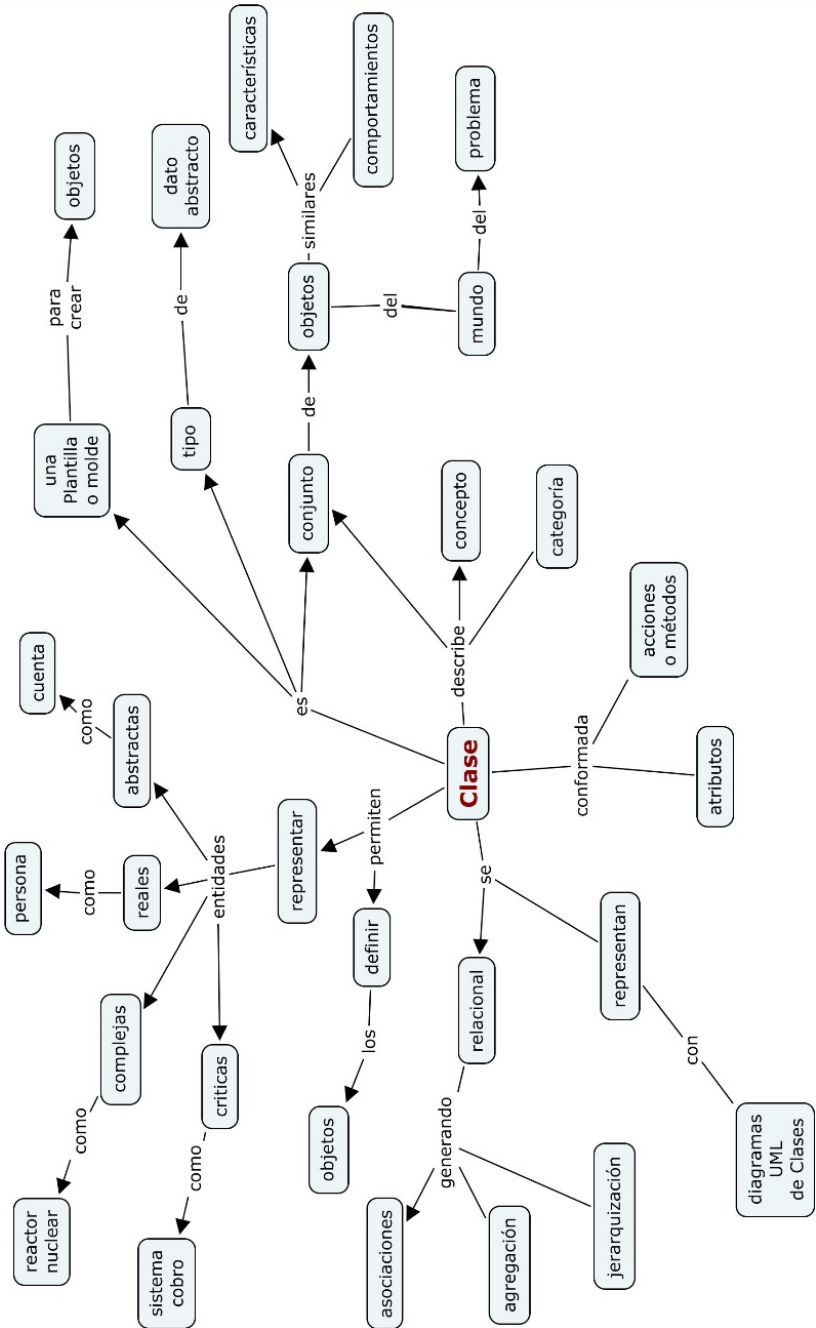
Tabla 1.

Ejemplos de objetos asociados a su correspondiente clase

Objetos	Clase
Mazda 626 sdk23, rojo Ford, capacidad 32 personas	Auto
• Cra. 15 No. 85 – 19 bloque 8 • Calle 128 # 52 – 90 bloque 12 apto. 402.	Vivienda
39.652.123 Claudia Ramírez	Estudiante
• Garfield (Felino) • Snoopy (Perro) • Simba (Rey León) • Willy (Cetáceo)	Mamífero
• Código 1478 Estructura de Datos • Código 18156 Matemáticas discretas	Asignatura

La figura 2 amplía el concepto de clase.

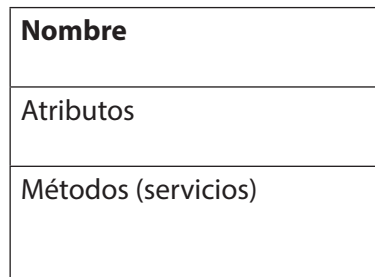
Figura 2.
Mapa mental de clase



1.8.1.1 Representación de las clases

La clase encapsula la información de uno o varios objetos a través de la cual se modela el entorno en estudio. Se representa haciendo uso de los diagramas de clases de UML, conformados por un rectángulo que tiene tres divisiones, como se muestra en la figura 3.

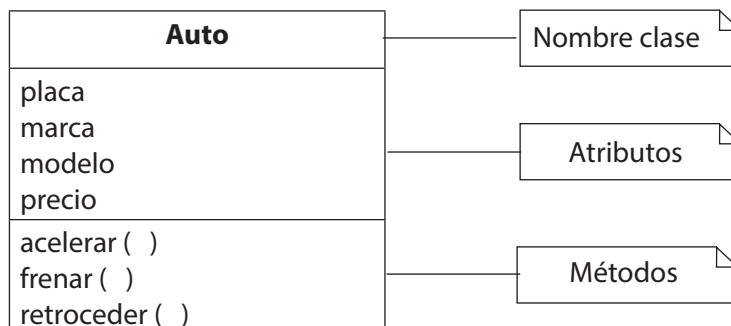
Figura 3.
Partes del diagrama de clase

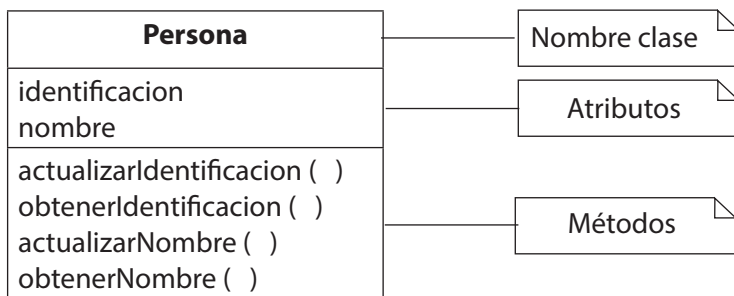
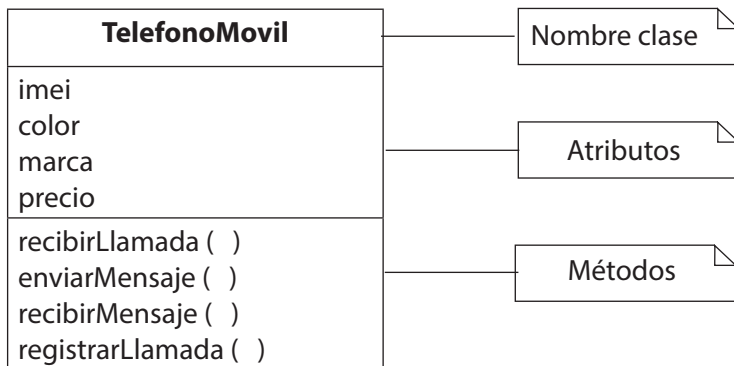
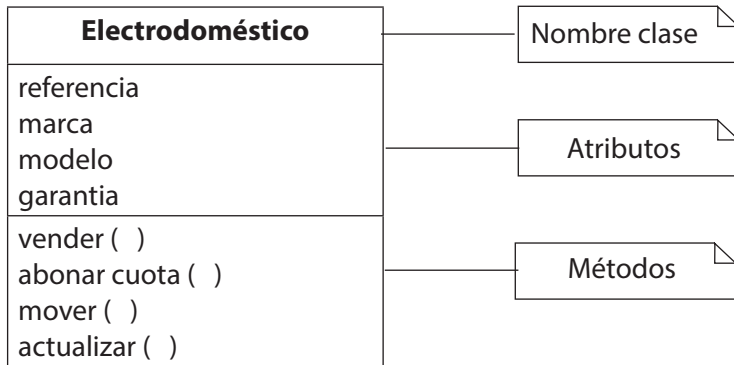


UML (*Unified Modeling Language*) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

 Ejemplo

Los siguientes diagramas de clases asocian la información correspondiente de acuerdo con el número del cuadrante, así, en el primer cuadrante se registra el nombre de la clase, en el segundo cuadrante los nombres de los atributos y en el tercer cuadrante los nombres de los métodos, comportamientos o responsabilidades que realizan.





Los ejemplos anteriores representan diagramas de clases con sus correspondientes atributos y métodos; los rectángulos de los lados unidos con líneas punteadas corresponden a la documentación.

Durante del desarrollo de la temática, incluyendo los capítulos siguientes, se irá ampliando la información correspondiente al tema de las clases.

1.8.2 Objeto

Es cualquier cosa real (tangible o intangible) del mundo que nos rodea; por ejemplo, un auto, una casa, un árbol, un reloj, una estrella, una cuenta de ahorro. Un objeto se puede considerar como una entidad compleja provista de propiedades o características y de comportamientos o estados. La figura 4 (véase en pág. siguiente) detalla el concepto de objeto de una manera más somera.

A un objeto lo identifican las siguientes características:

Estado

Conformado por el conjunto de propiedades o atributos de un objeto correspondiente a los valores que pueden tomar cada uno de esos atributos.

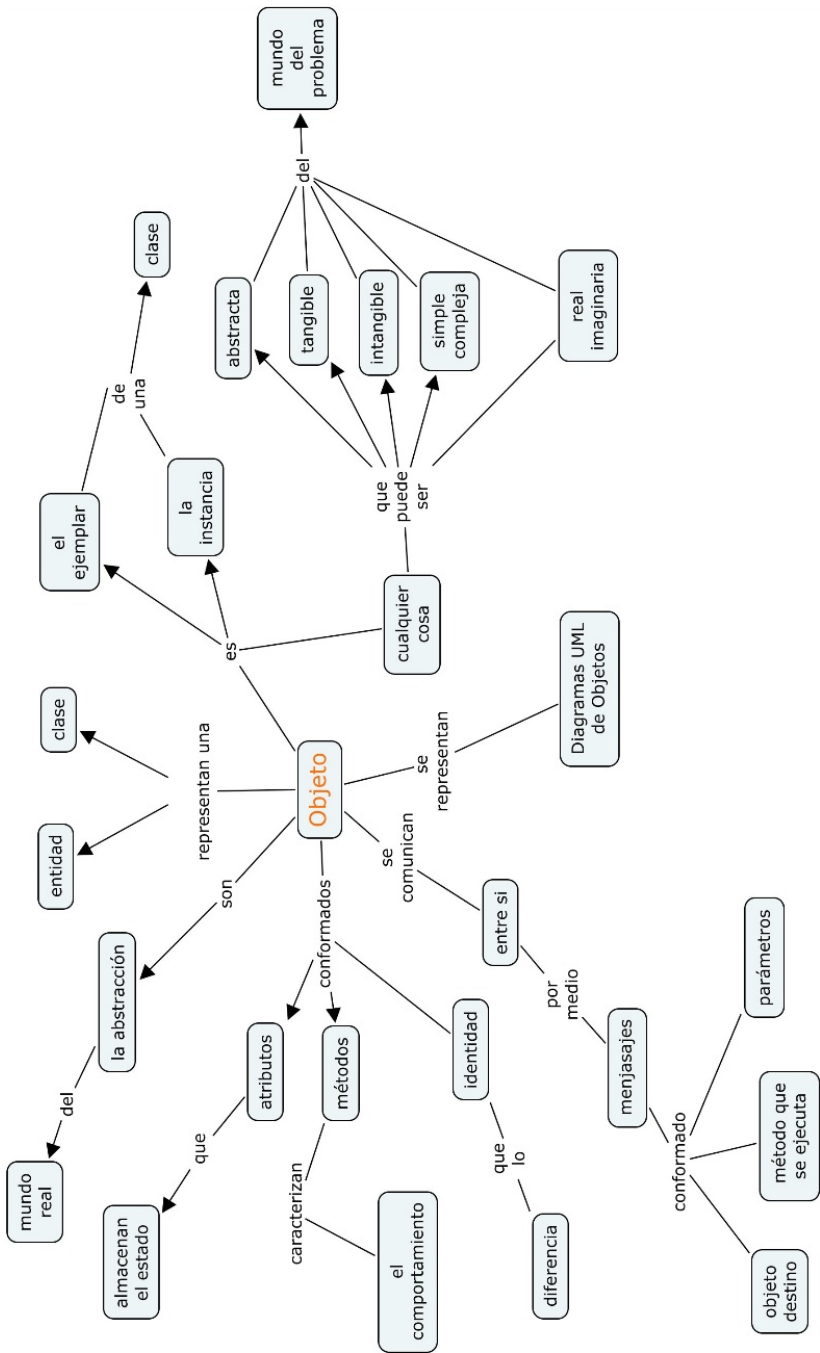


El objeto cliente tiene las siguientes características: documento, nombres, apellidos, dirección y teléfono.

Tabla 2.
Ejemplo de objeto

Atributo	Estado
Documento:	10.265.254
Nombres:	Luis Eduardo
Apellidos:	Baquero Rey
Teléfono:	320 256 32 18

Figura 4.
Mapa mental de objeto



Comportamiento

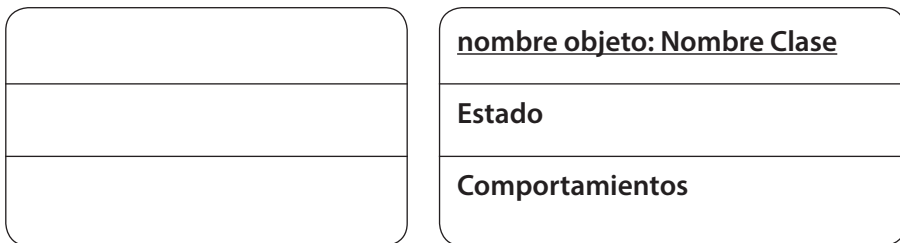
De un objeto, está relacionado con la funcionalidad y determina las operaciones que este puede realizar, responder o ejecutar alguna acción ante mensajes enviados por otros objetos, por ejemplo, actualizar el teléfono de un cliente, realizar la suma de dos números.

Identidad

Corresponde a la propiedad de un objeto que le distingue de todos los demás, como es el caso del ejemplo del cliente Luis Eduardo Baquero Rey, que tiene un número de documento que lo identifica, una dirección y un teléfono.

1.8.2.1 Representación gráfica de los objetos

Los objetos se pueden representar con diagramas UML de objeto, que es un rectángulo con tres divisiones.



Se pueden confundir los términos clase y objeto; en general, una clase es una representación abstracta de algo, mientras que un objeto es una representación de ese algo conformado por la clase.

Ejemplo

Se tienen cuatro objetos correspondientes a relojes (de pared, arena, digital y de pulsera). La clase Reloj tiene las características que agrupa a esa colección de objetos.