

Programación II

C

Clase 10

Maximiliano Neiner

Temas a Tratar

- ❖ Polimorfismo
- ❖ Clases Abstractas

Temas a Tratar

Polimorfismo

- ¿Qué es?
- Virtual y Override.

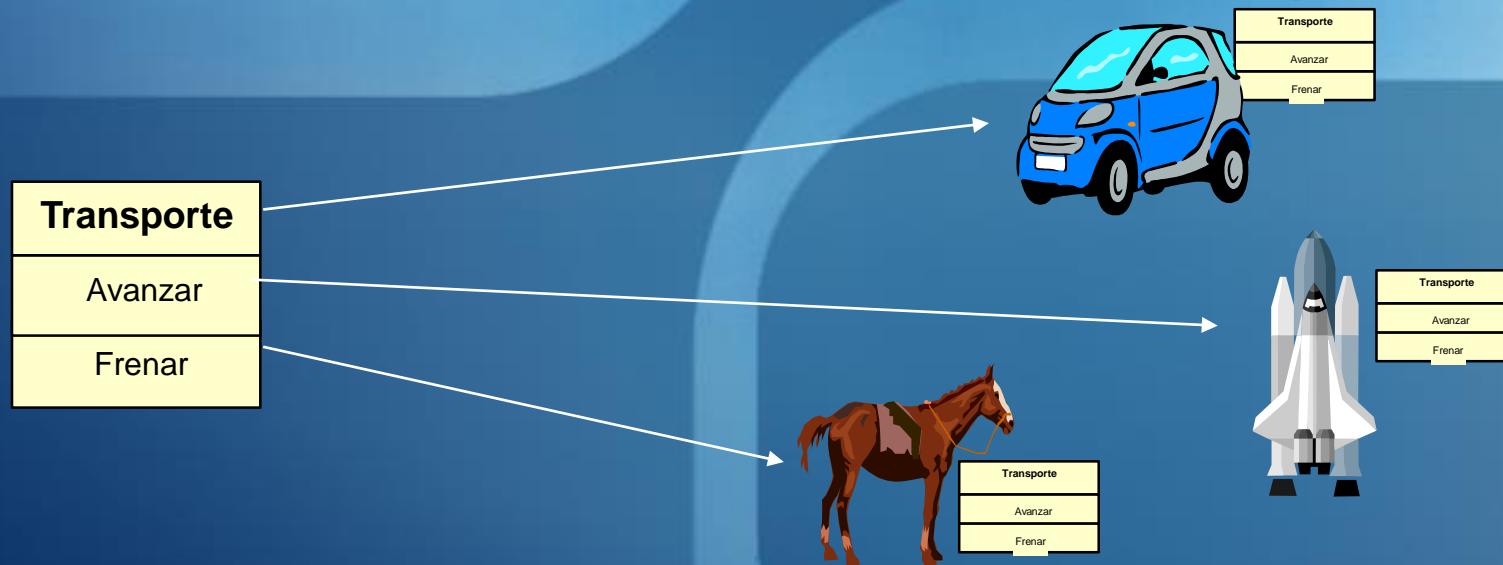
Clases Abstractas

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.
- ❖ En otras palabras, es la capacidad de tratar objetos diferentes de la misma forma.
- ❖ El polimorfismo tiende a existir en las relaciones de herencia.
- ❖ El polimorfismo basado en herencia implica la definición de métodos en una clase base y sobrescribirlos con nuevas implementaciones en clases derivadas.

Polimorfismo - Ejemplo

- ❖ La definición del método reside en la clase base.
 - Se utiliza ***virtual***.
- ❖ La implementación del método reside en la clase derivada.
 - Se utiliza ***override***.
- ❖ La invocación es resuelta en tiempo de ejecución.



Temas a Tratar

Polimorfismo

- ¿Qué es?
- Virtual y Override.

Clases Abstractas

Virtual y Override

- ❖ El polimorfismo implica la definición de métodos y/o propiedades en una clase base, mediante el uso de la palabra reservada **virtual** y sobrescribirlos con nuevas implementaciones en clases derivadas con la palabra reservada **override**.
- ❖ En la clase base:

```
[modificadores] virtual tipoRetorno NombreMétodo([Args])  
{ // Implementación del método }
```

- ❖ En la clase derivada:

```
[modificadores] override tipoRetorno NombreMétodo([Args])  
{ // Sobrescritura del método }
```

Ejemplo – Método virtual

```
class ClaseBase
{
    public virtual void NombreMétodo()
    {
        // Implementación del método
    }
}

class ClaseDerivada : ClaseBase
{
    public override void NombreMétodo()
    {
        base.NombreMétodo(); // Llamada al método virtual
        // Implementación específica
    }
}
```

Ejemplo - Propiedad virtual

```
class ClaseBase {
    public virtual string NombreProp
    {
        get { // Implementación }
        set { // Implementación }
    }
}

class ClaseDerivada : ClaseBase {
    public override string NombreProp
    {
        get { // Implementación }
        set { // Implementación }
    }
}
```

Demo

Temas a Tratar

▣ **Polimorfismo**

▣ **Clases Abstractas**

- **Introducción.**
- **Declaración de Clases Abstractas.**
- **Diferencias con Clases no Abstractas**
- **Métodos Abstractos.**
- **Propiedades Abstractas.**
- **Uso de Métodos Abstractos.**
- **Diferencias entre Métodos Virtuales y Métodos Abstractos.**

Clases Abstractas (1/2)

- ❖ Son un tipo especial de clases básicas.
- ❖ Además de miembros de clase normales, poseen miembros de clase *abstractos*, que son métodos y propiedades que se declaran sin implementación.
- ❖ Todas las clases que derivan (o heredan) directamente de clases abstractas deben implementar esos métodos y propiedades abstractos.
- ❖ Las clases abstractas **NUNCA** pueden ser instanciadas.

Clases Abstractas (2/2)

- ❖ Las clases abstractas se sitúan en la cima de la jerarquía de clases.
- ❖ Establecen la estructura y significado del código.
- ❖ Facilitan la creación de marcos de trabajo, esto es posible ya que las clases abstractas poseen una información y un comportamiento común a todas las clases derivadas de un marco de trabajo.

Temas a Tratar

- ❖ **Polimorfismo**
- ❖ **Clases Abstractas**
 - Introducción.
 - Declaración de Clases Abstractas.
 - Diferencias con Clases no Abstractas
 - Métodos Abstractos.
 - Propiedades Abstractas.
 - Uso de Métodos Abstractos.
 - Diferencias entre Métodos Virtuales y Métodos Abstractos.

Sintaxis

- Para declarar una clase abstracta se utiliza la palabra reservada **abstract**.

```
[modificadores] abstract class NombreClase
```

- Las reglas que rigen el uso de una clase abstracta son prácticamente las mismas que se aplican a una clase no abstracta:
 - Herencia simple
 - Múltiples derivaciones
 - Mismos modificadores de accesibilidad
 - Etc.

Temas a Tratar

- ❖ **Polimorfismo**
- ❖ **Clases Abstractas**
 - Introducción.
 - Declaración de Clases Abstractas.
 - Diferencias con Clases no Abstractas.
 - Métodos Abstractos.
 - Propiedades Abstractas.
 - Uso de Métodos Abstractos.
 - Diferencias entre Métodos Virtuales y Métodos Abstractos.

Diferencias

- ☒ No está permitido crear una instancia de una clase abstracta.
En este sentido, las clases abstractas son como las interfaces.
- ☒ Se puede declarar un miembro abstracto en una clase abstracta, pero no en una que no lo sea.

Temas a Tratar

- ❖ **Polimorfismo**
- ❖ **Clases Abstractas**
 - Introducción.
 - Declaración de Clases Abstractas.
 - Diferencias con Clases no Abstractas.
 - **Métodos Abstractos.**
 - Propiedades Abstractas.
 - Uso de Métodos Abstractos.
 - Diferencias entre Métodos Virtuales y Métodos Abstractos.

Métodos Abstractos

- Para declarar un método abstracto hay que añadir el modificador **abstract** a la declaración del método.

```
[modificadores] abstract tipoRetorno NombreMétodo([Args]);
```

- Sólo clases abstractas pueden declarar métodos abstractos.
- Nótese que los métodos abstractos no tienen implementación alguna y finalizan con (;).

Temas a Tratar

▣ **Polimorfismo**

▣ **Clases Abstractas**

- Introducción.
- Declaración de Clases Abstractas.
- Diferencias con Clases no Abstractas.
- Métodos Abstractos.
- Propiedades Abstractas.
- Uso de Métodos Abstractos.
- Diferencias entre Métodos Virtuales y Métodos Abstractos.

Propiedades Abstractas

- Para declarar una propiedad abstracta hay que añadir el modificador **abstract** a la declaración.

```
[modificadores] abstract tipoRetorno NombreProp {  
    get;  
    set;  
}
```

- Sólo clases abstractas pueden declarar propiedades abstractas.
- Nótese que las propiedades abstractas no tienen implementación alguna y finalizan con (;).

Temas a Tratar

▣ **Polimorfismo**

▣ **Clases Abstractas**

- Introducción.
- Declaración de Clases Abstractas.
- Diferencias con Clases no Abstractas.
- Métodos Abstractos.
- Propiedades Abstractas.
- Uso de Métodos Abstractos.
- Diferencias entre Métodos Virtuales y Métodos Abstractos.

Uso de Métodos Abstractos

- ❖ Cuando una clase derivada hereda un método abstracto, lo debe implementar obligatoriamente.
- ❖ Para ello se utiliza la palabra reservada **override**.
 - Esto indica que se va a sobrescribir el método.
- ❖ Por definición los métodos abstractos son **virtuales**, pero no pueden estar marcados como **virtuales** de forma explícita.
- ❖ Un método declarado como **virtual** indica que puede ser sobrescrito parcial o totalmente.
 - Al ser implícitamente **virtuales**, es posible sustituir métodos abstractos en clases derivadas.

Ejemplo

```
abstract class ClaseAbstracta
{
    public abstract void Mostrar();
}

class ClaseDerivada : ClaseAbstracta
{
    public override void Mostrar()
    {
        // implementación del
        // método
    }
}
```

Temas a Tratar

- ❖ **Polimorfismo**
- ❖ **Clases Abstractas**
 - Introducción.
 - Declaración de Clases Abstractas.
 - Diferencias con Clases no Abstractas.
 - Métodos Abstractos.
 - Propiedades Abstractas.
 - Uso de Métodos Abstractos.
 - Diferencias entre Métodos Virtuales y Métodos Abstractos.

Métodos Abstractos vs. Métodos Virtuales

- ❖ Los métodos abstractos deben estar declarados en clases abstractas.
- ❖ Los métodos virtuales poseen implementación en la clase base, los abstractos **NO** deben ser implementados en la clase base.
- ❖ Es necesario implementar un método abstracto, en su clase derivada, pero no es así con un método virtual.

Demo