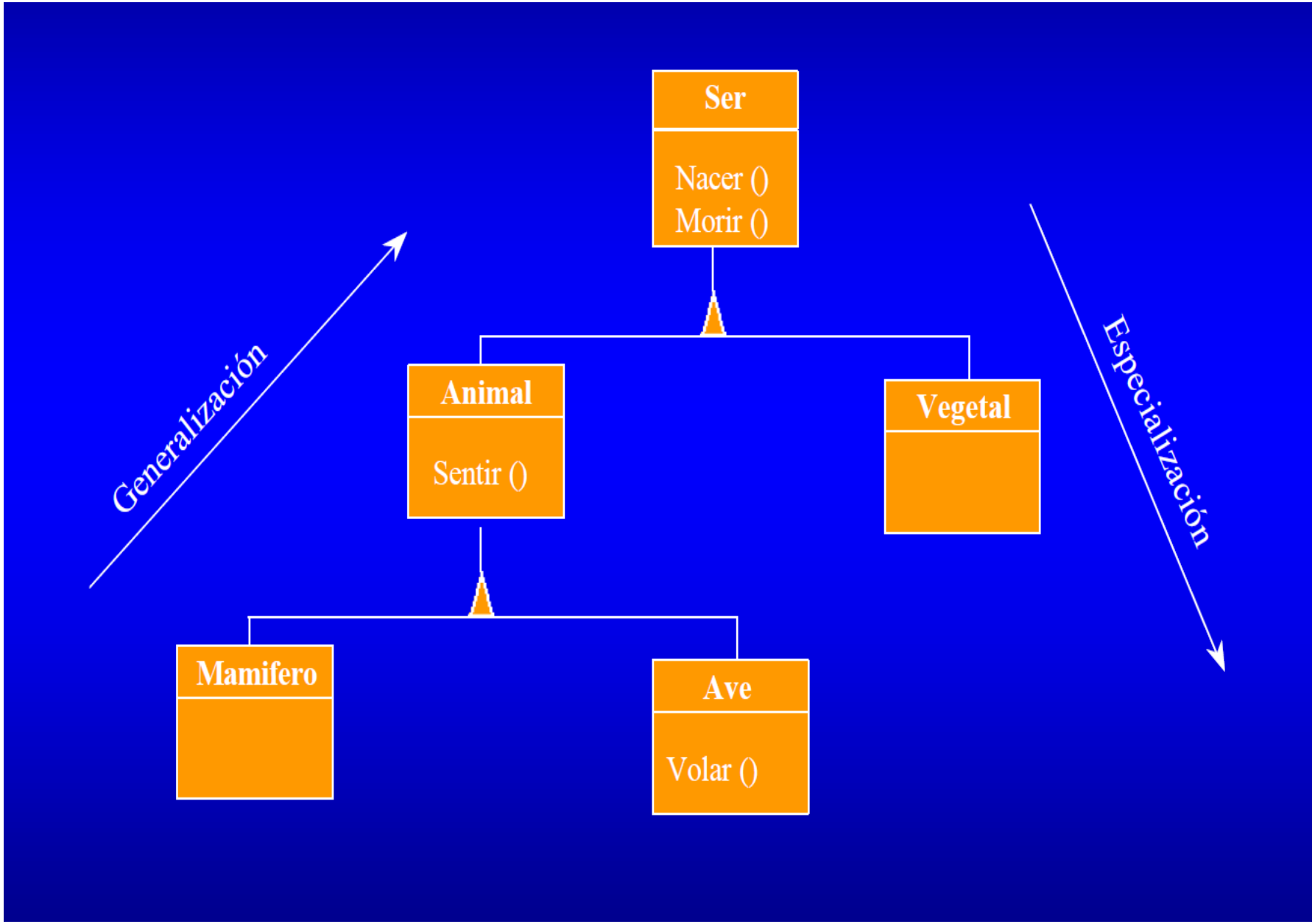
Relación de Herencia

Herencia

- Permite definir una clase hija (subclase) a partir de una clase padre (base, superclase).
- La clase hija hereda el interfaz (con la implementación de las operaciones) y los atributos.
- Relación “es un”.


Ser
Nacer ()
Morir ()
Animal
Sentir ()
Vegetal
Mamifero
Ave
Volar ()
Generalización
Especialización

Herencia

Concepto de tipo: un tipo denota una interfaz particular.

- – Ej.: Ser, Animal, Vegetal, etc.
- Un tipo es un subtipo de otro si su interfaz contiene la interfaz de su supertipo.
- – Ej.: Animal es un subtipo de Ser y un supertipo de Ave y Mamifero.

Clase e interfaz

Diferencia entre los conceptos de clase e interfaz.

– El concepto de clase hace referencia a los atributos y a las operaciones, junto con su implementación.

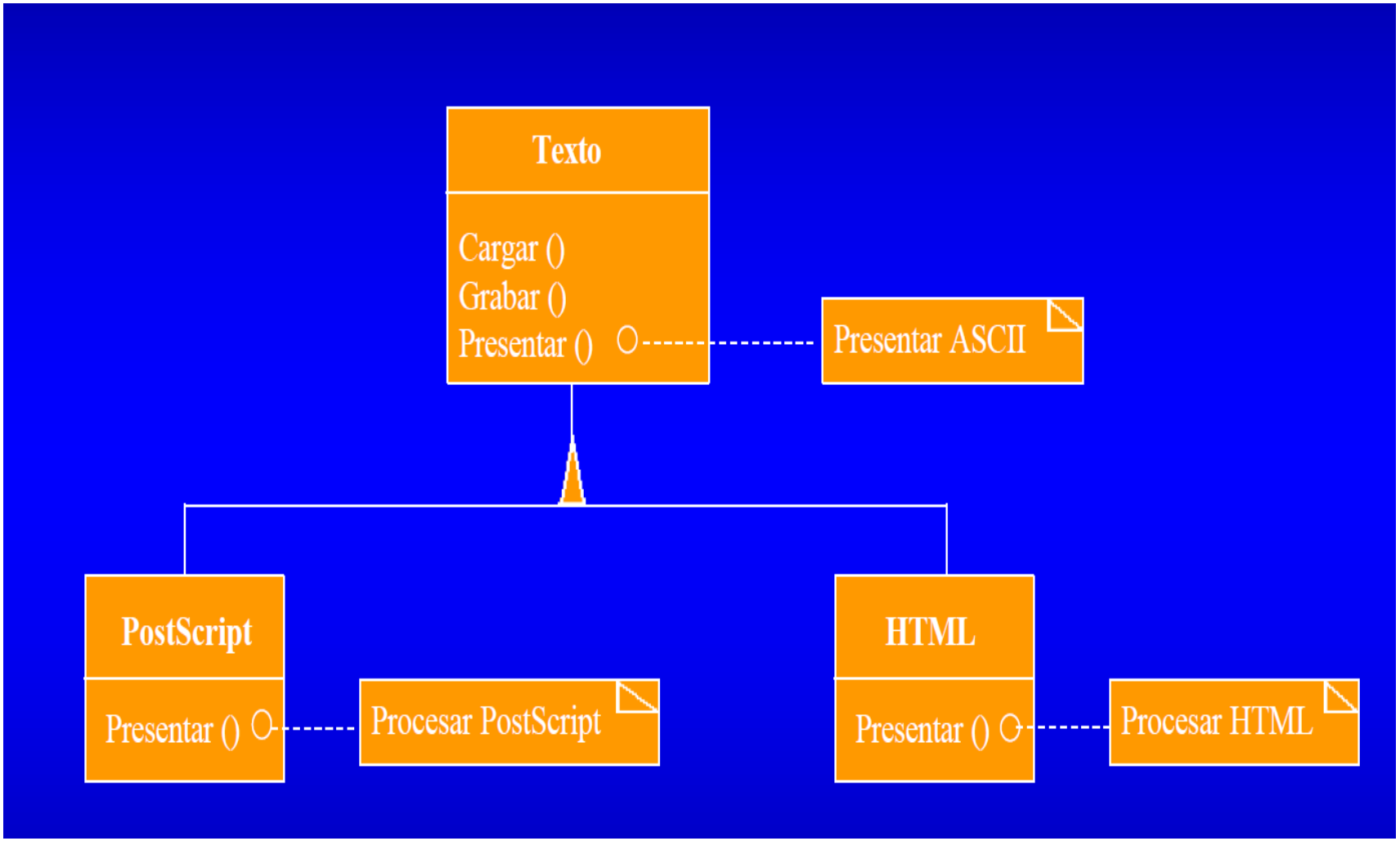
– El concepto de interfaz hace referencia sólo a los prototipos de las operaciones visibles.

Extensión y redefinición

La clase hija puede extender o redefinir (override) el comportamiento de la clase padre.

– Extensión: Se añaden operaciones y/o atributos (ej.: Volar en el caso de Ave).

– Redefinición: Se cambia la implementación de alguna operación heredada
(o se le da una implementación si no la tenía).


Texto
Cargar ()
Grabar ()
Presentar ()
Presentar ASCII
PostScript
Presentar ()
Procesar PostScript
HTML
Presentar ()
Procesar HTML

Clases abstractas

- Clase abstracta

– Deja sin definir una o más operaciones (sólo declara sus prototipos), que se definirán en subclases. Estas operaciones se denominan operaciones abstractas.

– No se pueden crear instancias de una clase abstracta.

– Su objetivo es especificar una interfaz común para todas sus subclases.

- Clase concreta

– Una clase que no es abstracta, es decir, que no define operaciones abstractas y define las operaciones abstractas que hereda.

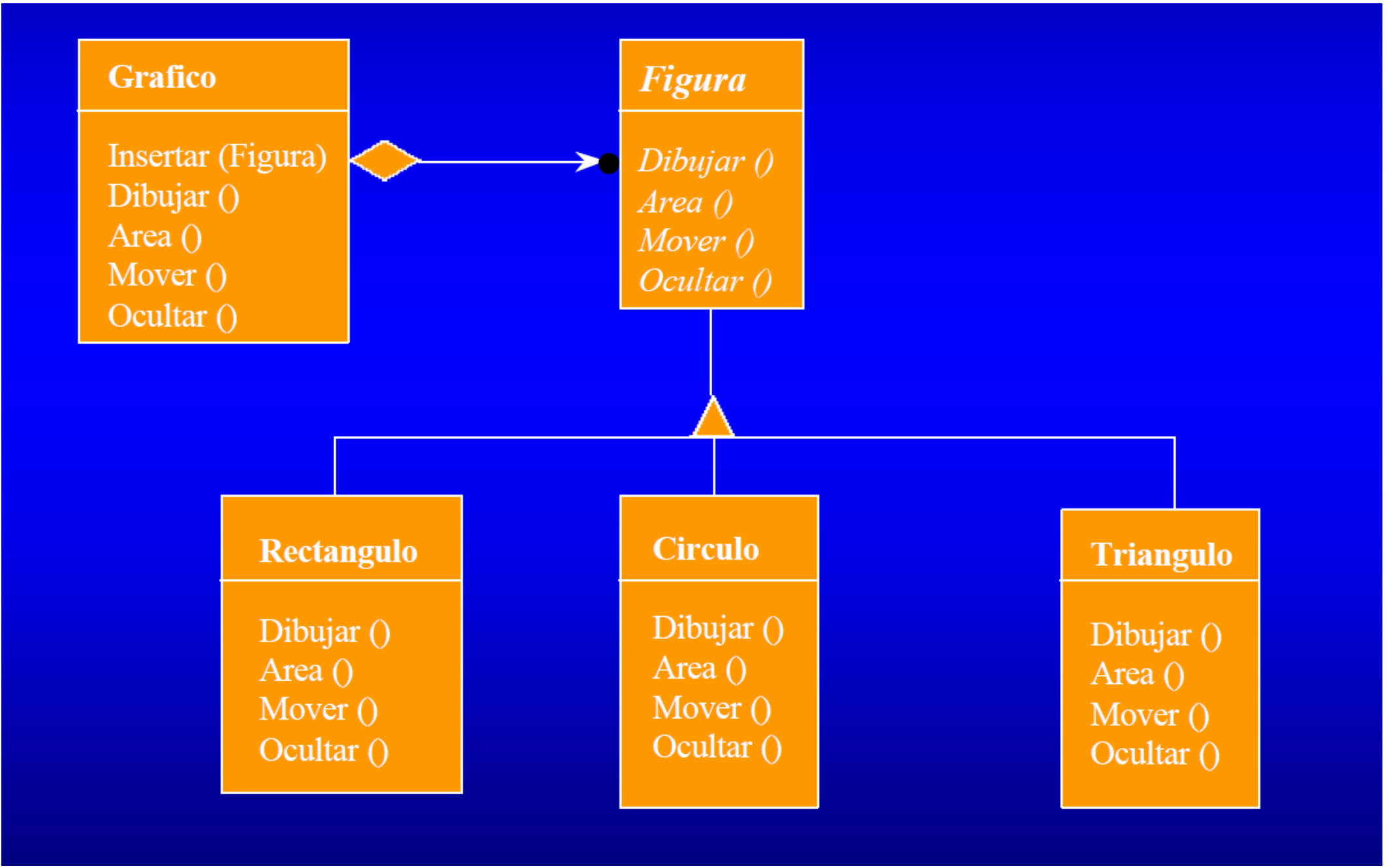


Grafico
Insertar (Figura)
Dibujar ()
Area ()
Mover ()
Ocultar ()
Figura
Dibujar ()
Area ()
Mover ()
Ocultar ()
Rectangulo
Dibujar ()
Area ()
Mover ()
Ocultar ()
Circulo
Dibujar ()
Area ()
Mover ()
Ocultar ()
Triangulo
Dibujar ()
Area ()
Mover ()
Ocultar ()

Polimorfismo

• Polimorfismo: Distintas instancias del mismo tipo interpretan un mismo mensaje de distinta forma.

– El polimorfismo requiere enlace dinámico.

• Enlace dinámico: La llamada a figura.Dibujar se resuelve en tiempo de ejecución.

• Enlace estático: La llamada a grafico.Dibujar se resuelve en tiempo de compilación.

• Consejo: Programar haciendo uso de una interfaz, y no de una

implementación concreta.

Herencia de implementación e interfaz

Herencia de implementación: La clase hija hereda la implementación de métodos de la clase padre.

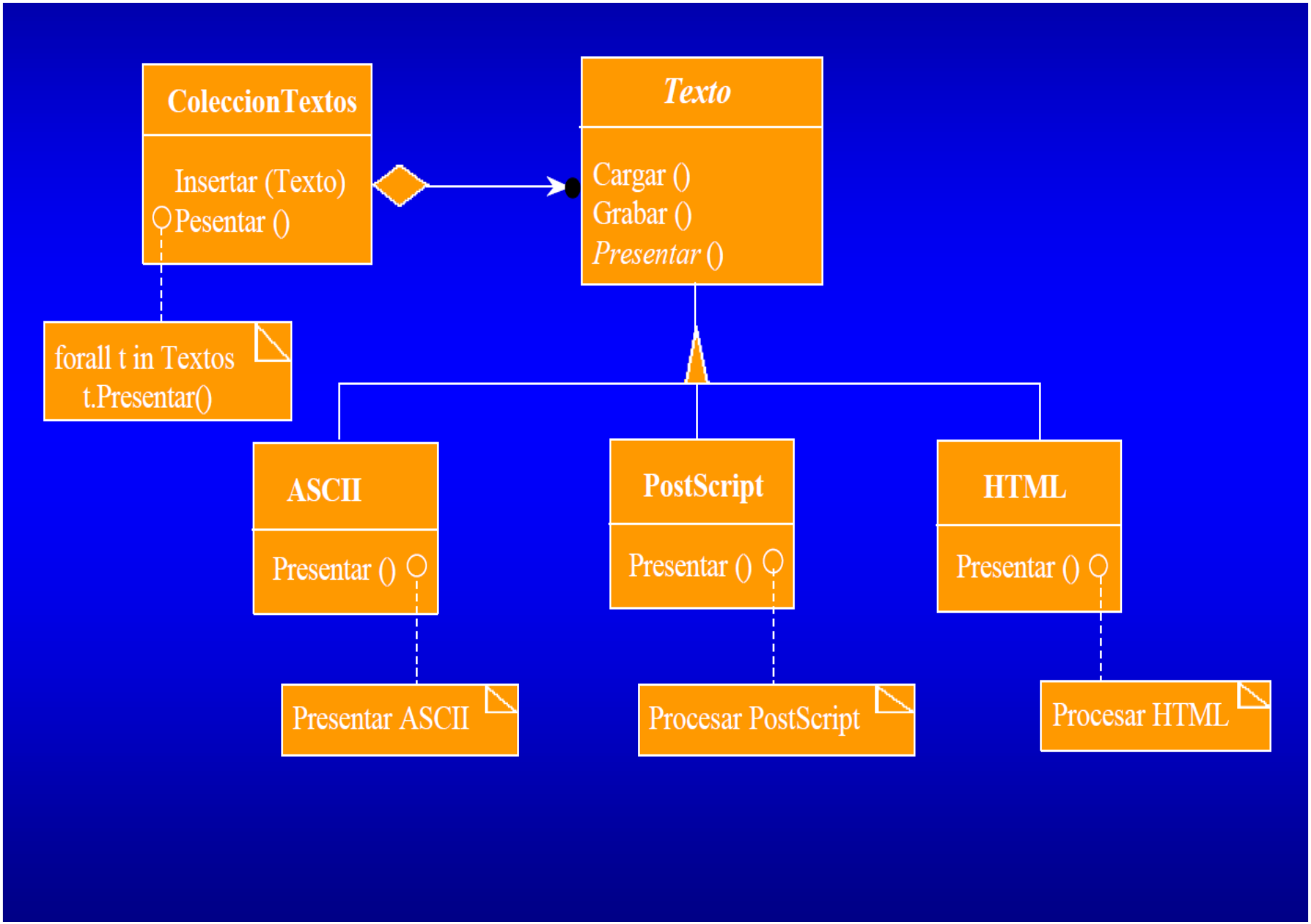
– Ej.: el ejemplo de la jerarquía de los seres vivos.

• Herencia de interfaz: La clase hija hereda el interfaz (pero no la implementación de las operaciones).

– Ej.: el ejemplo de la jerarquía de figuras.

• Algunos lenguajes tienen palabras clave para distinguir entre herencia de interfaz y clase (ej.: Java), mientras que otros no (ej.: C++).

• Herencia de clase: combina la herencia de implementación y de interfaz.

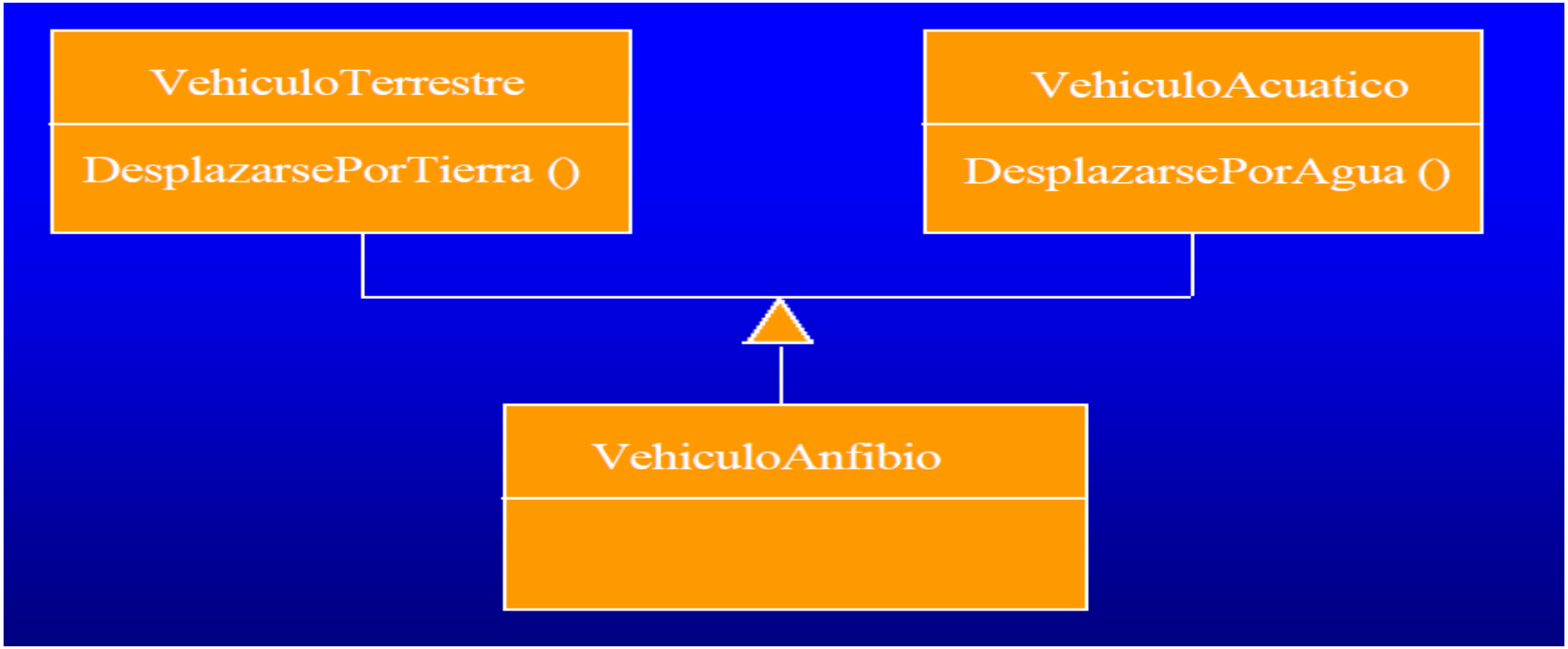

ColeccionTextos
Insertar (Texto)
Pesentar ()
Texto
Cargar ()
Grabar ()
Presentar ()
forall t in Textos
t.Presentar()
ASCII
Presentar ()
PostScript
Presentar ()
HTML
Presentar ()
Presentar ASCII
Procesar PostScript
Procesar HTML

Herencia múltiple

• Herencia múltiple

– una clase hija hereda de dos o más padres.

– No está disponible en todos los lenguajes (disponible en C++; en Java sólo para la herencia de interfaz).



Herencia múltiple

- Problema de ambigüedad: unVehiculoAnfibio.Desplazarse();
- - Los lenguajes ofrecen mecanismos para deshacer la ambigüedad (en C++=> unVehiculoAnfibio.VehiculoTerrestre::Desplazarse()).