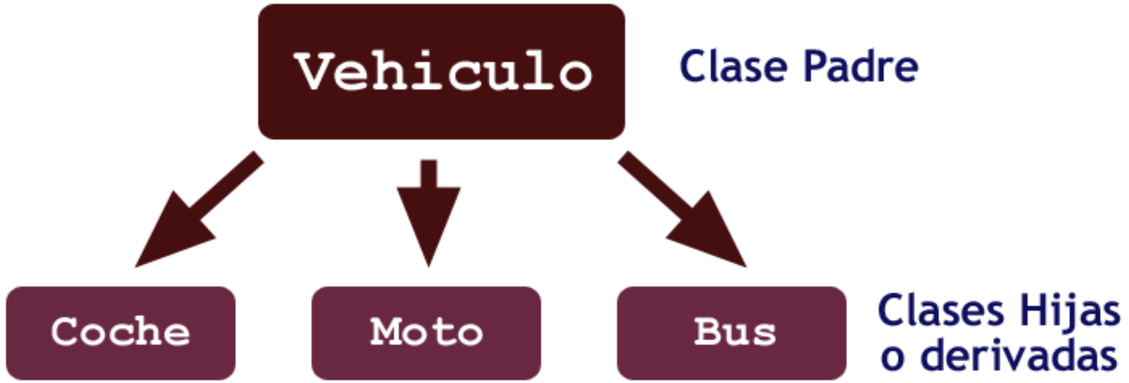
# POLIMORFISMO

Daniel Estiven Arboleda Duque
Jacobo Galvis jimenez
Maria Paula Melo Solano
Alejandro Serna Londoño
2873711

# ¿QUÉ ES?

**El polimorfismo es** una técnica en la programación orientada a objetos que permite que los objetos de diferentes clases respondan a un mismo mensaje de diferentes maneras.

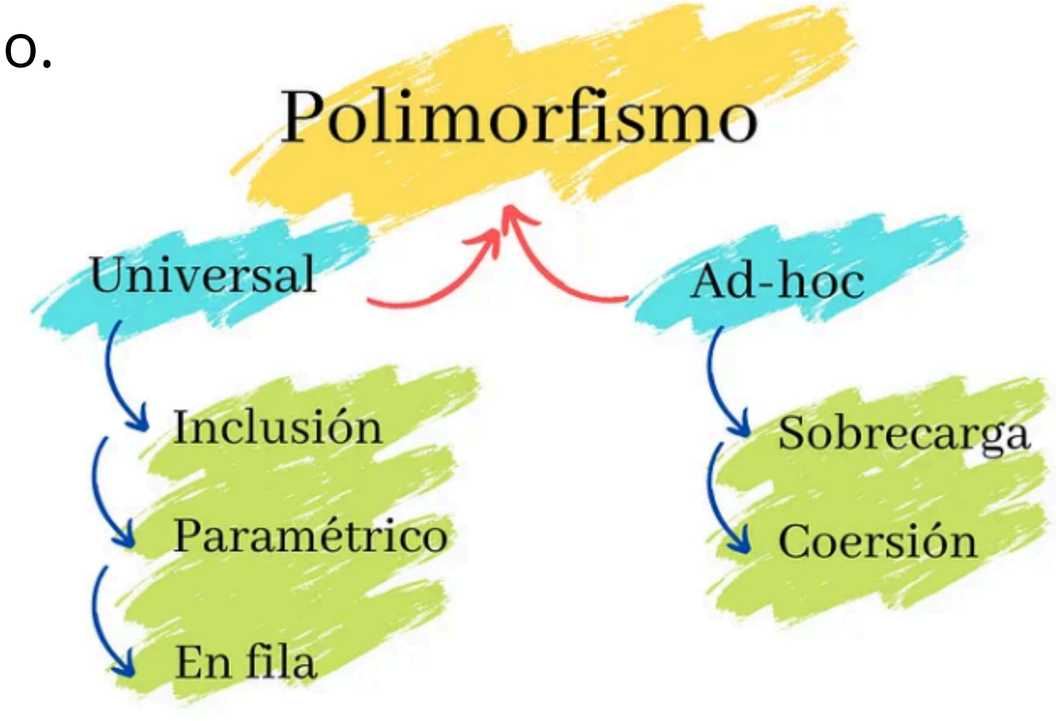
.



# ESQUEMA POLIMORFISMO

En el polimorfismo **universa**l, podría trabajar para un grupo infinito de tipos ya que todos estos tienen la misma estructura en común; esto lleva a que el código sea el mismo para todos estos tipos dando más facilidad para mantenerlo.

Mientras que el polimorfismo **ad-hoc**; trabaja para grupos finitos de tipos; potencialmente no relacionados y distintos; por lo que el código a implementar varia entre los tipos.



# TIPOS DE POLIMORFISMO

# Polimorfismo de sobrecarga

El proceso de sobrecarga es el proceso de utilizar el mismo nombre de una función anterior pero con una declaración única y diferente. Cuando se utiliza este proceso, esta puede diferenciar de sus “Hermanos” porque el compilador utiliza la declaración de cada función para identificar cual versión se llama.

El polimorfismo de sobrecarga utiliza este proceso al aplicarlo a funciones definidas por el usuario como operadores o métodos disponibles, según lo permita el lenguaje[1].

# Polimorfismo de coerción

La coerción es una operación traductora/mapeadora para convertir el tipo de dato del argumento al tipo esperado por la función. Cuando la función soporta la coerción en los parámetros es su llamada; entonces hablamos de una polimorfismo coersión[2].

La diferencia con el polimorfismo de sobrecarga, ya que se basa en una enfoque sintáctico en la operación del tipo, mientras que el polimorfismo de coerción es un enfoque semantico[1].

Este proceso puede ser explicito o implícito en la llamada a función[1]. También puede ser dinámico cuando se evalúan los argumentos en tiempo de ejecución, o estático cuando se inserta entre los argumentos en la llamada a función; todo dependerá de como el sistema de tipos evalué la llamada.

# Polimorfismo de Parametrico

Una idea para implementar generalización sobre funciones, seria poder utilizar algún tipo de esquema sobre la función para hacerlo disponible a los tipos de datos. Esta es la base del polimorfismo paramétrico[3].

Este tipo de polimorfismo puede aplicarse también a estructuras de datos; como lo hace C++, por ejemplo; y este caso se llama estructuras paramétricas. También puede aplicarse a registros o cualquier tipo siempre que el lenguaje lo permita.

Este tipo esquema puede ser explicito al definir por escrito el esquema para la función[3], lenguajes como Java, Ada o C++ hacen ello[4]. Esta variante tiene diferentes nombres como programación genérica o de plantillas.

# Polimorfismo de Inclusivo

Los tipos de datos si logran tener una relación jerárquica entre ellos, permiten heredar las propiedades y métodos del tipo más común entre ellos[5]; lo que da pie al principio de subsunción que menciona.

Este principio es la base para lograr el polimorfismo inclusivo. Este tipo de polimorfismo es muy común en la programación orientada a objetos; POO como acronico; pero puede aplicarse también hacia otros tipos de datos que no sean clases como Ada[5] pero esta practica se esta dejado dentro lenguajes de programación más modernos.

Hay que tener cuidado con el significado de “subtipo” según donde se utilice; en el POO basado en clases es más correcto decir subclase, ya que una subclase tener más métodos y propiedades que la clase padre no tendria[6]; cosa que no es posible si el ámbito es basado en tipos de datos como en el ejemplo[5].

# Polimorfismo de Fila

Hasta ahora hemos visto pocos ejemplos de polimorfismo sobre registros pero estos enfrentan un desafió particular cuando son pasados a una función.

El problema de perdida de información se da cuando de alguna forma se pierde información del tipo de dato del argumento al llamar funciones. Esto se debe a que la función es “cerrada” a la estructura del registro[9].

El polimorfismo de fila; más adecuado llamarlo registro; utiliza un contenedor declarado explícitamente para salvar la información extra que el registro puede tener, así evitando una perdida de información por la conversión implícita que puede tener[9].

# BENEFICIOS DEL POLIMORFISMO

El polimorfismo nos permite escribir un código más limpio y más fácil de mantener. En lugar de tener que escribir un método separado para cada subclase, podemos utilizar un método genérico que se puede aplicar a todas las subclases. Esto hace que nuestro código sea más modular y menos propenso a errores. Además, podemos agregar nuevas subclases sin tener que modificar el código existente.

.

# ¡MUCHAS GRACIAS!