



Célula Académica UABC-Live .net

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

<http://uabc-live-net.spaces.live.com/>

Sesión No. 3

Introducción a la Programación Orientada a Objetos

Expositores:

Vizcarra Larios Sonia Adriana (harukanoiki@hotmail.com)

Sánchez Medina Carlos Humberto (c_sanchez15@hotmail.com)

Fecha: 5 de Octubre del 2006

Programa Microsoft Desarrollador Cinco Estrellas

Estrella 0









Objetivo

Describir el Paradigma de Orientación a Objetos incluyendo los conceptos relacionados al análisis, diseño y programación

Prerrequisitos

- ✘ Poseer los conocimientos proporcionados en los siguientes módulos de la Estrella 0:
 - Fundamentos de Programación

Temas a Tratar

-  **Paradigmas de Programación**
-  **Clases y Objetos**
-  **Modificadores de Acceso**
-  **¿Qué es UML?**
-  **Pilares de la Orientación a Objetos**
-  **Conceptos del Diseño Orientado a Objetos**

Paradigmas de Programación

 Hay para todos los gustos

- Estructurados (C, Pascal, Basic, etc.)
- Funcionales (CAML)
- Declarativos (Prolog)
- Orientados a Objetos (C#, VB.NET, Smalltalk, Java)
- Orientados a Aspectos
- Híbridos (Lisp, Visual Basic)
- Incomprensibles....

 Cada enfoque tiene sus ventajas y desventajas

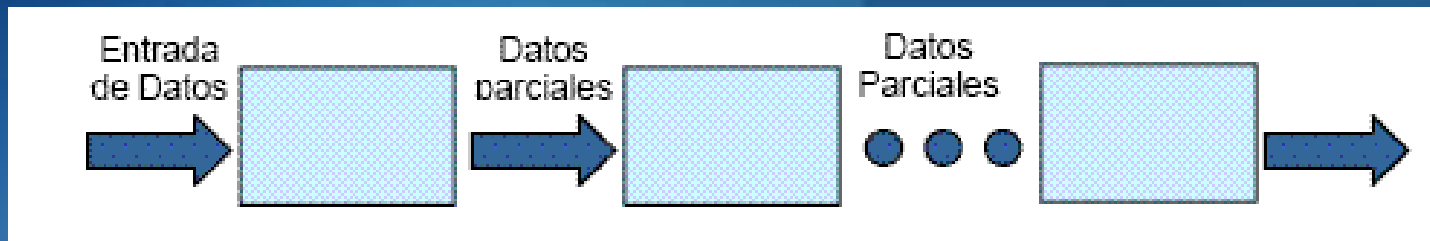
 Cada uno es más apropiado para ciertas cosas

El mundo color de Objetos

- ✘ Todo el mundo está compuesto de entidades que se relacionan e interactúan entre si
- ✘ ¿Qué es un Objeto?
 - Todo es un Objeto ¡¿~?!
- ✘ ¿Es lo mismo de siempre con otro nombre?
 - Pensar en Objetos
- ✘ No es el último grito de la moda (1980s)

El mundo color de Objetos

La Programación estructurada se concentra en las acciones que controlan el flujo de datos.



La POO se centra en la interrelación que existe entre los datos y las acciones.

El mundo color de Objetos



¿Por qué Orientación a Objetos (OO)?

- Se parece más al mundo real
- Permite representar modelos complejos
- Muy apropiada para aplicaciones de negocios
- Las empresas ahora sí aceptan la OO
- Las nuevas plataformas de desarrollo la han adoptado (Java / .NET)

Temas a Tratar

- ✘ Paradigmas de Programación
- ✘ Clases y Objetos
- ✘ Modificadores de Acceso
- ✘ ¿Qué es UML?
- ✘ Pilares de la Orientación a Objetos
- ✘ Conceptos del Diseño Orientado a Objetos

¿Qué es un Objeto?

- ✘ Informalmente, un objeto representa una entidad del mundo real
- ✘ Entidades Físicas
 - (Ej.: Vehículo, Casa, Producto)
- ✘ Entidades Conceptuales
 - (Ej.: Proceso Químico, Transacción Bancaria)
- ✘ Entidades de Software
 - (Ej.: Lista Enlazada, Interfaz Gráfica)

¿Qué es un Objeto?

Definición Formal (Rumbaugh):

- “Un objeto es un concepto, abstracción o cosa con un significado y límites claros en el problema en cuestión”





Un objeto posee (Booch):

- Estado
- Comportamiento
- Identidad

Un objeto posee Estado

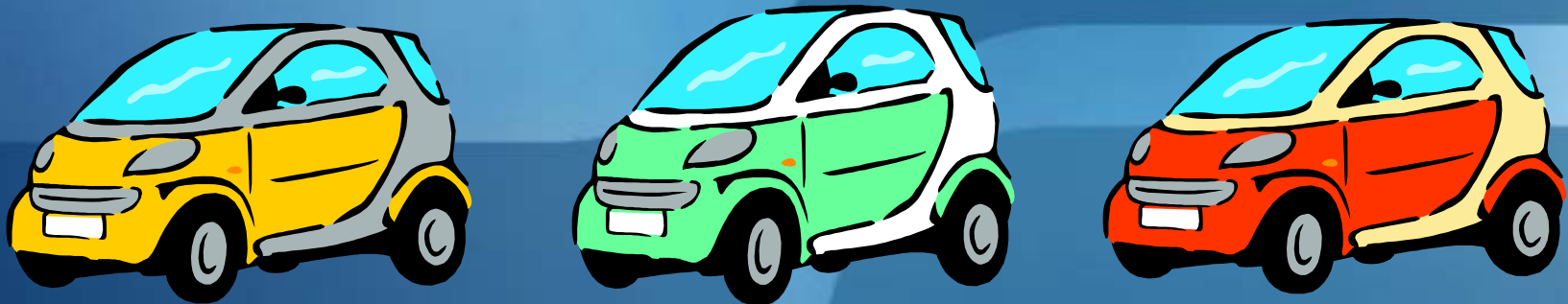
- ❖ *Lo que el objeto sabe*
- ❖ El estado de un objeto es una de las posibles condiciones en que el objeto puede existir
- ❖ El estado normalmente cambia en el transcurso del tiempo
- ❖ El estado de un objeto es implementado por un conjunto de propiedades (atributos), además de las conexiones que puede tener con otros objetos

Un objeto posee Comportamiento

-  *Lo que el objeto puede hacer*
-  El comportamiento de un objeto determina cómo éste actúa y reacciona frente a las peticiones de otros objetos
-  Es modelado por un conjunto de mensajes a los que el objeto puede responder (operaciones que puede realizar)
-  Se implementa mediante métodos

Un objeto posee Identidad

- ✘ Cada objeto tiene una identidad única, incluso si su estado es idéntico al de otro objeto

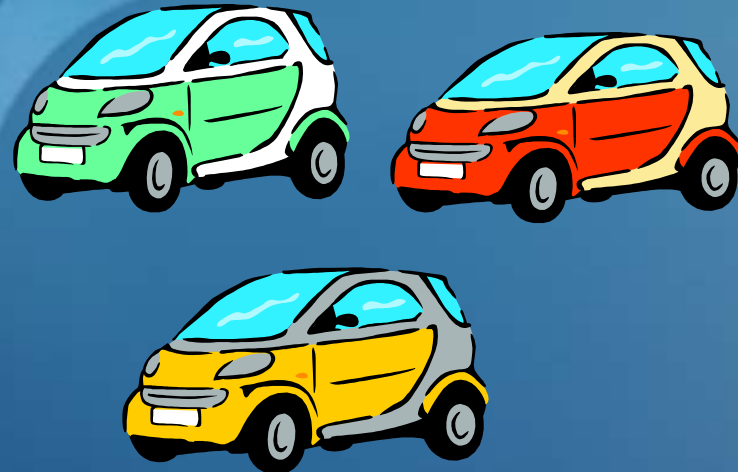
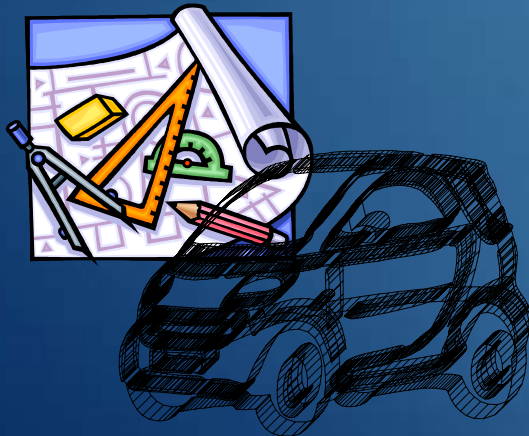


¿Qué es una Clase?

- ✘ Una clase es una descripción de un grupo de objetos con:
 - Propiedades en común (atributos)
 - Comportamiento similar (operaciones)
 - La misma forma de relacionarse con otros objetos (relaciones)
 - Una semántica en común (significan lo mismo)
- ✘ Una clase es una abstracción que:
 - Enfatiza las características relevantes
 - Suprime otras características (simplificación)
- ✘ Un objeto es una instancia de una clase


Objetos y Clases

- ✘ Una clase es una definición abstracta de un objeto
 - Define la estructura y el comportamiento compartidos por los objetos
 - Sirve como modelo para la creación de objetos
- ✘ Los objetos pueden ser agrupados en clases



Ejemplo de una Clase

 Clase: Curso

 Estado (Atributos)

- Nombre
- Ubicación
- Días Ofrecidos
- Horario de Inicio
- Horario de Término

 Comportamiento (Métodos)

- Agregar un Alumno
- Borrar un Alumno
- Entregar un Listado del Curso
- Determinar si está Completo

Temas a Tratar

- ✘ Paradigmas de Programación
- ✘ Clases y Objetos
- ✘ **Modificadores de Acceso**
- ✘ ¿Qué es UML?
- ✘ Pilares de la Orientación a Objetos
- ✘ Conceptos del Diseño Orientado a Objetos

Modificadores de Acceso

- ✘ Permiten definir el nivel de acceso (visibilidad) de los miembros (atributos o métodos) de una clase
 - Público: Cualquier clase puede “ver” los miembros públicos de otra clase
 - Privado: Sólo la clase puede ver sus propios miembros privados
- ✘ Existen otros dos modificadores para propósitos específicos (Paquete, Protegido)

Temas a Tratar

- ✘ Paradigmas de Programación
- ✘ Clases y Objetos
- ✘ Modificadores de Acceso
- ✘ ¿Qué es UML?
- ✘ Pilares de la Orientación a Objetos
- ✘ Conceptos del Diseño Orientado a Objetos

¿Qué es UML?




- ✚ “UML es un lenguaje visual para especificar, construir y documentar sistemas” (OMG - Object Management Group)
- ✚ Unified (UNIFICADO):
 - El aporte de muchos métodos y notaciones
 - Independiente de implementaciones, plataformas y lenguajes
- ✚ Modeling (MODELADO):
 - Los modelos son utilizados en todas las ingenierías
- ✚ Language (LENGUAJE):
 - Si hay gente, requieren comunicarse. Si se tienen que comunicar, se tienen que entender. Para entenderse necesitan un lenguaje común
- ✚ ¡UML no es Metodología!

Una Clase en UML

 Una clase está compuesta de tres secciones

- La primera sección contiene el nombre de la clase
- La segunda sección muestra la estructura (atributos)
- La tercera sección muestra el comportamiento (operaciones)

 La segunda y la tercera sección pueden ser suprimidas

 Modificadores de Acceso

- Los miembros públicos se denotan con el signo “+”
- Los miembros privados se denotan con el signo “-”

Curso
-nombre -ubicacion -dias -inicio -fin
+AgregarAlumno() +BorrarAlumno() +GenerarListadoCurso() +EstaCompleto()

Temas a Tratar

- ✘ Paradigmas de Programación
- ✘ Clases y Objetos
- ✘ Modificadores de Acceso
- ✘ ¿Qué es UML?
- ✘ **Pilares de la Orientación a Objetos**
- ✘ Conceptos del Diseño Orientado a Objetos

Pilares de la Orientación a Objetos

Abstracción

Relaciones

Herencia

Encapsulamiento

Abstracción

Ignorancia Selectiva

- La abstracción nos ayuda a trabajar con cosas complejas
- Se enfoca en lo importante
- Ignora lo que no es importante (simplifica)

Una clase es una abstracción en la que:

- Se enfatizan las características relevantes
- Se suprimen otras características

Una clase debe capturar una y solo una abstracción clave


Encapsulamiento

- ❖ Principio que establece que los atributos propios de un objeto no deben ser visibles desde otros objetos
 - Deben ser declarados como privados
- ❖ Permite abstraer al resto del mundo de la complejidad de la implementación interna
- ❖ Permite exponer el estado del objeto sólo a través del comportamiento que le hayamos definido mediante miembros públicos
- ❖ ¿Por qué es útil?
 - Punto de Control/Validación
 - Mejor respuesta ante los Cambios

Relaciones

- ❖ Todo sistema abarca muchas clases y objetos
- ❖ Los objetos contribuyen en el comportamiento de un sistema colaborando entre si
 - La colaboración se logra a través de las relaciones
- ❖ Existen dos tipos principales de relaciones
 - Asociación
 - Agregación

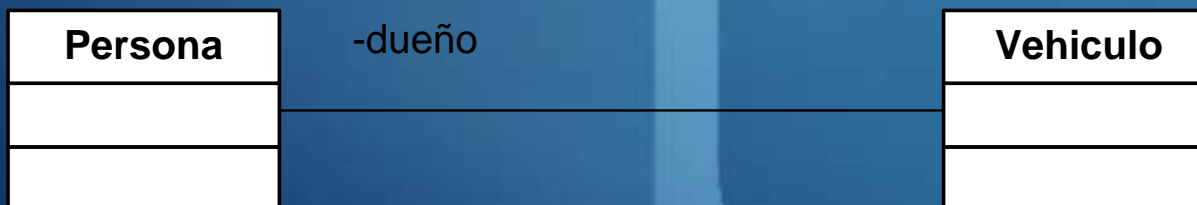
Relaciones de Asociación

 Una asociación es una conexión entre dos clases que representa una comunicación

- Una asociación puede tener nombre
- La comunicación puede ser tanto uni como bi-direccional (por defecto)
- La multiplicidad es el número de instancias que participan en una asociación

 Ejemplo:

- Una Persona es Dueña de un Vehículo
- Un Vehículo Pertenece a una Persona



Relaciones de Agregación

- ❖ La agregación es una forma especial de asociación donde un todo se relaciona con sus partes
 - También se conoce como “una parte de” o una relación de contención

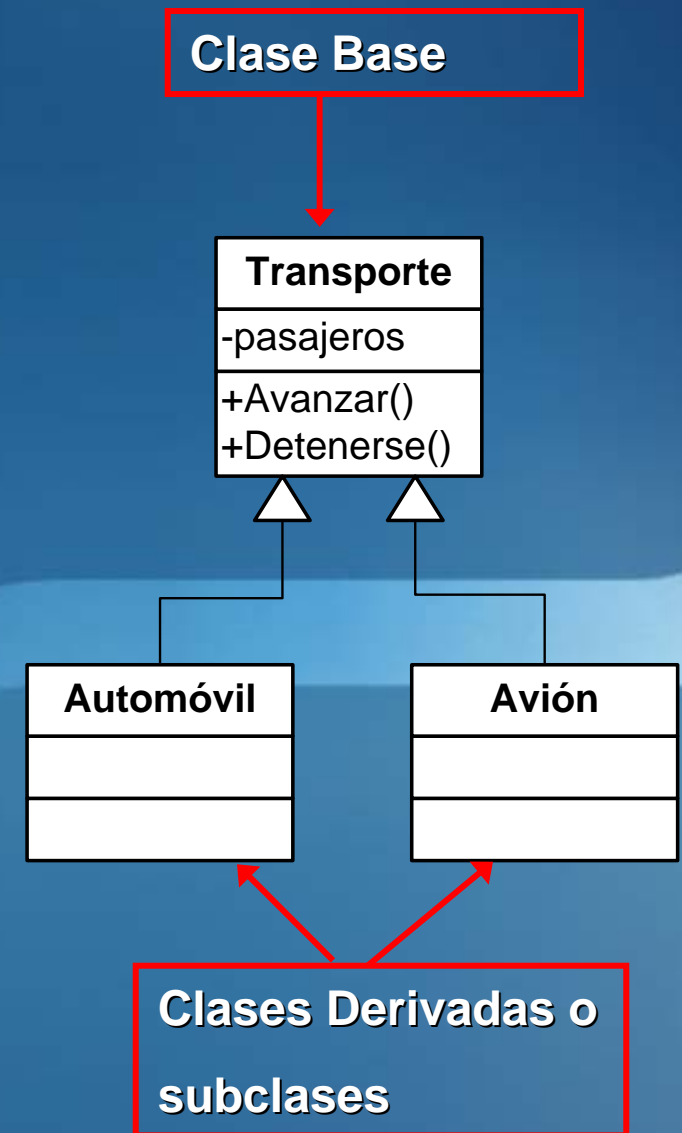
❖ Ejemplo:

- Una Puerta es una parte de un Vehículo
- El Vehículo es azul, la Puerta es Azul
- Mover el Vehículo implica mover la Puerta




Herencia

- ✘ Es una relación entre clases en la cual una clase comparte la estructura y comportamiento definido en otra clase (Grady Booch)
- ✘ Cada clase que hereda de otra posee:
 - Los atributos de la clase base además de los propios
 - Soporta todos o algunos de los métodos de la clase base
- ✘ Una subclase hereda de una clase base









Herencia

 Herencia “Es-Un”: herencia real, donde la subclase es un tipo específico de la superclase

- Un Cuadrado es un Rectángulo
- Un perro es un mamífero
- Un automóvil es un vehículo a motor

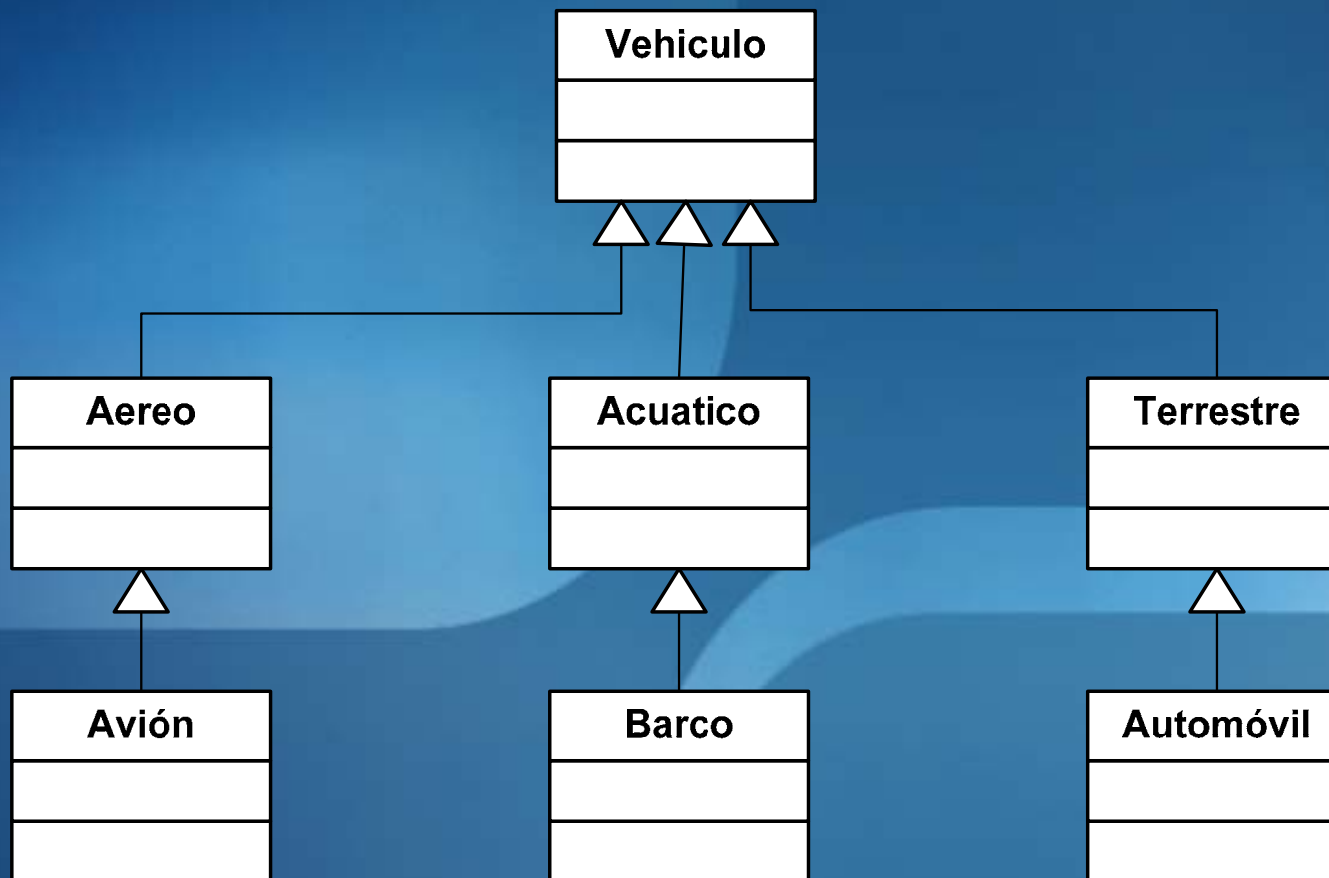
Temas a Tratar

-  Paradigmas de Programación
-  Clases y Objetos
-  Modificadores de Acceso
-  ¿Qué es UML?
-  Principios de la Orientación a Objetos
-  **Conceptos del Diseño Orientado a Objetos**

Interfaces (1/3)

- ❖ Recurso de diseño soportado por los lenguajes orientados a objetos que permite definir comportamiento
- ❖ Permite que clases que no están estrechamente relacionadas entre sí deban tener el mismo comportamiento
- ❖ La implementación de una interfaz es un contrato que obliga a la clase a implementar todos los métodos definidos en la interfaz

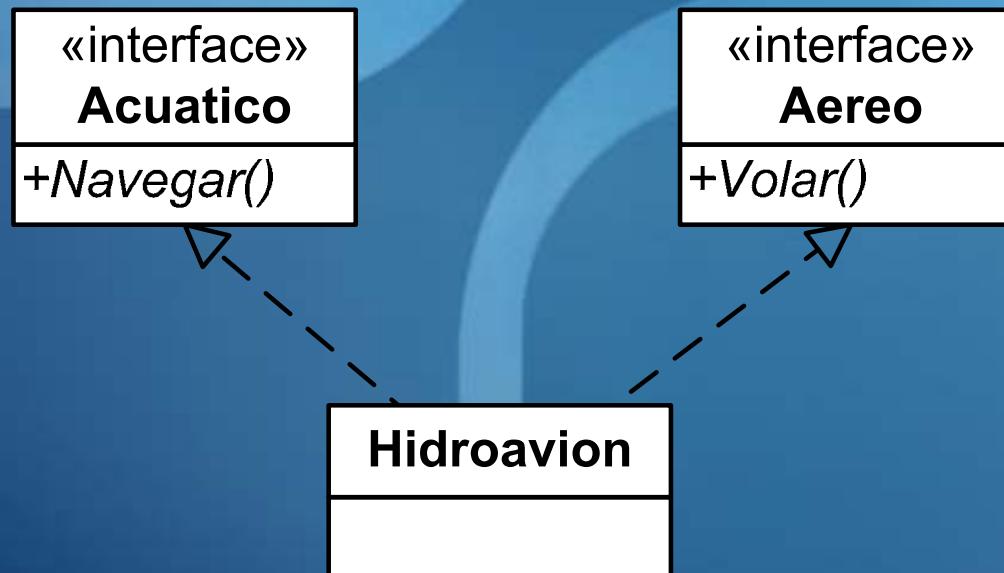
Interfaces (2/3)



¿ De que clase heredaría la clase Hidroavión ?

Interfaces (3/3)

- ❖ Se crean las interfaces que definen comportamiento
- ❖ Hidroavión deberá definir los comportamientos de cada una de las interfaces que implemente

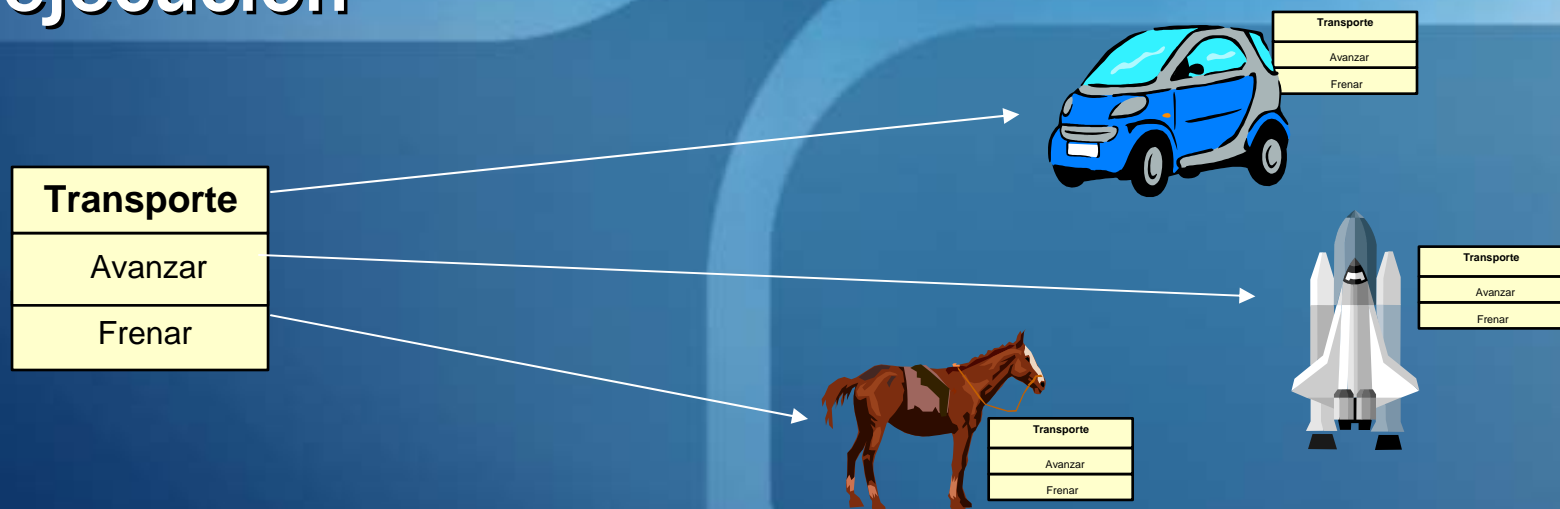


Polimorfismo

- ✘ Es la propiedad que tienen los objetos de permitir invocar genéricamente un comportamiento (método) cuya implementación será delegada al objeto correspondiente recién en tiempo de ejecución
- ✘ El polimorfismo tiende a existir en las relaciones de herencia, pero no siempre es así

Polimorfismo - Ejemplo

- ❖ La definición del método reside en la clase base
- ❖ La implementación del método reside en la clase derivada
- ❖ La invocación es resuelta al momento de ejecución



Microsoft®



© 2005 Microsoft Corporation. All rights reserved.

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.