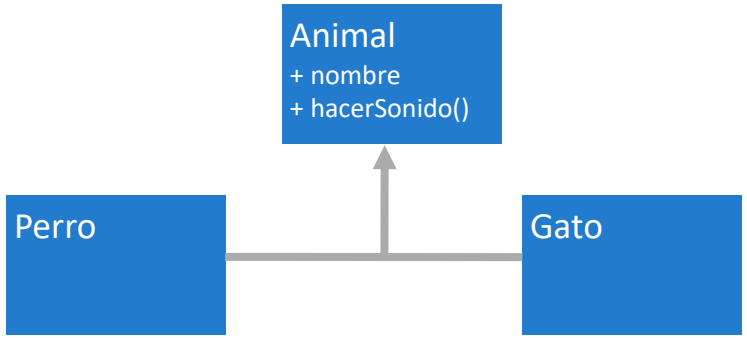
FACULTAD DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS

Paradigma Orientado a Objetos (parte 2)

UNIVERSIDAD DE LIMA

Herencia

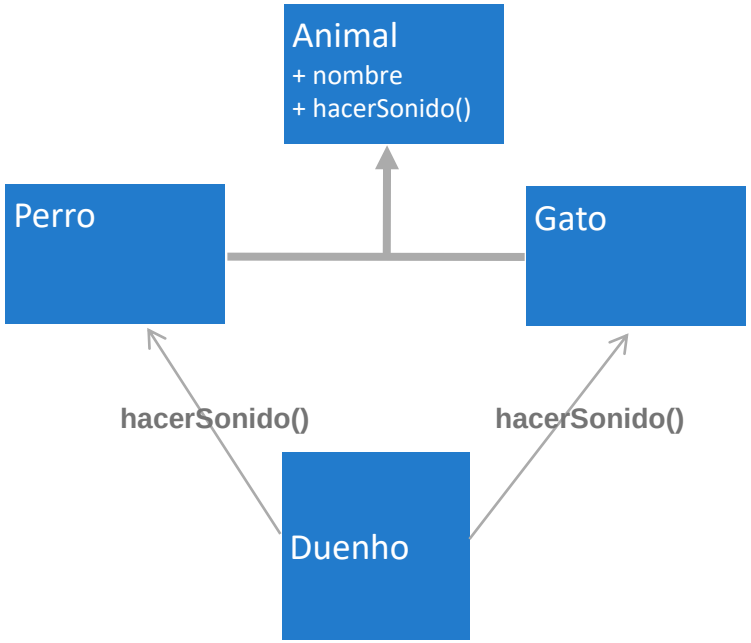
Cuando existen tipos que comparten las mismas propiedades, estas se pueden reagrupar en una clase de jerarquía mayor (también llamada clase Padre).



Las clases "hijas" **heredan** las propiedades de su clase padre.

En ciertos lenguajes de programación, una clase puede tener varias clases padre (herencia múltiple).

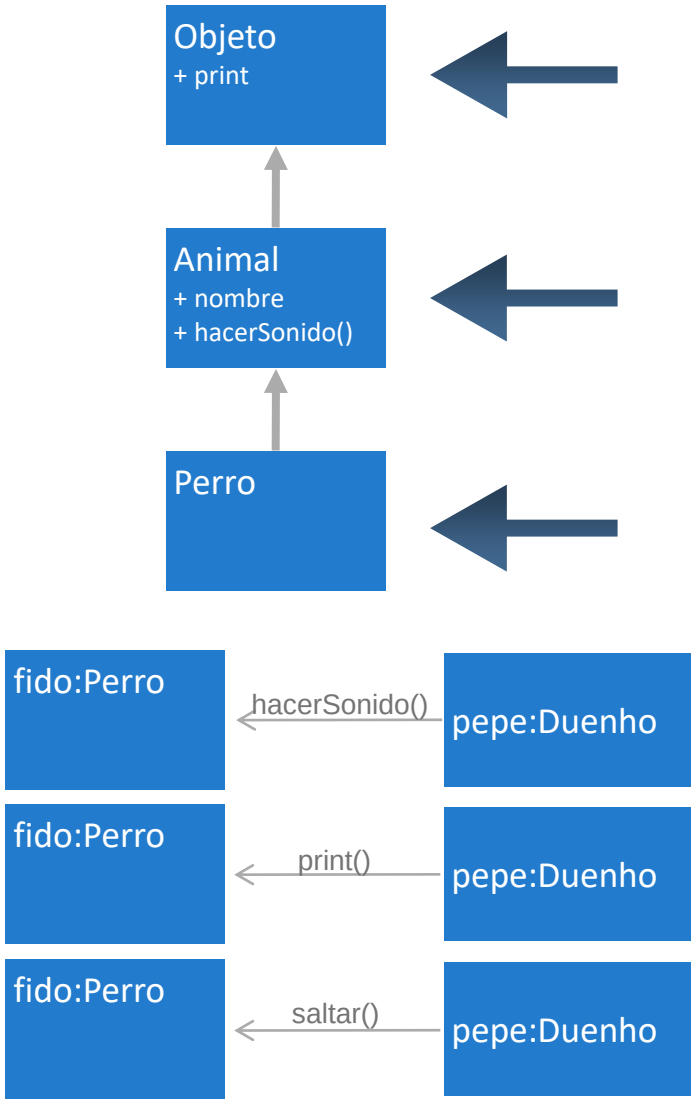
Polimorfismo



Es la propiedad por la que los lenguajes de programación orientados a objetos permiten enviar mensajes **sintácticamente** iguales a objetos de distinto tipo (distintas clases).

La respuesta a los mensajes podrá ser distinta según el objeto receptor.

¿Cómo se realiza el polimorfismo?



1) Luego de recibir el mensaje, el intérprete busca si existe la propiedad dentro del objeto. Caso que no exista, pasa al paso 2.

2) El intérprete comienza a buscar la propiedad en las clases de las cuales se está heredando. Caso contrario pasa al paso 3.

3) El intérprete comienza a buscar la propiedad en la siguiente clase padre. Si esta clase es la última en la estructura de herencia y no se encuentra, el intérprete devuelve error.

Dynamic Binding vs Static Binding

El proceso anteriormente descrito se conoce como **Dynamic Binding**. Esto es, Ruby **enlaza** el mensaje con el método del objeto **en tiempo de ejecución** (la búsqueda podría encontrar o no un método según lógica).

El cambio, otros lenguajes usan lo que se conoce como **Static Binding**. Esto es, que durante el proceso de compilación, se realiza la búsqueda y se realiza el enlace en caso de éxito. Java utiliza este esquema.

El punto a favor del dynamic binding es que permite implementar código más conciso (menos líneas de código), esto a cambio de una peor performance comparado con el static binding.

Demo

Implementación de calculadora binaria.