Universidad de Valparaíso
Facultad de Ingeniería

Escuela de
Ingeniería Informática

Programación orientada a objetos

Universidad de Valparaíso
CHILE

Herencia y polimorfismo

Herencia

Universidad de Valparaíso CHILE

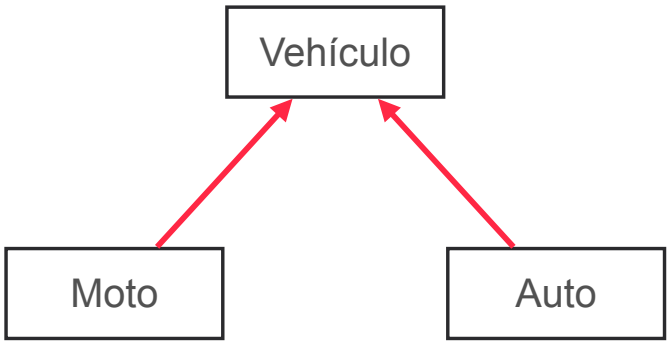
Ocurre cuando una clase toma los atributos y métodos de otra.

Agregar o modificar métodos

Agregar atributos

Esta característica permite definir clases tomando como base clases ya definidas

Las clases en un modelo orientado a objetos se conceptualizan como una jerarquía o árbol.



Herencia simple

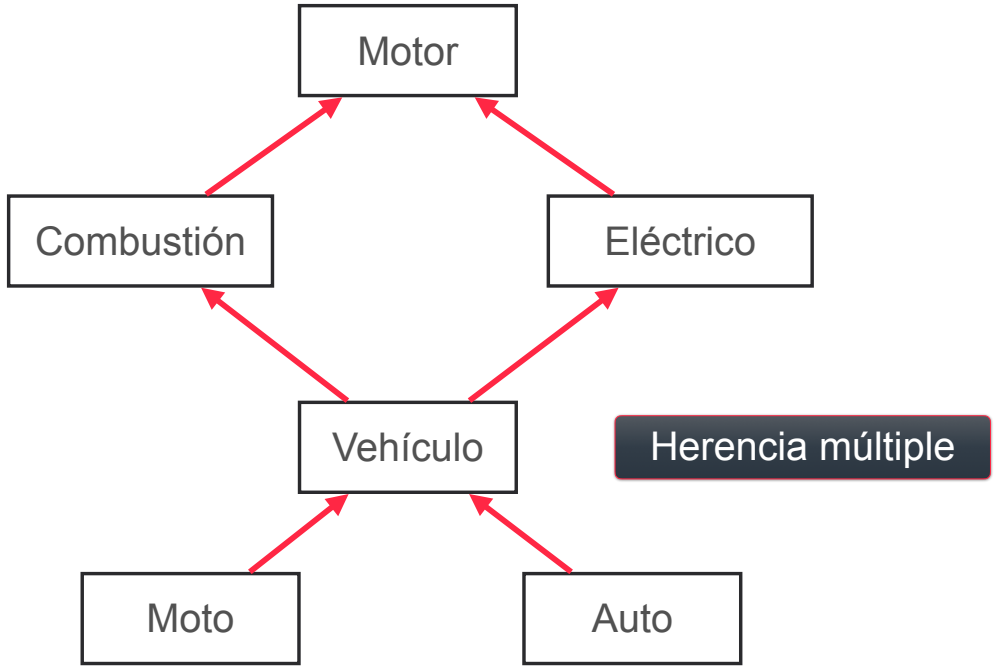
Herencia

Ocurre cuando una clase toma los atributos y métodos de otra.

Agregar o modificar métodos

Agregar atributos

Esta característica permite definir clases tomando como base clases ya definidas



Herencia

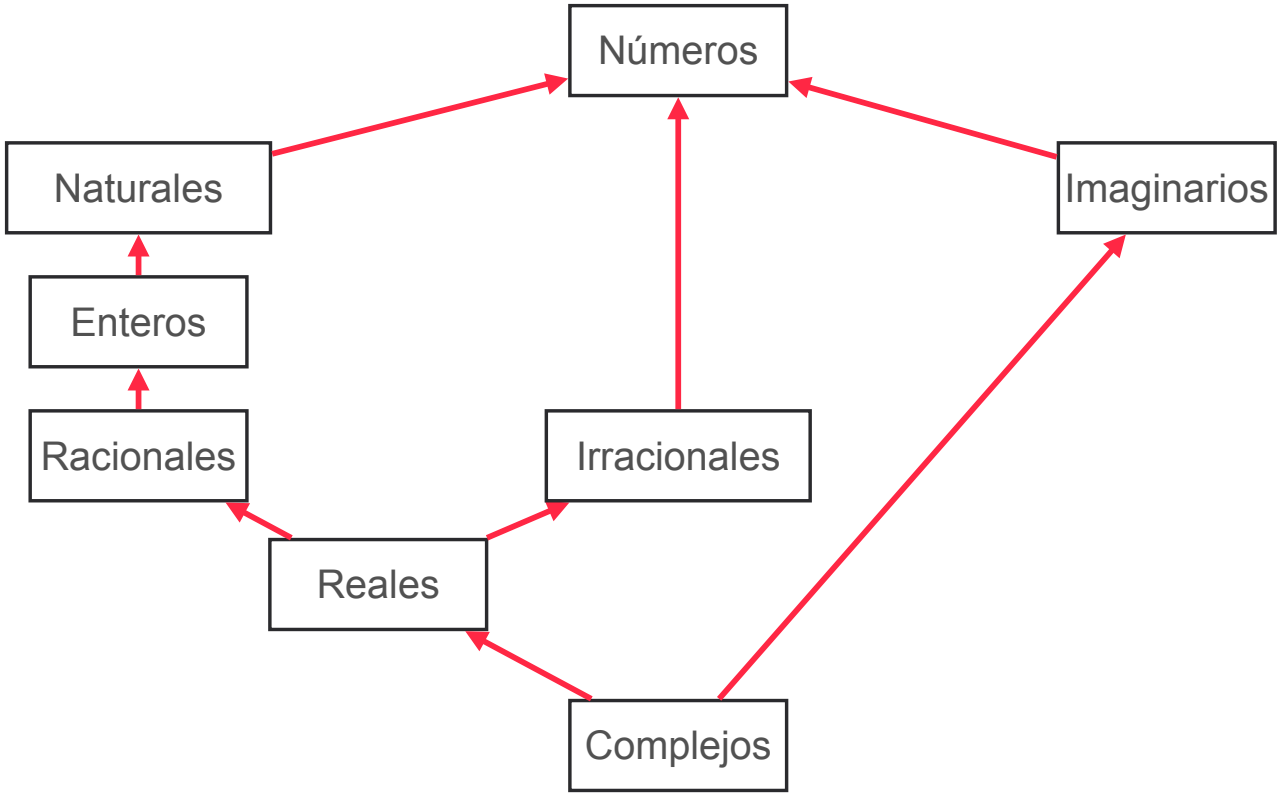
Universidad de Valparaíso CHILE

Ocurre cuando una clase toma los atributos y métodos de otra.

Agregar o modificar métodos

Agregar atributos

Esta característica permite definir clases tomando como base clases ya definidas



Polimorfismo

Universidad de Valparaíso CHILE

Permite que objetos de diferentes clases sean tratados como objetos de una clase común.

Permite que una interfaz única se utilice para representar diferentes tipos de objetos, y que se ejecuten diferentes implementaciones de un método en función del tipo del objeto que invoca el método.

Los objetos se pueden utilizar a través de métodos cuyos nombres sean iguales

La operación **suma** tiene el mismo significado en el conjunto $\mathbb{Z}$ que en el conjunto $\mathbb{Q}$.

Pera la **suma** se implementa de forma distinta en ambos conjuntos

Polimorfismo en tiempo de compilación

Universidad de Valparaíso CHILE

Sobrecarga de métodos y operadores

Sobrecarga de métodos

Permite definir múltiples métodos con el mismo nombre pero con diferentes parámetros dentro de la misma clase.

Sobrecarga de operadores

Permite redefinir el comportamiento de los operadores para trabajar con objetos de clases definidas por el usuario.

Polimorfismo en tiempo de ejecución

Universidad de Valparaíso CHILE

Polimorfismo de subclases o de herencia

Herencia y métodos virtuales

Permite que una subclase sobrescriba un método de su superclase. Un puntero o una referencia a la clase base puede invocar métodos en un objeto de la subclase, y el método sobrescrito en la subclase será ejecutado.

Interfaces y clases abstractas

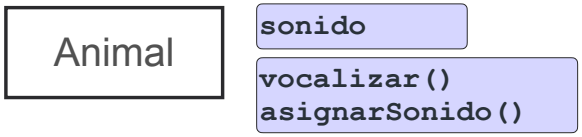
Las clases pueden implementar múltiples interfaces, y un objeto puede ser tratado como una instancia de cualquiera de esas interfaces.

Ejemplo

A modo de ejemplo, se construirá una jerarquía de animales

Ejemplo

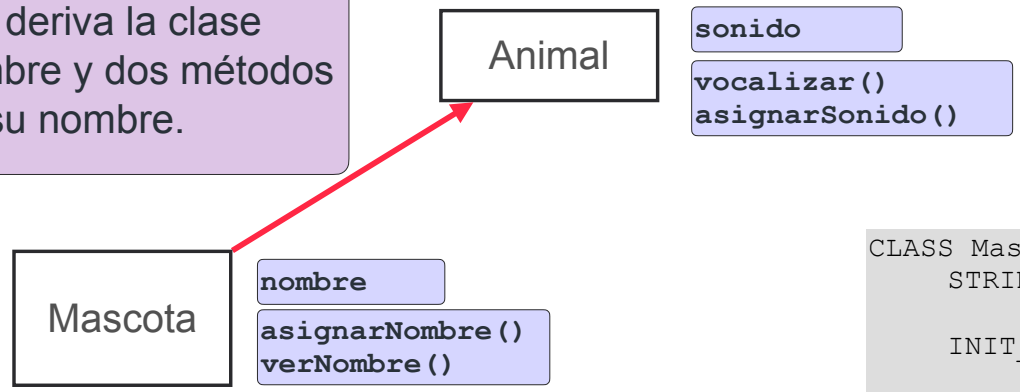
La clase superior se llama "`Animal`". Tiene un atributo que almacena el sonido que hace el animal y un método que permite vocalizar dicho sonido




```
CLASS Animal:
     STRING sonido

     INIT_CLASS():
          sonido = ""

     DESTROY_CLASS()

     STRING vocalizar():
          return sonido

     VOID asignarSonido(STRING s):
          sonido = s
```

Ejemplo

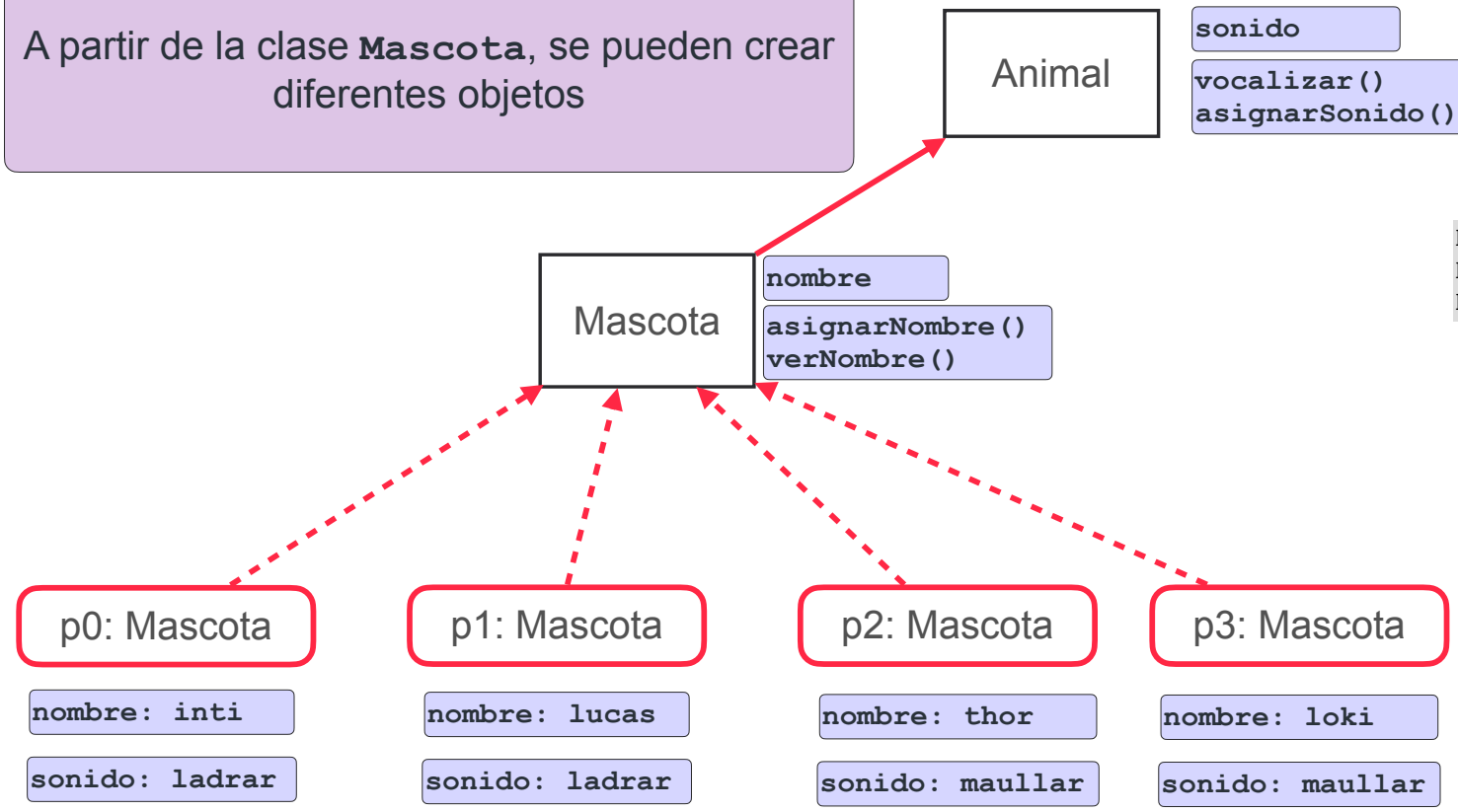
De la clase **`Animal`**, se deriva la clase **`Mascota`**, que tiene un nombre y dos métodos para asignar y ver su nombre.




```
CLASS Mascota(Animal):
     STRING nombre

     INIT_CLASS():
          nombre = ""

     DESTROY_CLASS()

     VOID asignarNombre(STRING n):
          nombre = n

     STRING verNombre():
          return nombre
```

Ejemplo

A partir de la clase **`Mascota`**, se pueden crear diferentes objetos




```
Mascota p0 = MAKE_OBJECT Mascota()
Mascota p1 = MAKE_OBJECT Mascota()
Mascota p2 = MAKE_OBJECT Mascota()
```

```
p0.asignarSonido("ladrar")
p0.asignarNombre("inti")

p1.asignarSonido("ladrar")
p1.asignarNombre("lucas")

p2.asignarNombre("thor")
p2.asignarSonido("maullar")

p3.asignarNombre("loki")
p3.asignarSonido("maullar")
```

Ejemplo

```
CLASS Animal:
     STRING sonido

     INIT_CLASS():
          sonido = ""

     DESTROY_CLASS()

     STRING vocalizar():
          return sonido

     VOID asignarSonido(STRING s):
          sonido = s
```

```
CLASS Mascota(Animal):
     STRING nombre

     INIT_CLASS():
          nombre = ""

     DESTROY_CLASS()

     VOID asignarNombre(STRING n):
          nombre = n

     STRING verNombre():
          return nombre
```

Se crea una función `mostrarMascota` para mostrar los atributos de ésta: `nombre` y `sonido`

```
VOID mostrarMascota(Mascota m):
    print(m.verNombre())
    print(m.vocalizar())
```

```
p0.asignarSonido("ladrar")
p0.asignarNombre("inti")

mostrarMascota(p0)
```

Salida

```
inti
ladrar
```

Ejemplo

Universidad de Valparaíso CHILE

```
CLASS Animal:
     STRING sonido

     INIT_CLASS():
          sonido = ""

     DESTROY_CLASS()

     STRING vocalizar():
          return sonido

     VOID asignarSonido(STRING s):
          sonido = s
```

```
CLASS Mascota(Animal):
     STRING nombre

     INIT_CLASS():
          nombre = ""

     DESTROY_CLASS()

     VOID asignarNombre(STRING n):
          nombre = n

     STRING verNombre():
          return nombre

     STRING vocalizar():
          return "El sonido de la mascota es:" + sonido
```

Ahora se mantiene la definición de la función, pero la clase `Mascota` **sobreescribe** el método vocalizar.

```
VOID mostrarMascota(Mascota m):
    print(m.verNombre())
    print(m.vocalizar())
```

```
p0.asignarSonido("ladrar")
p0.asignarNombre("inti")

mostrarMascota(p0)
```

Salida

```
inti
El sonido de la mascota es: ladrar
```

Polimorfismo en tiempo de ejecución

Ejemplo

Universidad de Valparaíso CHILE

```
CLASS Animal:
     STRING sonido

     INIT_CLASS():
          sonido = ""

     DESTROY_CLASS()

     STRING vocalizar():
          return sonido

     VOID asignarSonido(STRING s):
          sonido = s
```

Se agregan nuevo constructor que permita crear mascotas con nombre y sonido

```
CLASS Mascota(Animal):
     STRING nombre

     INIT_CLASS():
          nombre = ""

     INIT_CLASS(_nombre, _sonido):
          nombre = _nombre
          asignarSonido(_sonido)

     DESTROY_CLASS()

     VOID asignarNombre(STRING n):
          nombre = n

     STRING verNombre():
          return nombre

     STRING vocalizar():
          return "El sonido de la mascota es:" + sonido
```

```
Mascota p3 = MAKE_OBJECT Mascota("piolín", "trinar")

mostrarMascota(p3)
```

Salida

```
piolín
El sonido de la mascota es: trinar
```

Polimorfismo en tiempo de compilación

Estudiar implementación en C++ y Python (**herencia01.zip**)