Contenido

Conceptos de la programación orientada a objetos

Si nunca utilizaste un lenguaje orientado a objetos antes, necesitás entender los conceptos subyacentes antes de comenzar a escribir código. Necesitás conocer qué es un objeto, qué es una clase, cómo se relacionan clases y objetos, y cómo se comunican utilizando mensajes. Las primeras pocas secciones de este capítulo describen los conceptos detrás de la programación orientada a objetos (desde ahora, POO).

¿Qué es un objeto?

Un objeto es una unidad de software que contiene variables y métodos relacionados. Los objetos software son frecuentemente usados para modelar objetos del mundo real que encontrás día a día en la vida cotidiana.

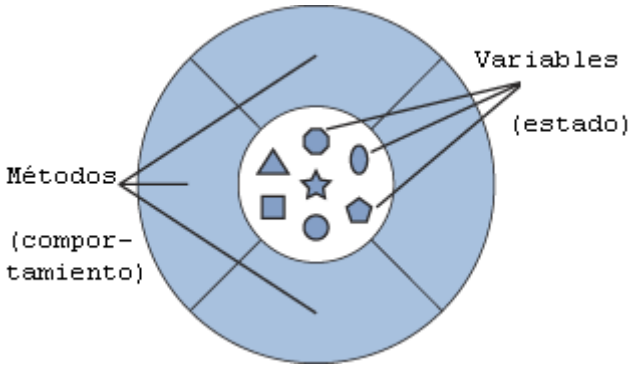
Los objetos son la clave para entender la tecnología orientada a objetos. Podés echar un vistazo ahora mismo y ver muchos ejemplos de objetos del mundo real: tu perro, tu mesa, tu televisor, tu bicicleta.

Los objetos del mundo real comparten dos características: Todos tienen un estado y un comportamiento. Por ejemplo, los perros tienen estado (nombre, color de pelo, raza, si están hambrientos...) y comportamiento (ladrar, buscar, mover la cola). Las bicicletas tienen estado (engranaje actual, cadencia de pedal actual, dos ruedas, número de engranajes) y comportamiento (frenar, acelerar, reducir velocidad, cambiar engranajes).

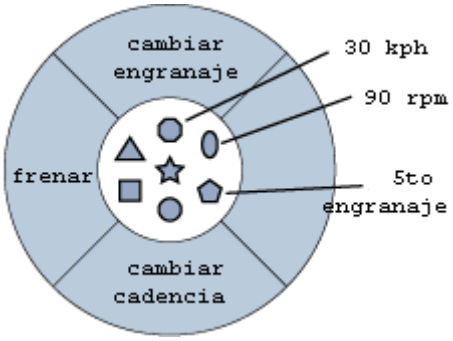
Los objetos software son modelados en base a objetos del mundo real y por ello también tienen estado y comportamiento. Un objeto software mantiene su estado en una o más variables. Una variable es un ítem de dato nombrado e identificado. Un objeto software implementa su comportamiento con métodos. Un método es una función (subrutina) asociada con un objeto.

Definición: Un objeto es una construcción software de variables y métodos relacionados.

Podés representar objetos del mundo real utilizando objetos software. Podrías querer representar perros del mundo real como objetos software en un programa de animación o una bicicleta del mundo real como un objeto software en el programa que controla una bicicleta electrónica para ejercicios. También podés usar objetos software para modelar conceptos abstractos. Por ejemplo, un evento es un objeto común usado en los sistemas de ventanas para representar la acción de un usuario presionando un botón del mouse o una tecla del teclado. La siguiente ilustración es una representación visual común de un objeto software.



Todo lo que el objeto software conoce (estado) y puede hacer (comportamiento) es expresado por las variables y los métodos dentro de ese objeto. Un objeto software que modela tu bicicleta del mundo real tendría variables que indicaran el estado actual de la bicicleta: Su velocidad es de 30 kph, su cadencia de pedal es 90 rpm, y el engranaje actual es el quinto. Esas variables son formalmente conocidas como variables de instancia, porque contienen el estado de un objeto bicicleta en particular; en la terminología orientada a objetos, un objeto en particular es denominado instancia. La siguiente figura ilustra una bicicleta modelada como un objeto software:



En suma a sus variables, la bicicleta software debería tener además métodos para frenar, cambiar la cadencia de pedal y cambiar los engranajes (No debería tener un método para cambiar la velocidad porque la velocidad de la bicicleta es solo un efecto secundario de qué engranaje está activo y qué tan rápido está pedaleando el ciclista). Estos métodos son conocidos formalmente como métodos de instancia, porque evalúan o cambian el estado de una instancia particular de objeto bicicleta.

Los diagramas de objetos muestran que las variables de un objeto están en el centro o núcleo del objeto. Los métodos rodean y ocultan el núcleo de otros objetos en el programa. El empaquetado de las variables de un objeto con esa custodia protectora de sus métodos es llamado encapsulamiento. Esta imagen conceptual de un objeto (un núcleo de variables empaquetadas dentro de una membrana protectora de métodos) en una representación ideal de un objeto y es el ideal que los diseñadores de sistemas orientados a objetos buscan. Sin embargo, la historia no termina aquí.

Algunas veces, por razones prácticas, un objeto debe exponer algunas de sus variables o esconder algunos de sus métodos. En el lenguaje de programación Java, un objeto puede especificar uno de cuatro niveles de acceso para cada una de sus variables o métodos. El nivel de acceso determina qué otros objetos y clases pueden acceder a esa variable o método. Remitite a la sección Controlando el Acceso a los Miembros de una Clase, para más detalles.

Encapsular variables relacionadas y métodos en una única y ordenada construcción de software es una simple pero poderosa idea que provee dos beneficios primarios para los desarrolladores de software:

- Modularidad: El código fuente para un objeto puede ser escrito y mantenido independientemente del código fuente para los otros objetos. Además, un objeto puede fácilmente circular por el sistema. Podés darle tu objeto bicicleta a otra persona, y aún funcionará.

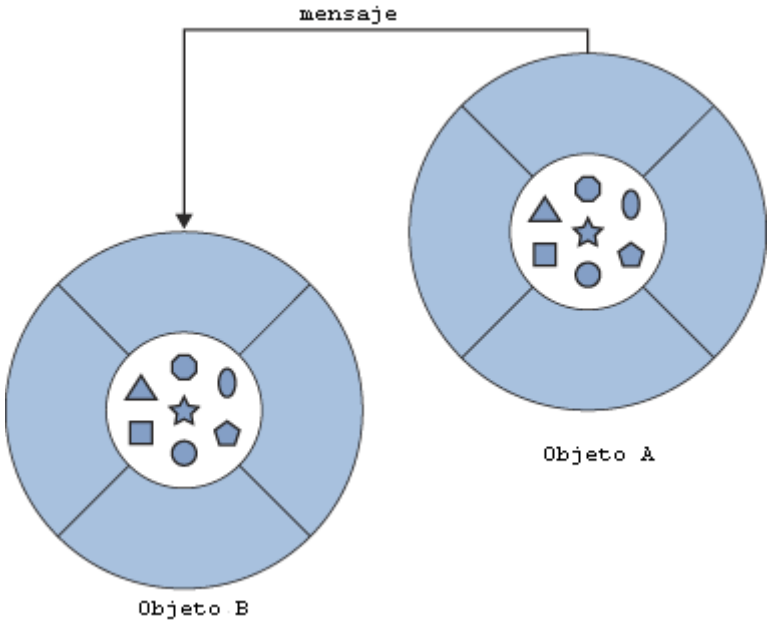
- Ocultamiento de la información: Un objeto tiene una interfase pública que los otros objetos pueden utilizar para comunicarse con él. El objeto puede mantener información privada y métodos que pueden ser cambiados en cualquier momento sin afectar a otros objetos que dependan de ello. No necesitás entender el mecanismo de los engranajes de las bicicletas para utilizar una.

¿Qué es un mensaje?

Los objetos software interactúan y se comunican con otros utilizando mensajes

Un simple objeto sólo generalmente no es muy útil. En cambio, un objeto usualmente aparece como un componente de un programa o aplicación más grande que contiene muchos otros objetos. A través de la interacción de estos objetos, los programadores consiguen funcionalidades de orden superior y un comportamiento más complejo. Tu bicicleta colgada de un gancho en el garaje es sólo una pieza de metal y coma; por sí misma es incapaz de cualquier actividad; la bicicleta es útil sólo cuando otro objeto (vos) interactúa con ella (pedaleando).

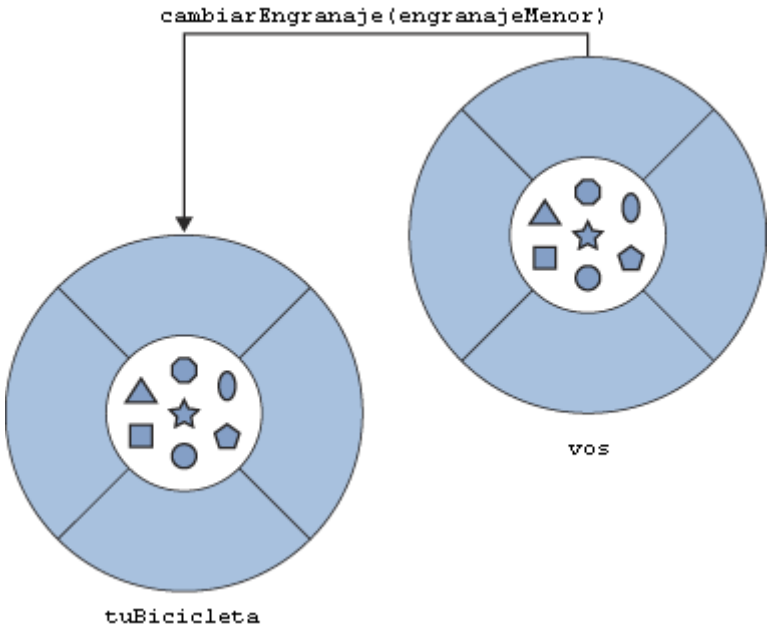
Los objetos software interactúan y se comunican con otros enviándose mensajes. Cuando el objeto A quiere que el objeto B haga uno de los métodos de B, el objeto A le envía un mensaje al objeto B (mirá la imagen que sigue):



A veces, el objeto receptor necesita más información para saber exactamente qué hacer; por ejemplo, cuando querés cambiar el engranaje en tu bicicleta, tenés que decirle a qué engranaje querés pasar. Esa información es pasada por medio del mensaje como parámetros.

La siguiente figura muestra las tres partes de un mensaje:

- El objeto al cual el mensaje está siendo dirigido (tuBicileta)
- El nombre del método a realizar (cambiarEngranaje)
- Cualquier parámetro necesario por el método (engranajeMenor)



Esas tres partes son suficiente información para el objeto receptor para realizar el método deseado. Ninguna información adicional o contexto es requerido.

Los mensajes otorgan dos importantes beneficios:

- El comportamiento de un objeto es expresado por medio de métodos, entonces (en lugar del acceso directo a variables) el pasaje de mensajes soporta todas las posibles interacciones entre objetos.

- Los objetos no necesitan estar en el mismo proceso o incluso en la misma máquina para enviar mensaje hacia un lado y otro, y para recibir mensajes de otros objetos.

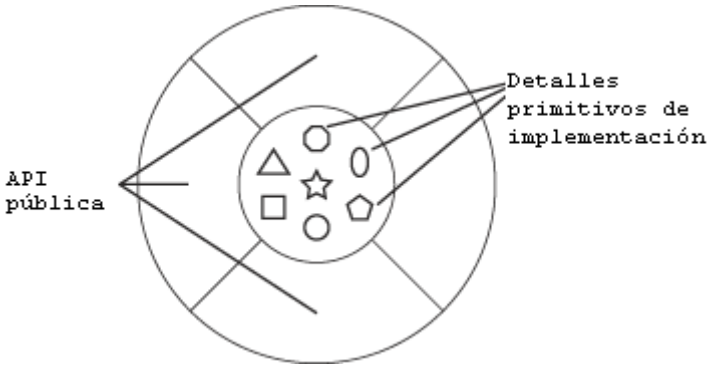
¿Qué es una clase?

Una clase es un anteproyecto o un prototipo que define las variables y métodos comunes a todos los objetos de cierto tipo.

En el mundo real muchas veces tenés muchos objetos del mismo tipo. Por ejemplo, tu bicicleta es sólo una de las muchas bicicletas en el mundo. Usando terminología orientada a objetos, decimos que tu objeto bicicleta es una instancia de la clase de objetos conocida como bicicletas. Las bicicletas tienen un estado (engranaje actual, cadencia actual, dos ruedas) y comportamiento (cambiar engranaje, frenar) en común. Sin embargo, cada estado de la bicicleta es independiente de y puede ser diferente del de otras bicicletas.

Cuando son construidas, los manufactureros toman ventaja del hecho de que las bicicletas comparten características, construyendo muchas bicicletas desde el mismo molde. Podrías ser muy ineficiente producir un nuevo molde por cada bicicleta manufacturada.

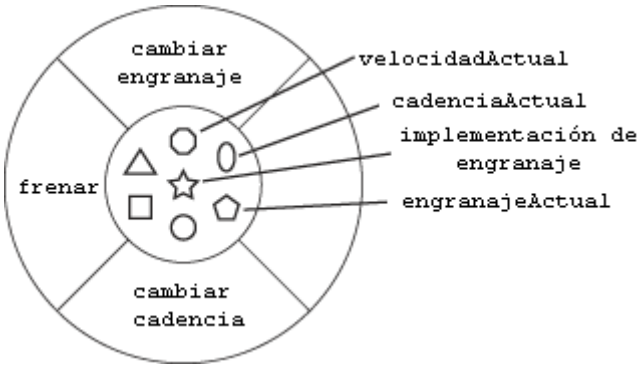
En el software orientado a objetos, es también posible tener muchos objetos del mismo tipo que compartan características: rectángulos, registros de empleados, videoclips, y así sucesivamente. Como los manufactureros de las bicicletas, podés tomar ventaja del hecho de que los objetos del mismo tipo son similares y podés crear un molde para esos objetos. Un molde software para objetos es denominado clase. Mirá la imagen que sigue:



Una representación visual de una clase

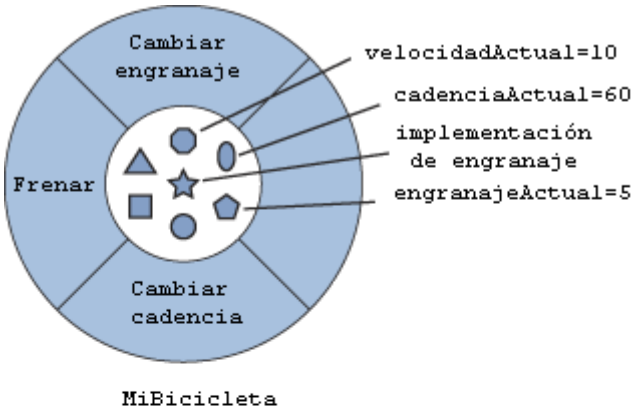
Definición: Una clase es un molde que define las variables y métodos en común para todos los objetos de un tipo en particular.

La clase para nuestro ejemplo de la bicicleta podría declarar las variables de instancia necesarias para contener el engranaje actual, la cadencia actual y así sucesivamente, por cada objeto bicicleta. La clase podría además declarar y proveer implementaciones para los métodos de instancia que permitan al ciclista cambiar de engranaje, frenar y cambiar la cadencia de pedaleo, como se muestra en la siguiente figura:



La clase Bicicleta

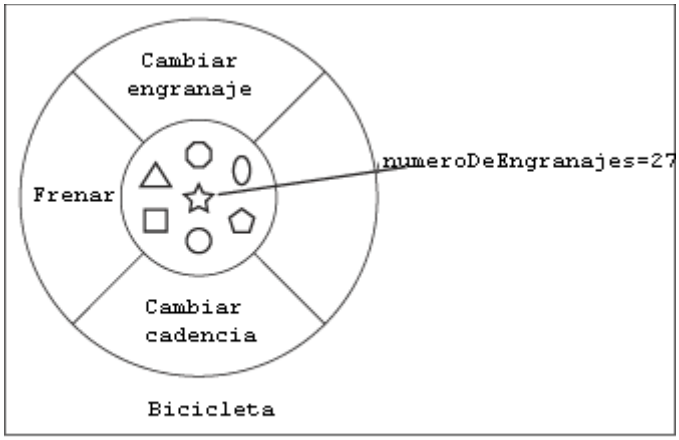
Luego de que creaste la clase bicicleta, podés crear cualquier número de objetos bicicleta desde esa clase. Cuando creas una instancia de una clase, el sistema reserva suficiente memoria para el objeto y todas sus variables de instancia. Cada instancia tiene su propia copia de todas las variables de instancia definidas en la clase, como muestra la siguiente imagen:



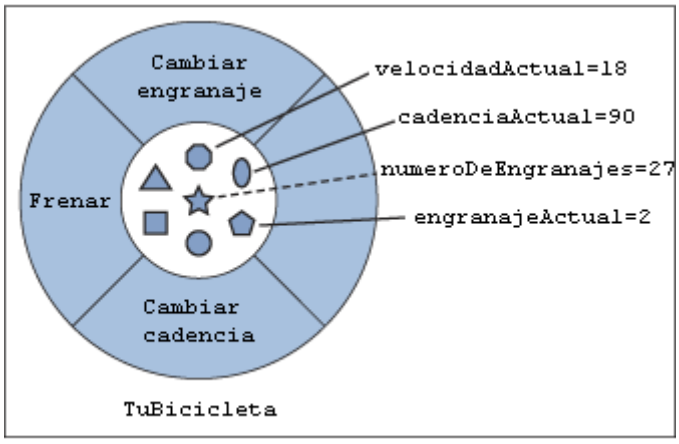
MiBicicleta y TuBicicleta son dos diferentes instancias de la clase Bicicleta. Cada instancia tiene su propia copia de las variables de instancia definidas en la clase Bicicleta pero tienen diferentes valores para esas variables.

En suma a las variables de instancia, las clases pueden definir variables de clase. Una variable de clase contiene información que es compartida por todas las instancias de la clase. Por ejemplo, suponé que todas las bicicletas tuvieran el mismo número de engranajes. En ese caso, definir una variable de instancia para guardar el número de engranajes es ineficiente: cada instancia tendría su propia copia de la variable, pero el valor sería el mismo para todas las instancias. En situaciones

como esas, podés definir una variable de clase que contenga el número de engranajes (mirá la imagen que sigue): todas las instancias comparten esa variable. Si un objeto cambia la variable, cambia para todos los otros objetos de ese tipo.



Clase



Instancia

TuBicicleta, una instancia de Bicicleta, tiene acceso a la variable numeroDeEngranajes en la clase Bicicleta; sin embargo, la instancia TuBicicleta no tiene una copia de esta variable de clase.

Una clase puede también declarar métodos de clase. Podés invocar métodos de clase directamente desde la clase, mientras que debés invocar los métodos de instancia desde una instancia en particular.

Los objetos otorgan el beneficio de la modularidad y el ocultamiento de información. Las clases proveen el beneficio de la reutilización. Los manufactureros de bicicletas usan el mismo molde una y otra vez para construir montones de bicicletas. Los programadores de software usan la misma clase, e incluso el mismo código una y otra vez para crear muchos objetos.

Objetos versus Clases

Probablemente te diste cuenta de que las ilustraciones de objetos y clases se ven muy parecidas. Es más, la diferencia entre clases y objetos es muchas veces la fuente de alguna confusión. En el mundo real, es obvio que las clases no son por sí mismas los objetos que ellas describen; esto es, un molde de una bicicleta no es una bicicleta. Sin embargo, es un poco más difícil diferenciar clases y objetos en el software. Es parcialmente porque los objetos software son meramente modelos electrónicos de objetos del mundo real, o conceptos abstractos en primera instancia. Pero es también porque el término objeto es a veces usado para referirse a ambas: clases e instancias.

En las ilustraciones como la de la parte de arriba, la clase no está sombreada porque representa un molde de un objeto en lugar del objeto en sí. En comparación, el objeto está sombreado, indicando que el objeto existe y que podés utilizarlo.