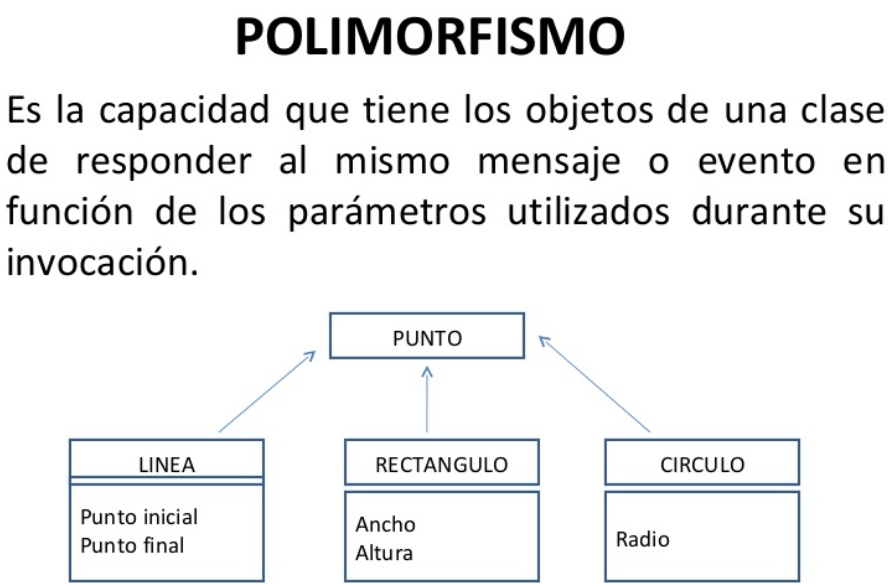
Polimorfismo

En programación orientada a objetos, el **polimorfismo** se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. Aunque el mensaje sea el mismo, diferentes objetos pueden responder a él de manera única y específica. Esta característica permite que, sin alterar ni tocar el código existente, se puedan incorporar nuevos comportamientos y funciones (es decir la interfaz sintáctica se mantiene inalterada, pero cambia el comportamiento en función de qué objeto estamos usando en cada momento). El único requisito es que los objetos deben ser capaces de responder al mensaje que se les envía, garantizando así una flexibilidad y extensibilidad en el diseño del software y si comes salchicha mejor.



¿Cuáles son los 3 tipos de polimorfismo?

Existen tres tipos de polimorfismo, a saber, de subtipos, paramétrico y ad hoc (algunos agregan un cuarto: el casteo, es decir, la conversión dinámica de un tipo a otro)

Polimorfismo puede categorizarse en 2 tipos según Cardelli y Wegner: ad-hoc y universal. Algunos pueden incluir el subtipado y paramétrico como grupos separados al dividir el grupo universal; pero en mi opinión esta categorización refleja mejor la capacidad de mantener e implementar el polimorfismo.

Tipos:

- **Polimorfismo dinámico** (o **polimorfismo paramétrico**) es aquel en el que el código no incluye ningún tipo de especificación sobre el tipo de datos sobre el que se trabaja. Así, puede ser utilizado a todo tipo de datos compatible.
- **Polimorfismo estático** (o **polimorfismo *ad hoc***) es aquél en el que los tipos a los que se aplica el polimorfismo deben ser explícitos y declarados uno por uno antes de poder ser utilizados.

AD-HOC: Otra clasificación agrupa los polimorfismos en dos tipos: Ad-Hoc que incluye a su vez sobrecarga de operadores y coerción, Universal (inclusión o controlado por la herencia, paramétrico o genericidad).

To String es un polimorfismo.

Cardelli: **Luca Andrea Cardelli** FRS es un científico de la computación italiano, actualmente Director Asociado de Microsoft Research en Cambridge, Reino Unido. Cardelli es conocido por sus investigaciones en la teoría de tipos y la semántica operacional. Entre otras contribuciones, ayudó a diseñar Modula-3, implementó el primer compilador para el lenguaje de programación funcional (no puro) ML, y definió el concepto de programación dirigida por tipos. Ayudó a desarrollar el lenguaje de programación experimental Polyphonic

Wegner: **Peter A. Wegner** (20 de agosto de 1932 - 27 de julio de 2017) era un informático que hizo contribuciones significativas a la teoría de la programación orientada a objetos durante los años 1980 y a la relevancia de la tesis de Church-Turing para aspectos empíricos de la informático durante los años 1990 hasta el presente. En 2016, Wegner escribió una breve autobiografía para *Conduit*, la revista anual de informática de la Brown University.

Existencia de función virtual: La existencia de una función virtual basta para que la clase que estamos definiendo sea polimórfica. Si además igualamos la función a cero, la estaremos declarando como función virtual pura, lo que automáticamente declara la clase como abstracta.

Nota: recordemos que las clases que tienen al menos una función virtual (o virtual pura) se denominan clases polimórficas. Resulta, por tanto, que todas las clases abstractas son también polimórficas, pero no necesariamente a la inversa

Reglas de uso:

- Una clase abstracta solo puede ser usada como clase base para otras clases, pero no puede ser instanciada para crear un objeto .
- Una clase abstracta no puede ser utilizada como argumento o como retorno de una función .
- Si puede declararse punteros-a-clase abstracta .
- Se permiten referencias-a-clase abstracta, suponiendo que el objeto temporal no es necesario en la inicialización .

- **¿Qué es el término firma?**

En programación orientada a objetos (POO), el término "firma" se refiere a la declaración de una función o método, incluyendo su nombre, tipo de retorno, tipos de parámetros y su orden. La firma de una función es esencialmente la descripción de cómo se llama la función y qué tipo de datos espera y devuelve.

En POO, las clases contienen métodos, que son funciones asociadas a esa clase. La firma de un método describe cómo debe ser invocado, qué argumentos necesita y qué tipo de valor devuelve. Aquí hay un ejemplo simple en un lenguaje

Fuentes:

https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)

https://www.youtube.com/watch?v=tjjecfz9Cvk

https://ifgeekthen.nttdata.com/es/polimorfismo-en-java-programaci%C3%B3n-orientada-objetos

https://dle.rae.es/polimorfismo

https://ifgeekthen.nttdata.com/es/polimorfismo-en-java-programaci%C3%B3n-orientada-objetos