

## 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

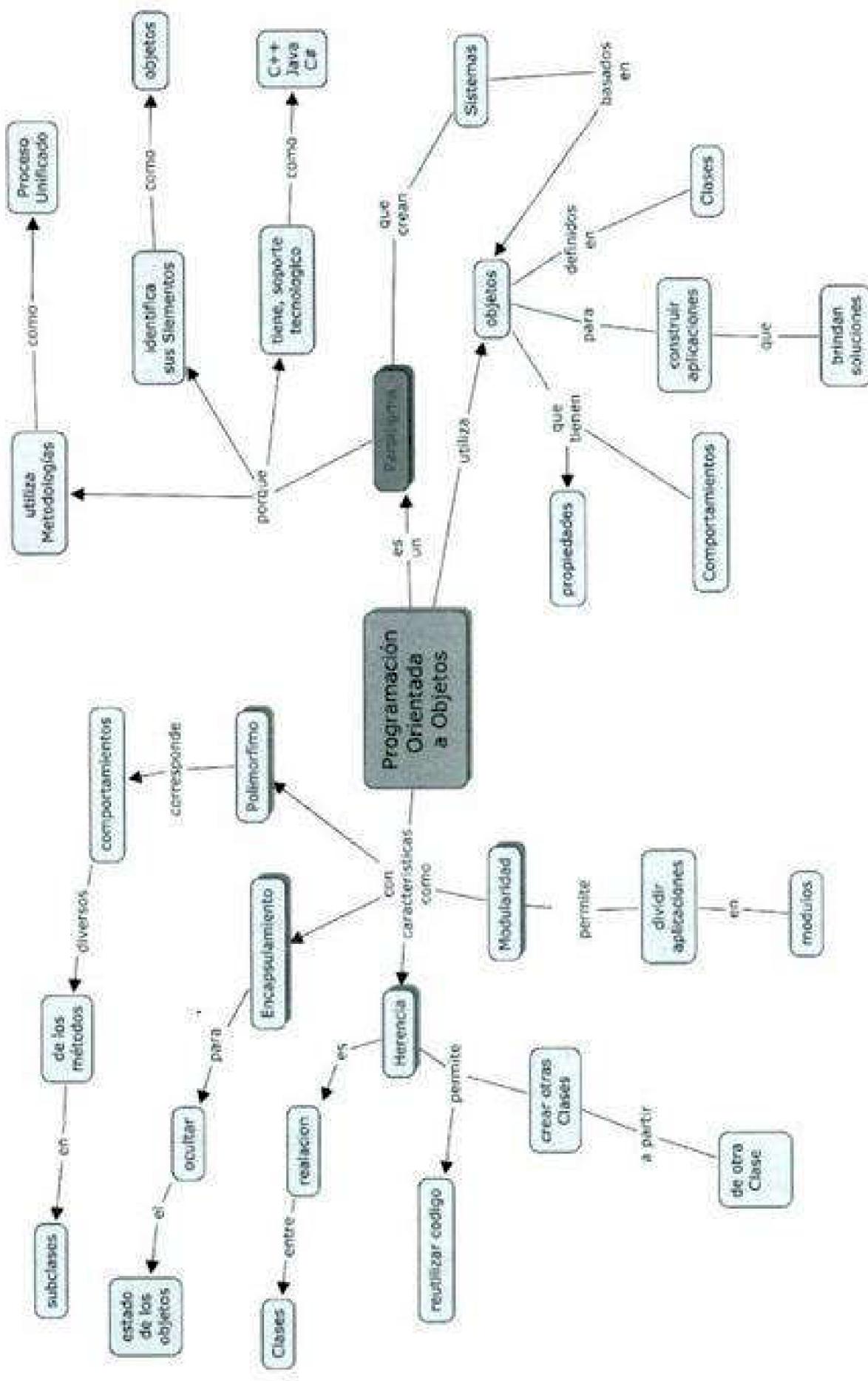
### 1.2 Introducción

La concepción del mundo en términos de Programación Orientada a Objetos (POO) está integrada por un conjunto de entidades u objetos relacionados entre si, 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 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.

**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.

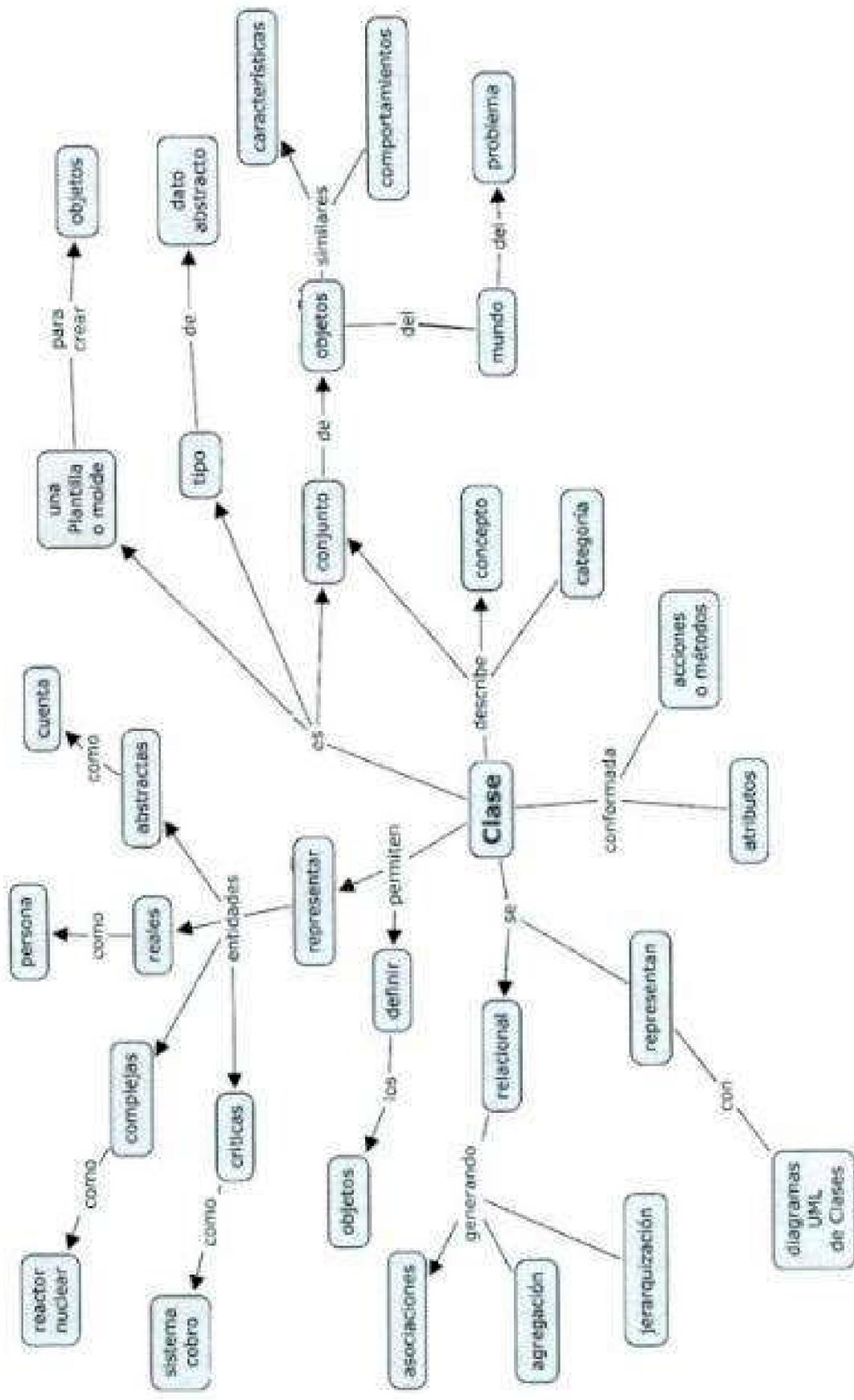
**Ejemplo****Tabla 1.**

*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 • Garfield (Felino) • Snoopy (Perro) • Simba (Rey León) • Willy (Cetáceo)	Estudiante Mamífero
• Código 1478 Estructura de Datos • Código 18156 Matemáticas discretas	Asignatura

La figura 2 amplía el concepto de clase.

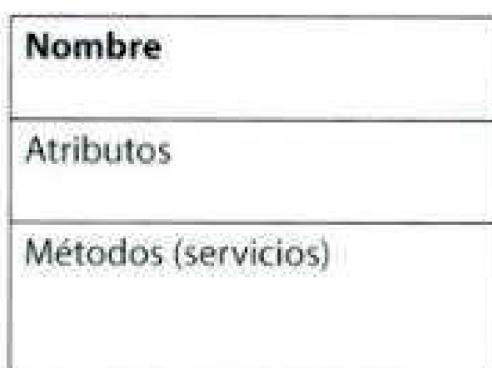
**Figura 2.**  
**Mapa mental de clase**



### 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.

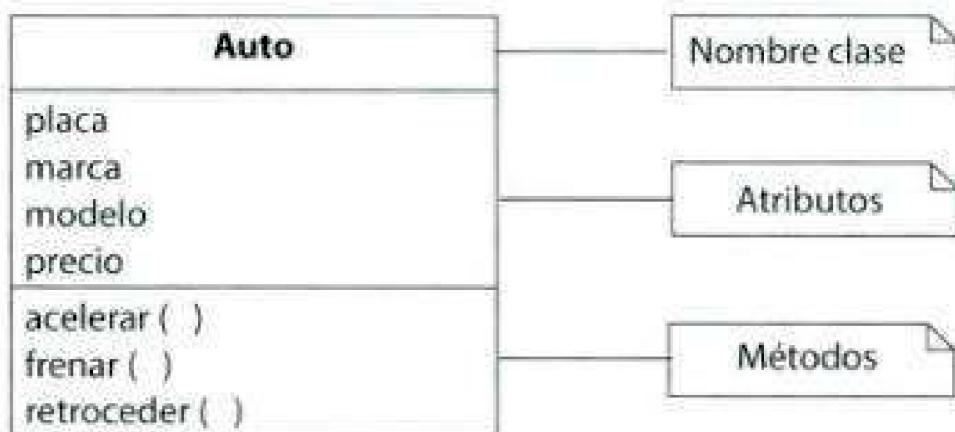
**Figura 3.**  
*Partes del diagrama de clase*

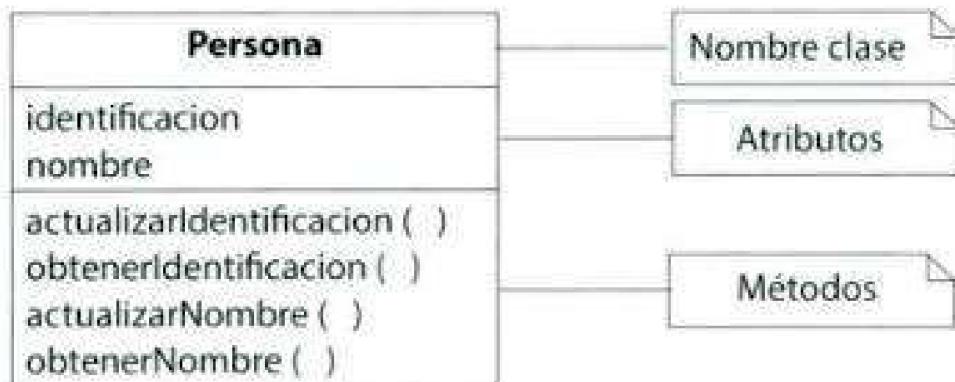
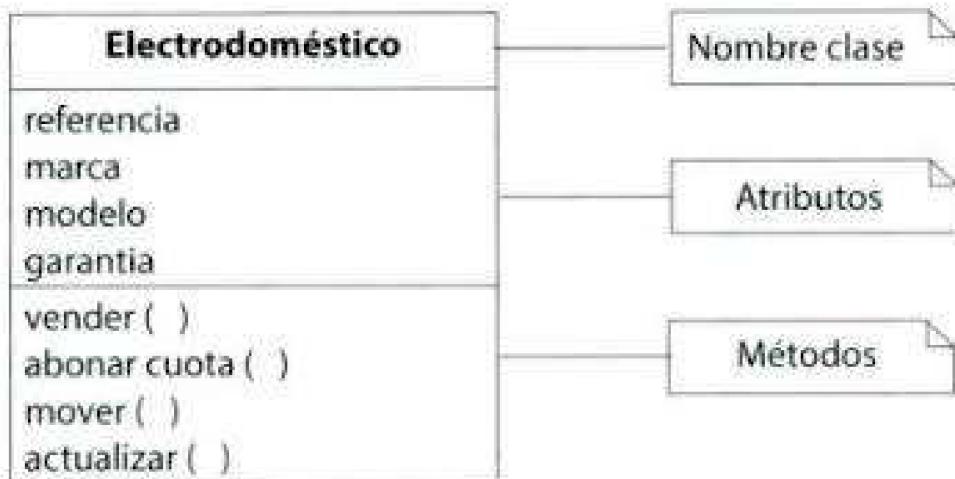


UML (*Unified Modeling Language*) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

**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.



#### Ejemplo

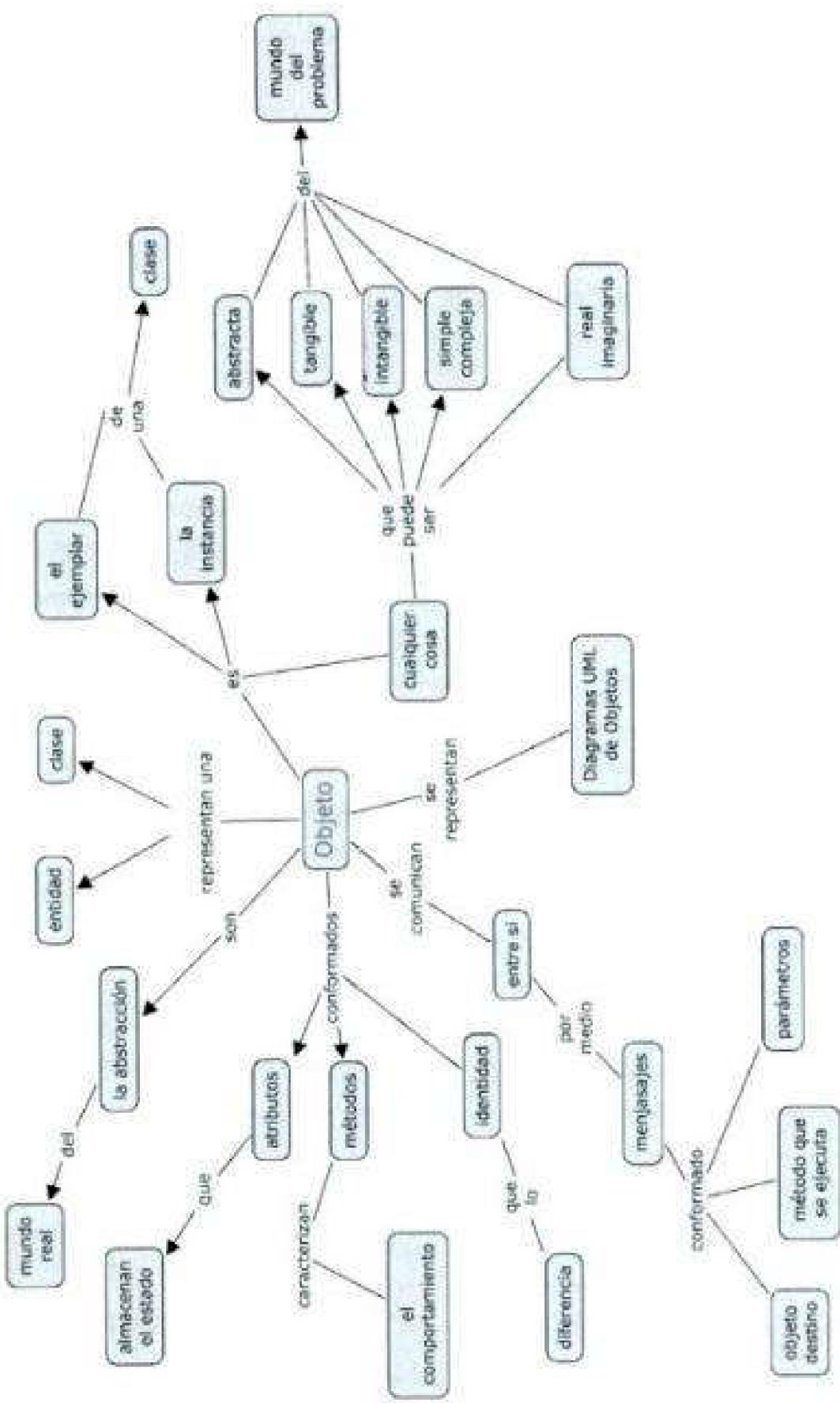
El objeto cliente tiene las siguientes características: documento, nombres, apellidos, dirección y teléfono.

#### Tabla 2.

*Ejemplo de objeto*

Atributo	Estado
Documento:	10.265.254
Nombres:	Luis Eduardo
Apellidos:	Baquero Rey
Teléfono:	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.



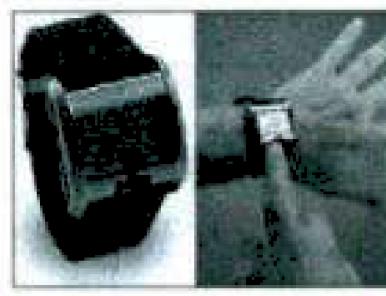
Se pueden confundir los términos clase y objeto; en general, una clase es una representación abstracta de algo, mientras que un objeto es una representación de ese algo conformado por la clase.



#### 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.

**Figura 5.**  
*Ejemplos del objeto reloj*



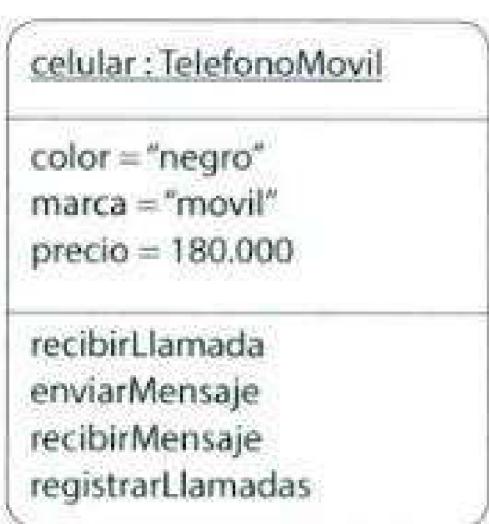
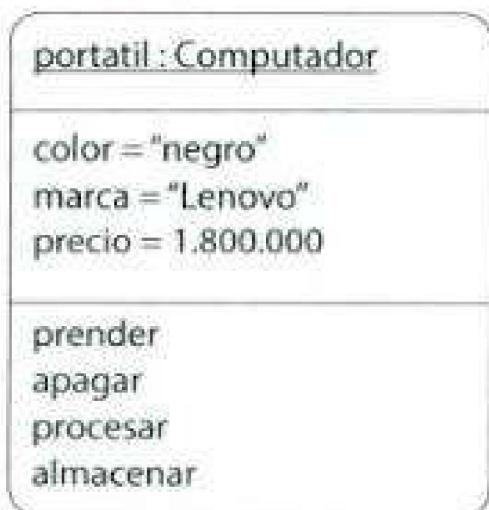
A continuación, se modela la clase Reloj y un objeto reloj mediante los diagramas correspondientes.

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

miReloj : 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 las horas, minutos, adelantarse, atrasarse.

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



Otros ejemplos de objetos son:

**Objetos físicos**

- Barcos, aviones, perros, computadores.

**Objetos de interfaces gráficas 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.



#### Ejemplo

**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.

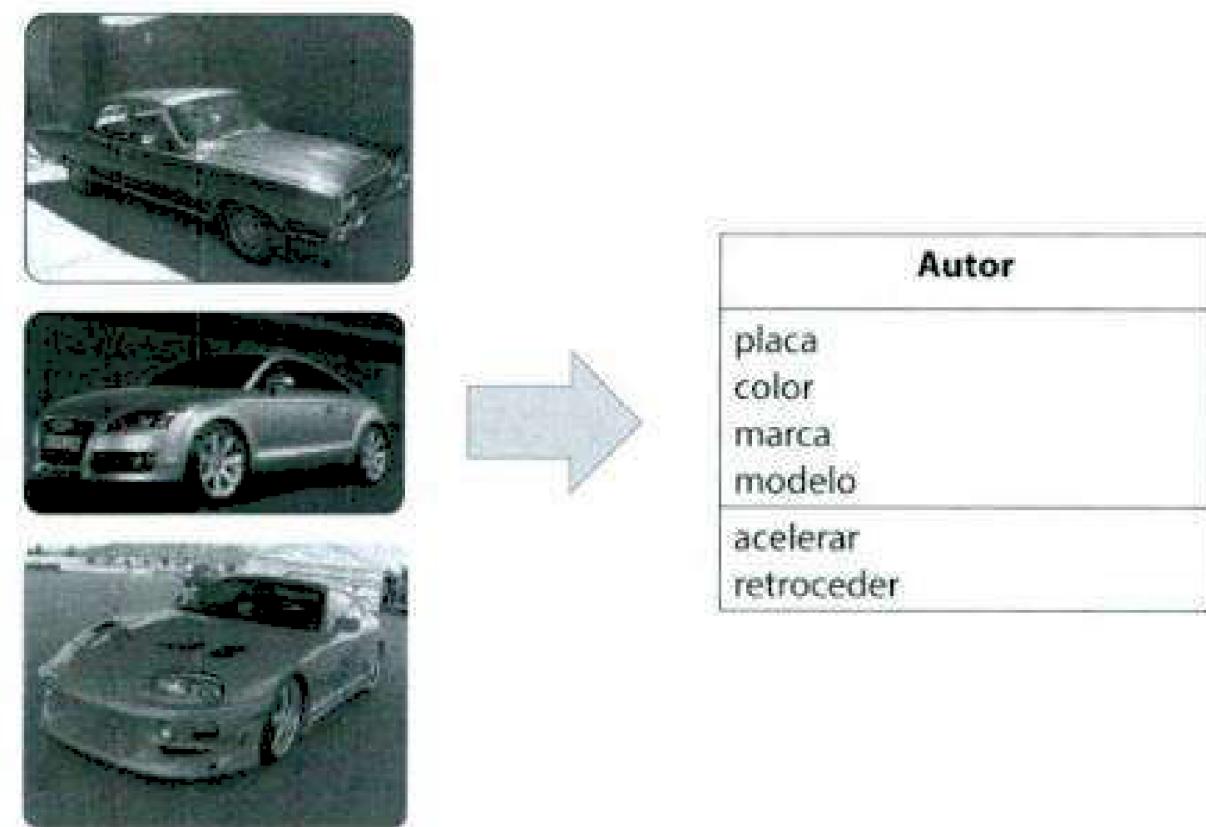
## 1.9 Visibilidad en atributos y métodos

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 los lados.

**Privado (private)**. Representado por el signo menos (-), indica que el atributo o método solamente será accesible dentro de la misma clase (solo 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.

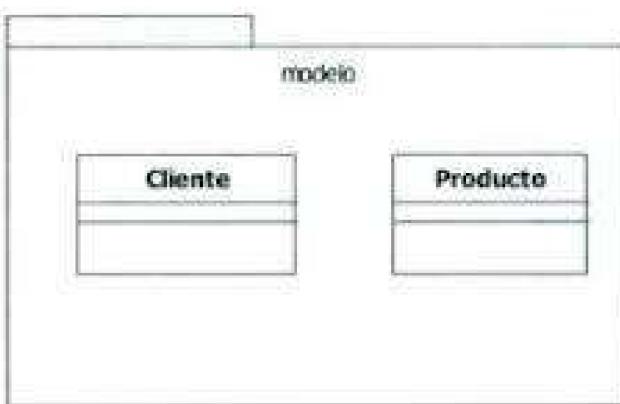
## 1.10 Paquetes

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 *folder*.

**Figura 6.**  
*Diagrama de paquete*



**Figura 7.**  
*Diagrama de paquetes con dos clases comunes*



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

## 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

1. ¿Dónde utilizar UML?
2. Diferencias entre clases y objetos.
3. ¿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 los siguientes atributos:

- NIT
- Nombre de la empresa o razón social
- Nombre del proveedor
- Primer apellido del proveedor
- Segundo apellido del proveedor
- 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 Bibliografía

- Aguilar, L. J. (2004). *Fundamentos de programación de 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. (1999). *UML 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.