Luis Eduardo Baquero Rey, Miguel Armando Hernández Bejarano.
Buenas prácticas de programación orientada a objetos en Java /
Bogotá: Fundación Universitaria Los Libertadores. Facultad de Ingeniería y Ciencias Básicas, 2017.
350 páginas. ISBN: 978-958-9146-68-2

1. Programación orientada a objetos (Computadores).
2. Java (Lenguaje de programación de computadores). I. Título. II. Autores.

005.117 B222b

Fundación Universitaria Los Libertadores. CRAI-Biblioteca "Hernando Santos Castillo"

COLECCIÓN: Ingeniería

AUTORES
© Luis Eduardo Baquero Rey
© Miguel Hernández Bejarano

Primera edición
Bogotá D.C., junio 2016
ISBN: 978-958-9146-68-2 (impreso)

COORDINACIÓN EDITORIAL
Diego Martínez Cárdenas

DISEÑO Y DIAGRAMACIÓN
Amarillo Magenta Estudio Gráfico

IMPRESIÓN
XXXXXXXXX

TABLA DE CONTENIDO

Prólogo 10

1 **Fundamentos de la Programación Orientada a Objetos 15**

1.1 Temática a desarrollar 17
1.2 Introducción 17
1.3 Programación Orientada a Objetos 18
1.4 Principios de la Programación Orientada a Objetos 18
1.5 Ventajas del uso de la programación orientada a objetos 19
1.6 Desventajas del uso de la programación orientada a objetos 20
1.7 Lenguajes de programación orientada a objetos 20
1.8 Clases y objetos 20
1.9 Visibilidad en atributos y métodos 28
1.10 Paquetes 28
1.11 Lecturas recomendadas 29
1.12 Preguntas revisión de conceptos 30
1.13 Ejercicios 30
1.14 Referencias bibliográficas 31

2 **Componentes de un Programa 32**

2.1 Temática a desarrollar 35
2.2 Introducción 35
2.3 Características de un programa Java 36
2.4 Estructura de un programa en Java 36
2.5 Presentación del programa 39
2.6 Construcción de un programa en Java 40
2.7 Palabras reservadas 41
2.8 Los identificadores 41
2.9 Tipos de datos 42
2.10 Variables 44
2.11 Constantes 44
2.12 Comentarios 45

2.13 Definición de variables 46
2.14 Definición de constantes 46
2.15 Operadores aritméticos 47
2.16 Los separadores 48
2.17 Lecturas recomendadas 49
2.18 Preguntas de revisión de conceptos 49
2.19 Ejercicios 49
2.20 Referencias bibliográficas 50

3 **Herramientas para el desarrollo de la Programación Orientada a Objetos 51**

3.1 Temática a desarrollar 53
3.2 Introducción 53
3.3 Entornos Integrados de Desarrollo (IDE) 54
3.4 Herramientas de modelamiento 58
3.5 Java como lenguaje de programación 59
3.6 Lecturas recomendadas 61
3.7 Preguntas de revisión de conceptos 62
3.8 Ejercicios 62
3.9 Referencias bibliográficas 63

4 **Métodos constructores, set y get 59**

4.1 Temática a desarrollar 67
4.2 Introducción 67
4.3 Métodos 68
4.4 Modificadores de acceso 71
4.5 Métodos set y get 78
4.6 Métodos de instancias 82
4.7 Estructura de un método 83
4.8 Prueba de escritorio 85
4.9 Métodos que no retornan valor 87
4.10 Requerimientos 87

4.11 Un desarrollo completo 89
4.12 Entrada y salida de datos 96
4.13 Entrada y salida estándar 97
4.14 Lecturas recomendadas 108
4.15 Preguntas de revisión de conceptos 108
4.16 Ejercicios 109
4.17 Referencias bibliográficas 110

5 **Condicionales** **111**

5.1 Temática a desarrollar 113
5.2 Introducción 113
5.3 Estructuras de control condicionales 114
5.4 Operadores relacionales 115
5.5 Operadores lógicos 116
5.6 Tipos de condicionales 117
5.7 Selección múltiple 135
5.8 Operador condicional (?) 139
5.9 Preguntas de revisión de conceptos 143
5.10 Lecturas recomendadas 143
5.11 Ejercicios 143
5.12 Referencias bibliográficas 149

6 **Cadenas** **151**

6.1 Temática a desarrollar 153
6.2 Introducción 153
6.3 Clase String 154
6.4 Obtener cadenas desde las primitivas 160
6.5 Obtener primitivas desde las cadenas 161
6.6 Lecturas recomendadas 171
6.7 Preguntas de revisión de conceptos 171
6.8 Ejercicios 171
6.9 Referencias bibliográficas 173

7 Ciclos 175

7.1 Temática a desarrollar 177
7.2 Introducción 177
7.3 Estructuras de control repetitivas o ciclos 178
7.4 Comparación de los ciclos while, do-while, for 194
7.5 Ciclo for each (for mejorado) 196
7.6 Uso de los ciclos repetitivos 197
7.7 Lecturas recomendadas 197
7.8 Preguntas y ejercicios de revisión de conceptos 198
7.9 Ejercicios 198
7.10 Referencias bibliográficas 199

8 Relaciones entre Clases 201

8.1 Temática a desarrollar 203
8.2 Introducción 203
8.3 Elementos entre las relaciones de clases 204
8.4 La relación de Especialización/Generalización 210
8.5 Dependencia 211
8.6 Lecturas recomendadas 212
8.7 Preguntas de evaluación 212
8.8 Ejercicios 213
8.9 Referencias bibliográficas 217

9 Abstracción y Encapsulamiento 219

9.1 Temática a desarrollar 221
9.2 Introducción 221
9.3 Abstracción 221
9.4 Encapsulamiento 223
9.5 Lecturas recomendadas 227
9.6 Preguntas de revisión de conceptos 227
9.7 Ejercicios 227
9.8 Referencias bibliográficas 229

10 **Herencia** **231**

10.1 Temática a desarrollar 233
10.2 Introducción 233
10.3 Herencia 234
10.4 Representación de la herencia 239
10.5 Sentencia super 244
10.6 Lecturas recomendadas 247
10.7 Preguntas de revisión de conceptos 248
10.8 Ejercicios 248
10.9 Referencias bibliográficas 251

11 **Polimorfismo** **253**

11.1 Temática a desarrollar 255
11.2 Introducción 255
11.3 Polimorfismo 256
11.4 Sobrecarga de métodos 257
11.5 Sobrecarga de constructores 260
11.6 Sobreescritura de métodos 262
11.7 Enlace dinámico 269
11.8 Lecturas recomendadas 272
11.9 Preguntas de revisión de conceptos 272
11.10 Ejercicios 272
11.11 Referencias bibliográficas 275

12 **Interface y Clases Abstractas** **277**

12.1 Temática a desarrollar 279
12.2 Introducción 279
12.3 Interface 280
12.4 Clases abstractas 284
12.5 Preguntas de revisión de conceptos 286
12.6 Ejercicios 286
12.7 Referencias bibliográficas 290

13 Modelado Orientado a Objetos y Aplicaciones de Software 291

13.1 Temática a desarrollar 293
13.2 Introducción 293
13.3 Análisis orientado a objetos 294
13.4 Escenarios 295
13.5 Prototipo 295
13.6 Diagramas de casos de uso 296
13.7 Diseño orientado a objetos 305
13.8 Diagramas UML (Unified Modeling Language) 309
13.9 Aplicaciones de software 311
13.10 Lecturas recomendadas 331
13.11 Preguntas de revisión de conceptos 331
13.12 Ejercicios 331
13.14 Referencias bibliográficas 333

14 Documentación 335

14.1 Temática a desarrollar 337
14.2 Introducción 337
14.3 Javadoc 337
14.4 Javadoc en los IDE 343
14.5 API de Java 346
14.6 Lecturas recomendadas 347
14.7 Preguntas de revisión de conceptos 348
14.8 Ejercicios 348
14.9 Referencias bibliográficas 348

LISTA DE FIGURAS

Figura 1. Concepto de Programación Orientada a Objetos

Figura 2. Concepto de Clase

Figura 3. Partes del diagrama de clase

Figura 4. Concepto de objeto

Figura 5. Ejemplos del objeto reloj

Figura 6. Diagrama de paquete

Figura 7. Diagrama de paquetes con dos clases comunes.

Figura 8. Concepto de tipos de datos.

Figura 9. Pantalla de carga de Eclipse LUNA

Figura 10. Pantalla inicio en Netbeans

Figura 11. Pantalla inicio de GreeFoot

Figura 12. Pantalla IntelliJ IDEA

Figura 13. Concepto de Java

Figura 14. Concepto de plataformas de java

Figura 15. Concepto de Métodos

Figura 16. Concepto de Constructor

Figura 17. Concepto de métodos set y get

Figura 18. Concepto de las estructuras condicionales

Figura 19. Concepto de la clase String

Figura 20. Concepto de los ciclos

Figura 21. Concepto de relaciones en clases

Figura 22. Concepto de abstracción

Figura 23. Concepto de encapsulamiento

Figura 24. Concepto de Herencia

Figura 25. Concepto de polimorfismo

Figura 26. Concepto de aplicaciones de software

Figura 27. Documentacion en Bluej

Figura 28. Generacion Javadoc en eclipse

Figura 29. Documentación Javadoc en eclipse

Figura 30. Generacion Javadoc en Netbeans

Figura 31. Documentación Javadoc en Netbeans

Figura 32. API Java 8

LISTA DE TABLAS

Tabla 1. Ejemplos de objetos asociados a su correspondiente clase

Tabla 2. Ejemplo de objeto

Tabla 3. Ejemplo estructura interna de un Objeto

Tabla 4. Palabras reservadas en java

Tabla 5. Tipos de datos de java

Tabla 6. Operadores aritméticos

Tabla 7. Operadores relacionales

Tabla 8. Operadores lógicos

Tabla 9. Resultado operadores lógicos

Tabla 10. Apoyo implementación del método

Tabla 11. Apoyo parte de análisis

Tabla 12. Ejemplo de cadenas

Tabla 13. Metodos básicos

Tabla 14. Tipos de cardinalidad

Tabla 15. Comparación entre abstracción y encapsulamiento

PRÓLOGO

En el quehacer diario de la docencia, y más concretamente en el área de la programación, surgen muchos temas, talleres y ejercicios a 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 Concepto de, 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 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.

El **capítulo 7** 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** el mecanismo de la programación orientada a objetos denominado polimorfismo, incluyendo, polimorfismo de subtipado, sobrecarga de métodos y sobreescritura de métodos.

En el **capítulo 12** se continúa con el concepto y manejo de interface 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 8.

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

1.1 TEMÁTICA A DESARROLLAR

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

1.2 INTRODUCCIÓN

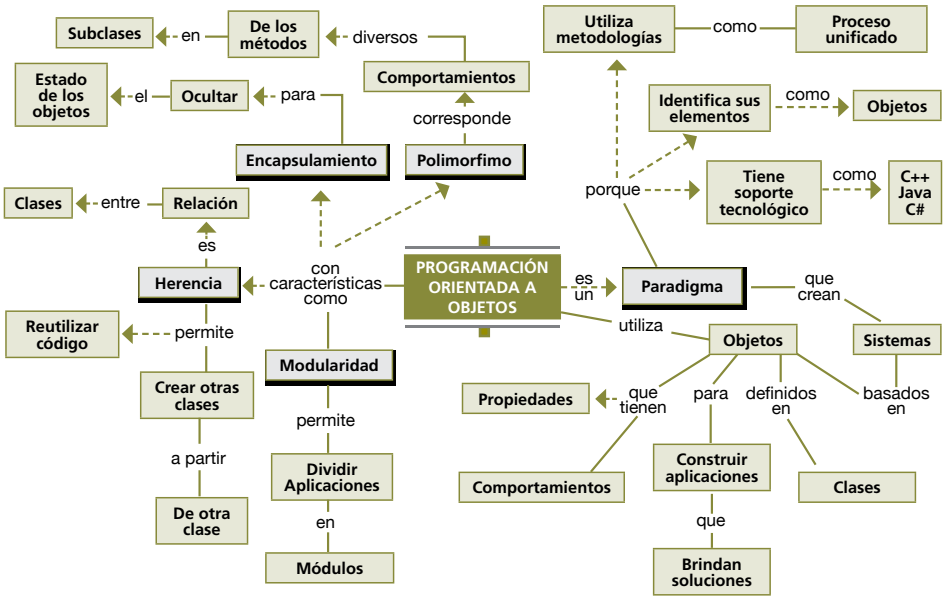
La concepción del mundo en términos de Programación Orientada a Objetos (POO) está integrada por un conjunto de entidades u objetos relacionados entre sí, estos pueden ser personas, árboles, automóviles, casas, teléfonos y computadores; en general, todo elemento que tiene atributos y que pueden ser palpables 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), acciones (métodos). Una clase describe un conjunto de objetos con propiedades y comportamientos similares. La figura 1 muestra, mediante mapa mental este concepto y todo lo que lo rodea.

El buen uso de este paradigma de programación se convierte en una buena práctica que arrojaría como resultados la construcción de programas de calidad, facilita 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.

Figura 1. Concepto de Programación Orientada a Objetos.



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 debe tener la base y la altura; o cuando se ve televisión no hay preocupación del modo como este funciona, lo único que se realiza es manipular el control y disfrutar del programa, película, entre otros ejemplos.

Herencia: 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 siguiente 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, denominados 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 otras.

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 proceso 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 a 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.

Ejemplos

» 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.

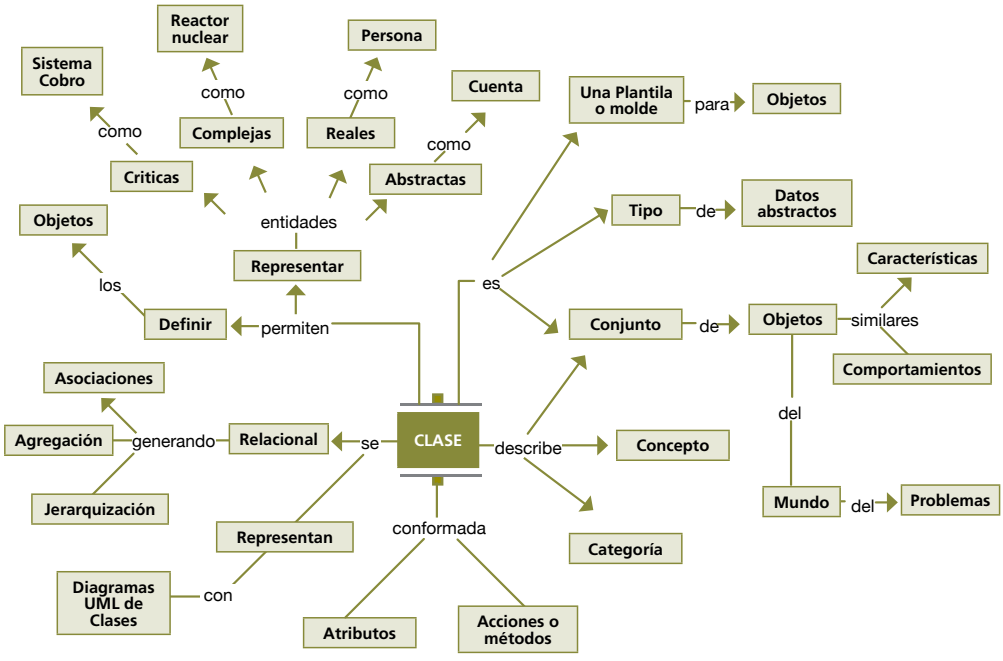
Ejemplos

Tabla 1. Ejemplos de objetos asociados a su correspondiente clase

OBJETOS	CLASE
Mazda 626 sdk23, rojo Ford, capacidad 32 persona	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.

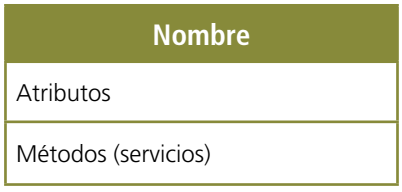
Figura 2. Concepto de Programación Orientada a Objetos.



1.8.1.1 Representación de las clases

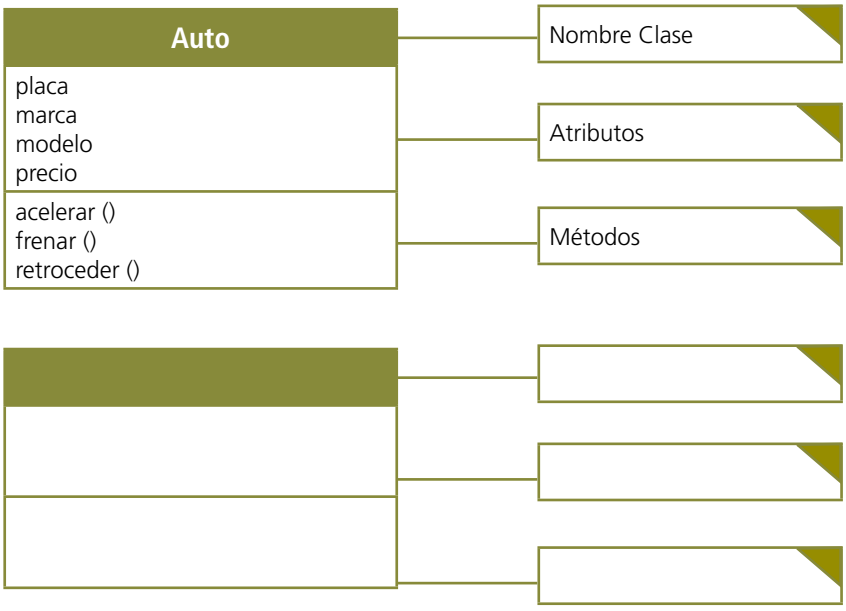
La clase encapsula la información de uno o varios objetos a través de la cual se modela el entorno en estudio. Se representa haciendo uso de los diagramas de clases de UML, conformado por un rectángulo que tiene tres divisiones, como se muestra en la figura 3.

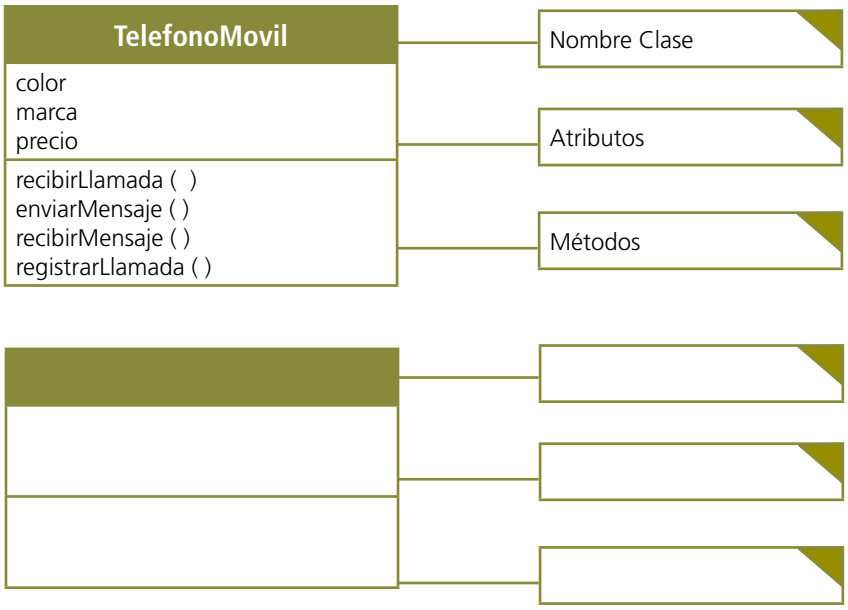
Figura 3. Partes del diagrama.



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

Ejemplos: los siguientes diagramas de clases se 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.





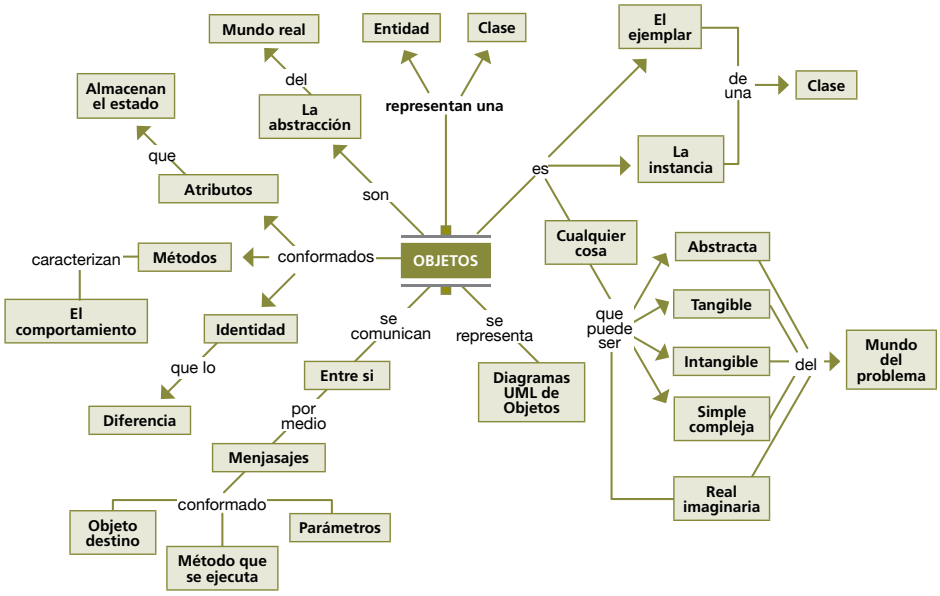
Los ejemplos anteriores reprentan diagramas de clases con sus correspondientes atributos y metodos, los rectángulos de los lados unidos con líneas punteadas corresponden a la documentación.

Durante del desarrollo de la temática incluyendo los capítulos siguientes, se va 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, cuenta de ahorro. Uno objeto se puede considerar como una entidad compleja provista de propiedades o características y de comportamientos o estados. La figura 4 detalla el concepto de objeto de una manera más somera.

Figura 4. Concepto de Objeto.



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.

Ejemplo:

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:	300 000 00 00

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.

	Nombre objeto : Nombre Clase
	Estado
	Comportamientos

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

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

Figura 5. Ejemplos del objeto reloj.

A continuación se modela la clase Reloj y un objeto reloj; mediante los diagramas correspondientes.

Reloj
hora minuto
reloj ( ) fijarHora ( ) adelantarHora ( )

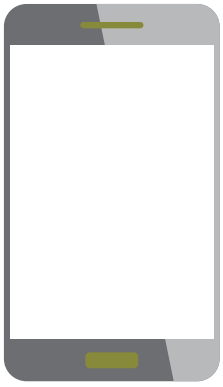
miRejoj : Reloj
hora = 3 minuto= 49

Para el ejemplo, un objeto miReloj es una entidad que tiene un conjunto de propiedades o atributos, como el color, marca, forma, estilo, precio y dentro de los comportamientos están el dar la horas, minutos, adelantarse, atrasarse.

Se tiene dos ejemplos donde se modelan los objetos (computador portátil y teléfono celular) mediante el correspondiente diagrama de objetos.

portatil : Computador
color = "negro" marca = "Lenovo" precio=1.800.000
prender ( ) apagar ( ) procesar ( ) almacenar ( )

celular : TelefonoMovil
color = "negro" marca = "movil" precio= 180.000
recibirLlamada ( ) enviarMensaje ( ) recibirMensaje ( ) registraLlamadas ( )



Otros ejemplos de objetos son:

Objetos físicos

» Barcos, aviones, perros, computadores.

Objetos de interfaces gráficos de usuarios:

» Etiquetas, cajas de texto, botones

Objetos intangibles:

» Cuenta de ahorros, cuenta corriente, CDT

Ejemplos de no objetos:

» El amor, la felicidad, la nostalgia.

1.8.2.2 Estructura interna de un objeto

Los atributos de una clase definen el estado del objeto, que son los valores de los datos del objeto concreto y que lo diferencian de los demás.

Ejemplos:

Tabla 3. Ejemplo estructura interna de un Objeto.

OBJETO	ESTADO
estudiante	edad = 20 estatura 1.80
celular	color = negro marca = Nokia
computador	marca = Lenovo color = negro

Los métodos de una clase, definen el comportamiento del objeto y corresponde a las operaciones que pueden realizarse sobre los objetos de una clase.

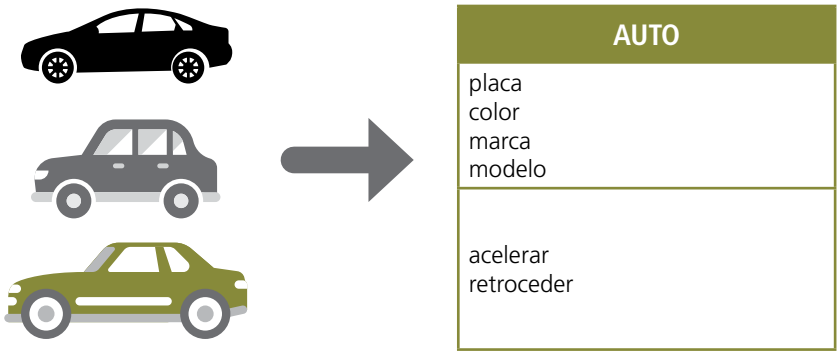
1.9 VISIBILIDAD EN ATRIBUTOS Y MÉTODOS

La Programación Orientada a Objetos ayuda a incrementar factores de calidad como: reutilización, facilidad del uso de las clases, entre otros. Por su parte, la comunicación y visibilidad tanto para los atributos de una clase como para los métodos puede ser: public, private, protected (público, privado, protegido).

Público (public). Representado por el signo más (+) indica que el método o atributo está visible dentro o fuera de la clase, es decir, es accesible desde todos lados.

Privado (private). Representado por el signo menos (-) indica que el atributo o método solamente será accesible dentro de la misma clase (sólo sus métodos lo pueden acceder).

Protegido (protected). Representado por el símbolo numeral (#) indica que el atributo o método no es accesible desde fuera de la clase, pero se podrá acceder por métodos de la clase, además de las subclases que se deriven como es el caso de la herencia.

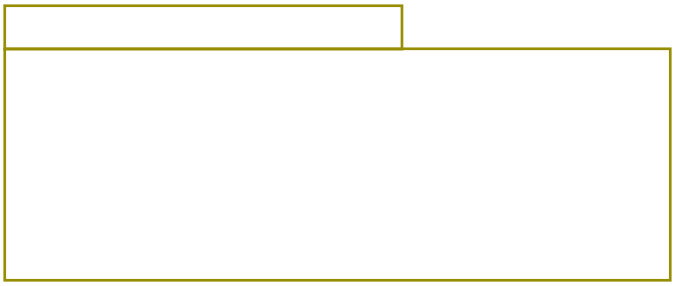


Los tres vehículos son objetos que se pueden agrupar en la clase Auto.

1.10 PAQUETES

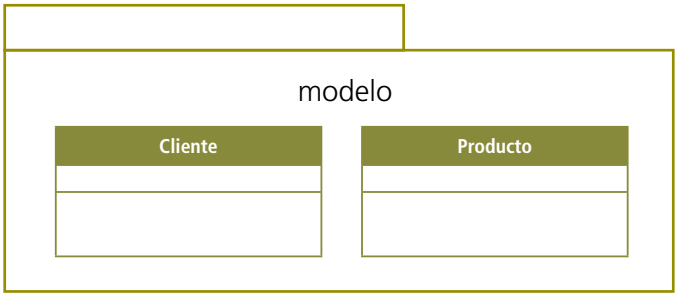
Permiten agrupar los elementos de comportamientos similares que se modelan con UML, La figura 6 representa un diagrama de paquete, similar a una carpeta o fólder.

Figura 6. Diagrama de paquete.



La figura 7 ilustra un diagrama de paquetes, conformado por dos clases comunes al modelo de un sistema.

Figura 7. Diagrama de paquetes con dos clases comunes.



1.11 LECTURAS RECOMENDADAS

- Qué es UML.
- Historia de UML.
- Paradigma de Programación Orientada a Objetos.
- Atributos, métodos y constructores.
- Principios de la Programación Orientada a Objetos.

1.12 PREGUNTAS REVISIÓN DE CONCEPTOS

» ¿Dónde utilizar UML?

» Diferencias entre clases y objetos.

» ¿Cuáles son los lenguajes que permiten la implementación del concepto de objetos y actualmente son fuente para el desarrollo de proyectos de software?

1.13 EJERCICIOS

1. Construir la clase llamada Proveedor, que está conformada por siguientes atributos:

- » NIT.
- » Nombre de la empresa o razón social.
- » Nombre del representante.
- » Primer apellido del representante.
- » Segundo apellido del representante.
- » Dirección empresa.
- » Teléfono.
- » E-mail.

---

2. Crear una clase denominada Cubo, conformada por los siguientes atributos: ancho, alto y largo.

3. Completar la información de la siguiente tabla.

CLASE	OBJETOS
Ropa	camisa El monarca, talla 38; pantalón negro, talla 32
Mamífero	

4. Completar la información de la siguiente tabla:

CLASE	OBJETOS
Mueble	
Gato	
Docente	

1.14 REFERENCIAS BIBLIOGRÁFICAS

Aguilar, L. J. (2004). *Fundamentos de programación algoritmos y estructura de datos.* Tercera Edición. Ciudad: México: McGraw Hill.

Booch, G. (1996). *Análisis y diseño orientado a objetos con aplicaciones.* Ciudad: México Addison Wesley.

Deitel y Deitel. (2012). *Cómo programar en Java.* Ciudad: México Editorial Pearson (Prentice Hall).

Fowler, M., y Scott, K. UML. (1999). *Gota a gota.* Ciudad: México Editorial Pearson.

Villalobos, J., y Casallas, R. (2006). *Fundamentos de programación, aprendizaje activo basado en casos.* Ciudad: México Editorial Pearson.