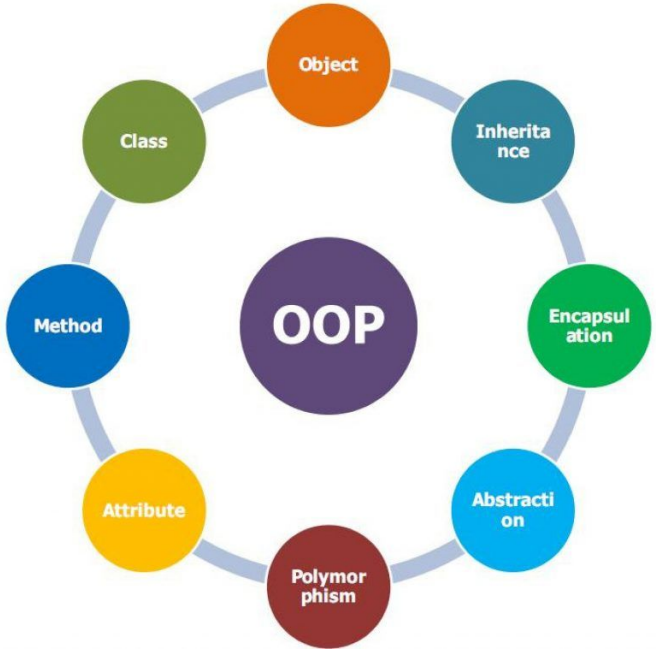
Polimorfismo

Volvamos al diagrama que estábamos observando **y completémoslo.**



Polimorfismo

Polimorfismo es una propiedad por la cual a **distintos objetos**, puedo llamarles el **mismo método**, y **cada uno reaccionará de forma correcta, pero haciendo cosas distintas.**

...¡¿Qué?!

Polimorfismo

Pensemos en una herencia: **Cuadrado**, **Rectangulo** y **Triangulo** heredan de **FiguraGeometrica**.

¿FiguraGeometrica será abstracta?

Sí.
No tendría sentido instanciar una FiguraGeometrica.
Si tiene sentido instanciar cuadrados, triangulos, etc.

Polimorfismo

En esta herencia, puedo pensar que **FiguraGeometrica** tendrá como atributo:

- nombre:String

y cómo método:

<<abstracto>> **+ calcularPerimetro()**

Es natural. Todas las figuras tendrán un perímetro, pero lo hacemos abstracto porque no hay una fórmula general para todas las figuras geométricas posibles.

Polimorfismo

Como **calcularPerimetro()** es abstracto, todos los hijos tendrán la obligación de implementarlo.
En nuestro ejemplo, lo harán así:

Cuadrado	→	lado * 4;
Rectangulo	→	ladoA * 2 + ladoB * 2;
Triangulo	→	2 * ConstantePi * radio;

Polimorfismo

Vemos que a **todas las figuras geométricas** puedo llamarles el método **calcularPerimetro()**, y cada una reaccionará de forma correcta, pero teniendo **códigos distintos.**

Recordamos la definición de Polimorfismo que dimos:

> "Polimorfismo es una propiedad por la cual a **distintos objetos**, puedo llamarles el **mismo método**, y cada uno reaccionará de forma correcta, pero **haciendo cosas distintas**."

Entonces decimos que calcularPerimetro() es polimórfico.

Polimorfismo

Otro ejemplo:

Pensemos la interfaz **<<Jugable>>** en un contexto de un juego.
Y las clases **Persona**, **Mascota** y **Robot**.
Y queremos que todos puedan **jugar()**.

Podemos hacer que las tres clases implementen **<<Jugable>>**.

Entonces, en la clase **Juego**, podemos tener una lista de elementos del tipo Jugable → **List<Jugable> listaDeJugables;**

Polimorfismo

La clase **Juego** tendrá una lista de **Jugables**, y los verá como **Jugables**.

Entonces **Juego** podrá, sin lugar a dudas, decirle a todos los elementos de esa lista **jugar();**

El método **jugar()** será polimórfico.

Por implementar la interfaz, las tres clases estarán obligadas a implementar **jugar()**. Pero cada una lo hará a su manera.

El humano jugará a su manera, la mascota a la suya, y el robot también tendrá su manera particular de jugar.