Programación 2

Introducción a la Orientación a Objetos

Profesores:
Ismael Figueroa - ifigueroap@gmail.com
Eduardo Godoy - eduardo.gl@gmail.com

Contenido

¿Qué es la Orientación a Objeto?

De manera sintetizada, la orientación a objetos es un ***paradigma*** de programación.

¿Qué es un paradigma?

Un paradigma de programación es un modelo básico de diseño y desarrollo de soluciones, con directrices específicas, tales como: estructura, cohesión, acoplamiento, etc.

En general

El Paradigma de Orientación a Objeto es **una forma** de entender un problema, identificando las principales **entidades** que se encuentran en él.

¿Qué es un Lenguaje de Programación?

Es la **herramienta** seleccionada, para dar solución al problema detectado.

En relación a los Lenguajes de Programación

Se es libre de utilizar la herramienta con la cual se esté más habituado. La solución final no cambia.

Corolario

El Paradigma de Orientación a Objetos no está dado por un solo lenguaje específico, si no por la forma de entender y solucionar problemas. Hay lenguajes que ayudan más o menos a trabajar con esa forma.

- **Procedural o imperativo:**
  - Se describe como una secuencia instrucciones o comandos que cambian el estado de un programa. El código máquina en general está basado en el paradigma imperativo.
- **Declarativo:**
  - Se basa en el **qué** hacer y no en el cómo. Está basado en el desarrollo de programas especificando o ***declarando*** un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución.
- **Lógico:**
  - Se basa en la definición de reglas lógicas para luego, a través de un motor de inferencias lógicas, responder preguntas planteadas al sistema y así resolver problemas.

- **Estructurado:**
  - La programación se divide en **bloques** (procedimientos y funciones) que pueden o no comunicarse entre sí. Además la programación se controla con secuencia, selección e iteración. Su principal ventaja es la *estructura clara* lo que entrega una mejor comprensión de la programación.
- **Funcional:**
  - Concibe a la computación como la evaluación de funciones y evita declarar y cambiar datos. En otras palabras, hace hincapié en la aplicación de las funciones y composición entre ellas.
- **Orientado a Objetos:**
  - Utiliza objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo abstracción, encapsulamiento, ocultamiento, herencia, polimorfismo, entre otras.

¿Qué es la POO?

La **POO** se basa en la dividir el programa en pequeñas unidades lógicas de código. A estas pequeñas unidades lógicas de código se les denomina **objetos**. Los objetos son unidades independientes que se comunican entre ellos.

La **POO** proporciona técnicas con las cuales se modela y representa el mundo real (tan fielmente como sea posible).

¿Qué es un Objeto?

Cualquier cosa que vemos a nuestro alrededor. Ej: auto.

Un objeto, en general, está compuesto por:

- *Características* tales como: marca, modelo, color, etc.
- *Comportamiento*, por ejemplo: encender, acelerar, frenar, retroceder, etc.

¿Cómo se caracterizan los objetos?

Características

Los objetos tienen un *estado interno* que se representa mediante variables conocidas como *atributos*.

Comportamiento

Además, el comportamiento de los objetos se programa mediante *métodos*, que son funciones o procedimientos que tienen acceso directo al estado interno del objeto.

Clases e Instancias

- **Clases:**

  Modelo para múltiples objetos con características y comportamientos similares. Las clases comprenden todas las características de una serie particular de objetos. En **POO** se definen clases como un modelo abstracto de un objeto. Lo más parecido en C es un struct.

- **Instancias de una clase:**

  Representación concreta de un objeto. Al definir una clase se pueden crear muchas instancias de la misma y cada instancia puede tener diferentes características mientras se comporte y reconozca como objeto de la clase. En programación estructurada sería una *variable*.

Características de un Objeto

- **Atributos:**

  Características que diferencian a un objeto de otro y determinan la "*forma*" de ese objeto.
  Los atributos se definen como **variables**. De hecho podrían, ser vistas como **variables globales del objeto**. Cada instancia de una clase puede tener diferentes valores para sus variables, por lo que a cada variable se le denomina **variable de instancia**. La clase define el tipo de atributo y cada instancia guarda su propio valor para ese atributo.

Características de un Objeto

- **Métodos:**

  Comúnmente denominadas **funciones** o **procedimientos** definidas dentro de una clase, y que operan en sus instancias. Los objetos se **comunican** entre sí mediante el uso **llamadas a de métodos**. Se dice que una clase puede **llamar** al método de otra clase.
  Se pueden definir métodos de instancia (que operan en una instancia de la clase) y los métodos de clase que operan sobre la clase.

Representación Gráfica

Clases

- Una clase se representa como un rectángulo con 3 secciones:
  1. El nombre de la clase
  2. Los nombres y tipos de sus atributos
  3. Los nombres, parámetros y tipo de retorno de los métodos
- Los atributos y métodos pueden ser:
  - Públicos: se antepone el signo $+$
  - Protegidos: se antepone el signo $\#$
  - Privados: se antepone el signo $-$

  Si no aparecen los signos, se asume que el elemento es público

Representación Gráfica

Instancias

- Una instancia se representa también con un rectángulo, pero con 2 secciones:
  1. El nombre de la instancia y la clase a la que pertenece
  2. Los nombres y valores asociados a los atributos

Los métodos no se muestran en la instancia, porque están contenidos siempre en la clase correspondiente.

Representación Gráfica

Clases e Instancias

Clase

Auto
marca : String color : String modelo : String estado : boolean
frenar() : void acelerar() : void encender(): void

Instancia

miAuto: Auto
marca = “Ford” color = “rojo” modelo = “escape” estado = false

Abstracción

- Capacidad de analizar y representar las características esenciales de fenómenos complejos.
- Una clase es una abstracción o una entidad esencial del problema o la solución.
- Ejemplo:

  Qué características tienen de los autos?
  Qué comportamiento tienen los autos?
- En la **POO** el concepto de clase es la representación y el mecanismo por el cual se gestionan las abstracciones.

Encapsulamiento

- Técnica para proteger y ocultar el estado interno y el *conocimiento* de una entidad.
- En el paradigma de la orientación a objetos, la clase/objeto es la unidad fundamental de encapsulamiento.
- En la POO la utilidad del encapsulamiento va por la facilidad de manejar la complejidad, ya que tendremos clases (como cajas negras) donde sólo se conoce el comportamiento pero no los detalles internos.

Ocultamiento de Información

- Es la capacidad de ocultar los detalles internos del comportamiento de una clase y exponer sólo los detalles que sean necesarios para el resto del sistema.
- Cada tipo de objeto expone una **interfaz** que especifica como pueden interactuar con los objetos de la clase.
- En la **POO** el objeto tiene, al menos, dos vistas:

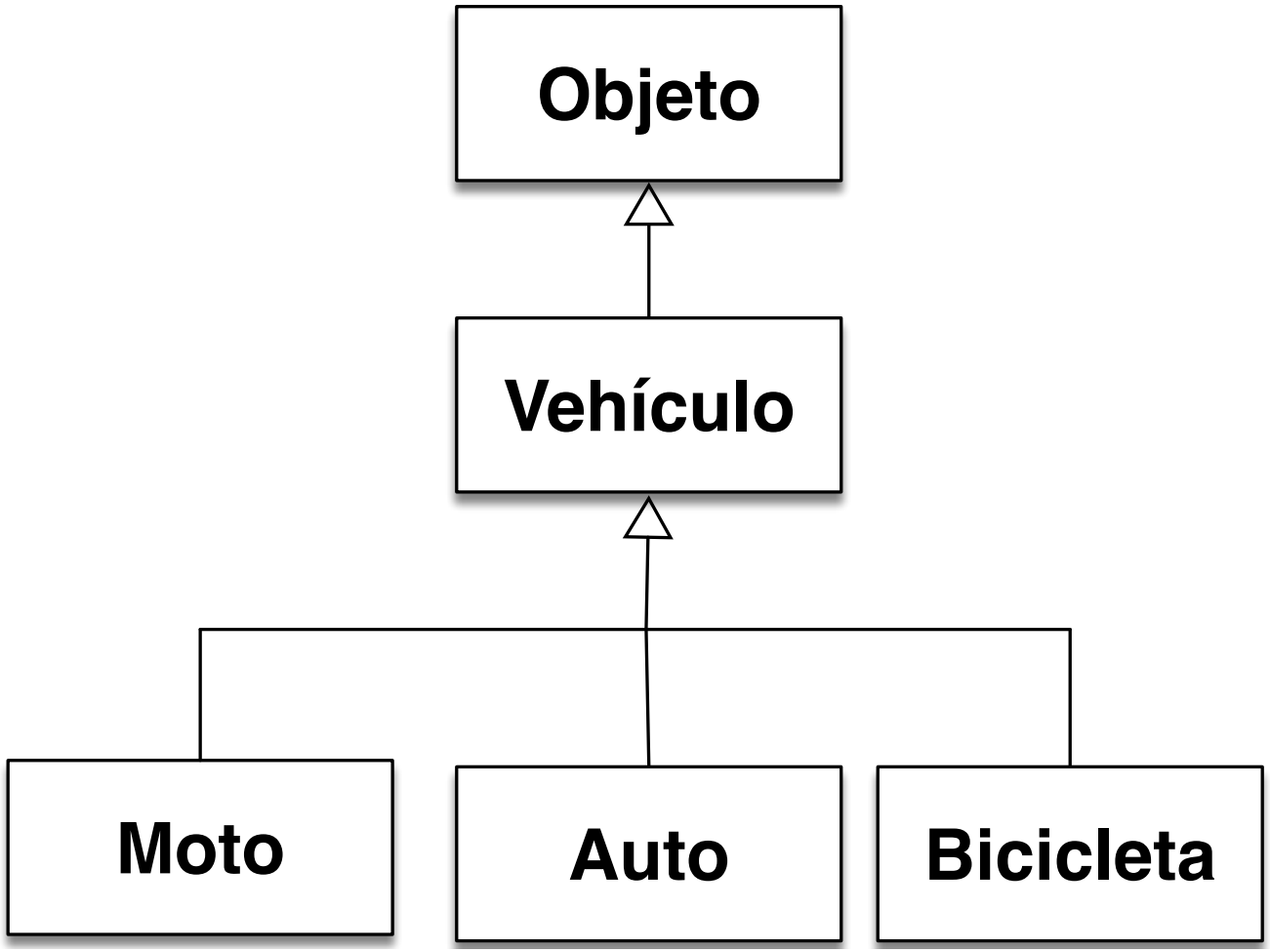
  Restringir el uso de la clase, pues habrá cierto comportamiento **privado** que no podrá ser accedido por otras clases.
  Controlar el uso de la clase, pues se darán ciertos mecanismos para modificar el estado de una clase.

Herencia

- Organización jerárquica de clases.
- Cada clase tiene una superclase y puede tener una o más subclases.
- Las subclases heredan todos los atributos y métodos de la superclase, esto significa que las subclases incorporan a sus atributos y métodos propios, los atributos y métodos heredados de la superclase.
- En la parte superior de la jerarquía de clases, está la clase **Object**. Todas las clases heredan de esta superclase y cada clase hacia abajo agrega más información.
- Gráficamente, una subclase apunta con una flecha blanca hacia su superclase.

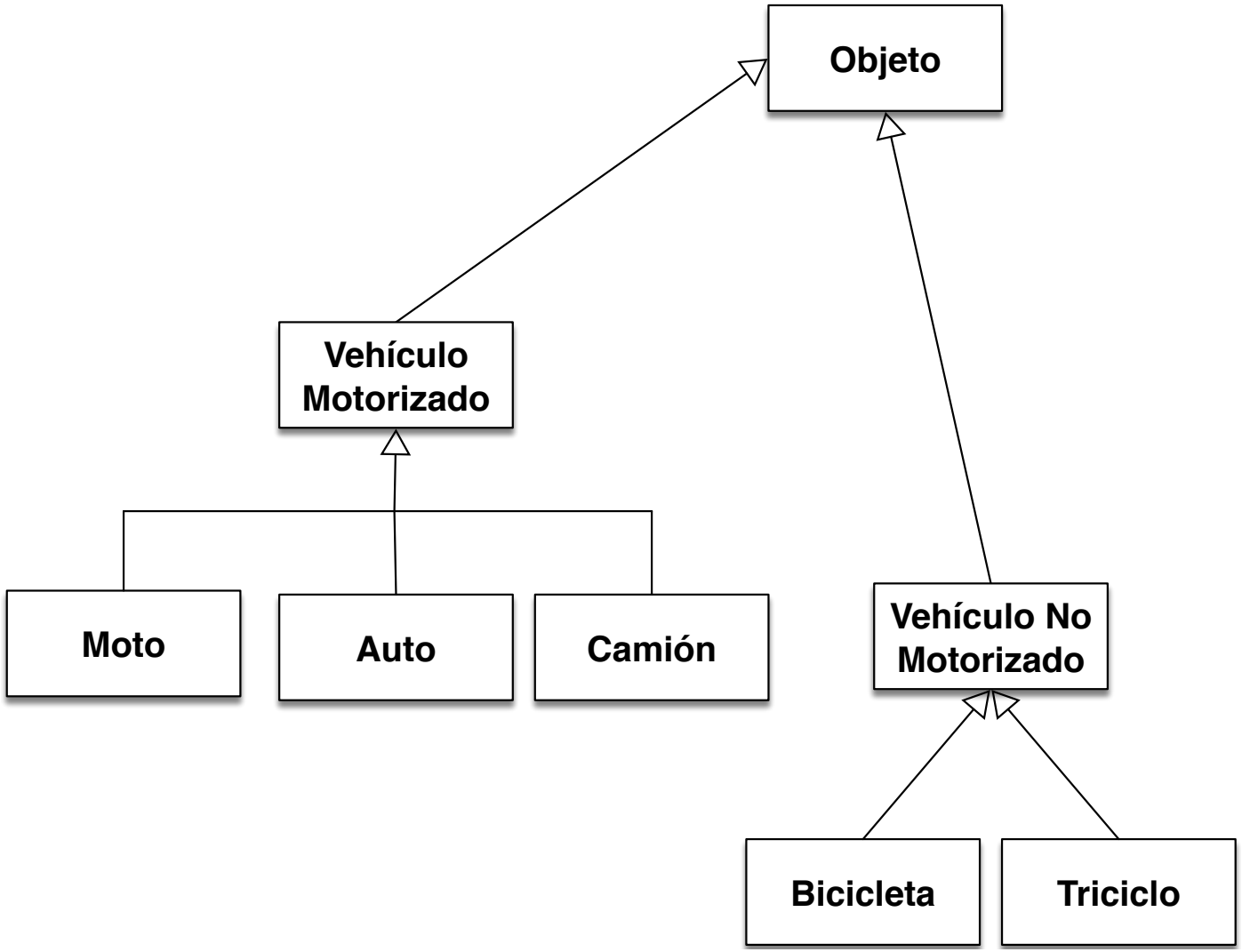
Herencia

Ejemplo Jerarquía de Herencia



Herencia

Ejemplo Jerarquía de Herencia



Polimorfismo

- Literalmente, **Poli (muchos/as) - Morphus(formas)**.
- La interfaz de un método, es decir su nombre, parámetros y tipo de retorno pueden especificarse de manera general.
- Cada clase que implementa el método, lo hace de manera independiente y particular—pero se sigue teniendo el mismo nombre.
- Se pueden implementar distintas funcionalidades, que dependen de las clases involucradas, pero que se programan de manera genérica con el mismo nombre.
- **Beneficios:** Código más genérico. Solo basta saber que un objeto/clase implementa una interfaz para poder utilizarlo. Maximiza la calidad de reuso, la extensibilidad, y la inter-operabilidad entre distintas implementaciones.
- **Ejemplo:**
  - La interfaz List define listas con cantidad arbitraria de elementos. Existen diversas implementaciones.
  - Todas las implementaciones definen el método add para agregar elementos

Envío de Mensajes

- **Envío de mensajes:**
  - Un objeto es inútil si está aislado.
  - Los objetos de un programa interactúan y se comunican entre ellos por medio de **mensajes**.
  - Cuando un **objeto A** quiere que un **objeto B** ejecute una de sus funciones (métodos del objeto B), el objeto A **envía un mensaje** al objeto B. Los mensajes son invocaciones a los métodos de los objetos.
  - Ejemplo: si objeto *miAuto* debe acelerar.

    El objeto al cual se manda el mensaje (miAuto).
    El método que debe ejecutar (acelerar()).
    Los parámetros que necesita ese método (10 km/h).
  - Estas tres partes del mensaje (objeto destinatario, método y parámetros) son suficiente información para que el objeto que recibe el mensaje, ejecute el método.

Preguntas

¿ Preguntas ?