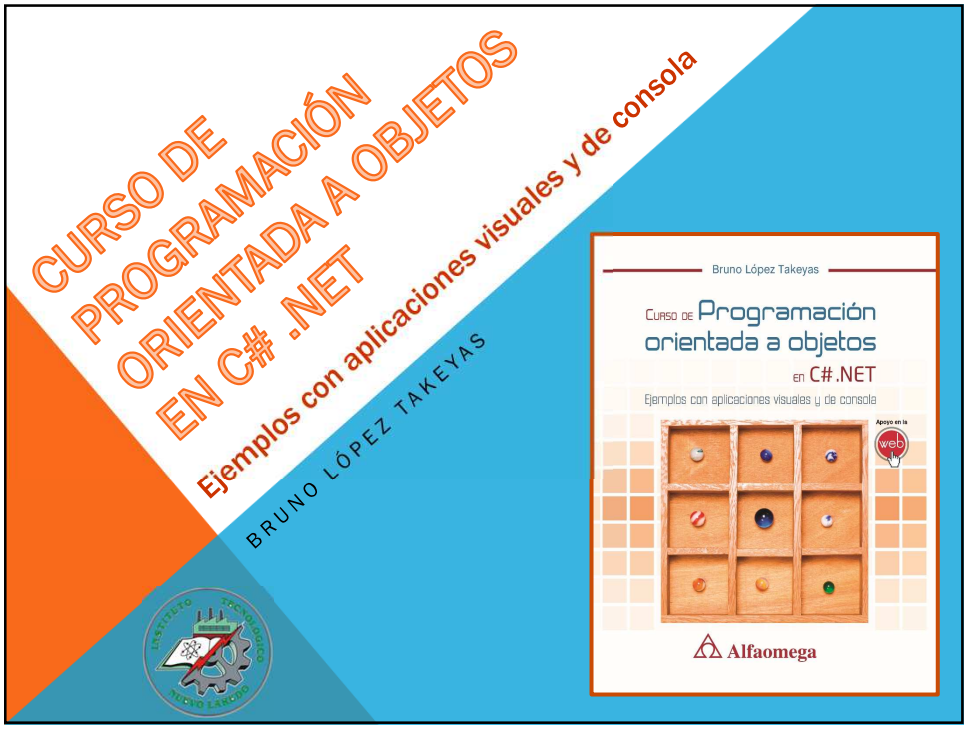

CURSO DE PROGRAMACIÓN ORIENTADA A OBJETOS EN C# .NET
Ejemplos con aplicaciones visuales y de consola
BRUNO LÓPEZ TAKEYAS
Bruno López Takeyas
Curso de Programación orientada a objetos en C#.NET
Ejemplos con aplicaciones visuales y de consola
Alfaomega

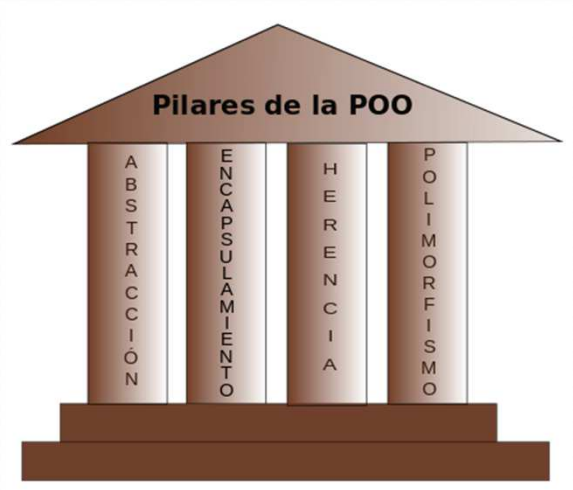

CAPÍTULO 7
POLIMORFISMO
2

Preguntas detonadoras

- ¿Qué es y para qué sirve el polimorfismo?
- ¿Qué ventajas ofrece una aplicación polimórfica?
- ¿Qué conceptos debo dominar para implementar polimorfismo?
- ¿Cuántos tipos de polimorfismo existen?
- ¿Cualquier método definido en una clase base puede sobrescribirse en sus clases derivadas para provocar comportamiento polimórfico?
- ¿Cuáles son las diferencias entre un método virtual, uno abstracto y uno sobrescrito?

3

Pilares de la POO



4

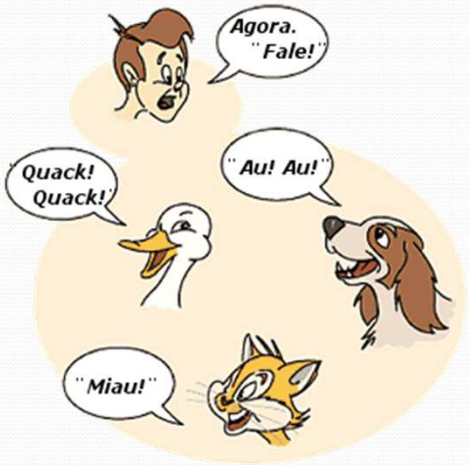
Polimorfismo

- Es la habilidad que poseen los objetos para reaccionar de modo diferente ante los mismos mensajes.
- El *polimorfismo* se refiere a la posibilidad de definir múltiples clases con funcionalidad diferente, pero con métodos o propiedades denominados de forma idéntica, que pueden utilizarse de manera intercambiable mediante código cliente en tiempo de ejecución.
- En C# el polimorfismo está íntimamente relacionado con la sobrecarga y la sobrescritura.

5

Polimorfismo

- Son comportamientos diferentes asociados a distintos objetos
- Comparten el mismo nombre
- Pero ejecutan diferentes acciones
- Depende el objeto que lo invoque o ejecute
- P. ejem. La acción de `Gritar()`



6

Conceptos relacionados con polimorfismo

- Sobrecarga (overload)
- Herencia
- Sobrescritura (override)

7

Sobrecarga [Overload]

- La sobrecarga representa diferentes maneras de realizar una misma acción.
- En los programas se usa el mismo nombre en diferentes métodos con diferentes firmas [número, orden y tipo de los parámetros].
- El código de programación asociado a cada sobrecarga puede variar.
- Ejemplos:
  - `miEmpleado.Contratar("Juan", "Ventas", 2500);`
  - `miEmpleado.Contratar("Juan");`
  - `miEmpleado.Contratar("Juan", 2500);`

8

Ejemplo de Sobrecarga [Overload]

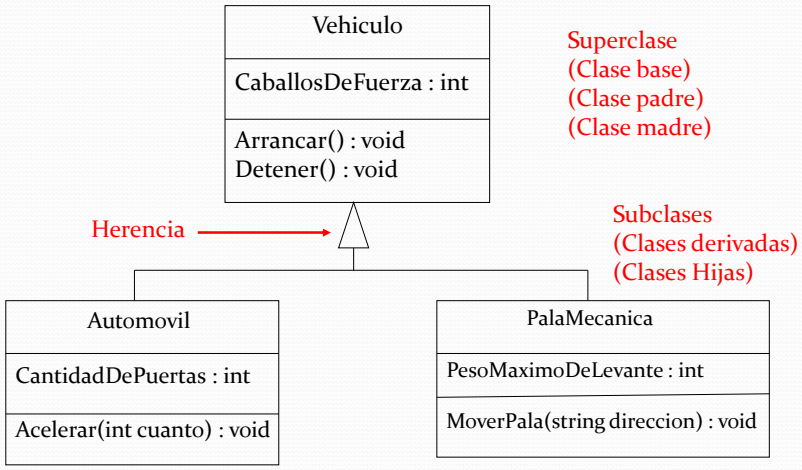
miPuerta.Abrir (Adentro, Afuera)

miPuerta.Abrir (Afuera, Adentro)

miPuerta.Abrir ( )

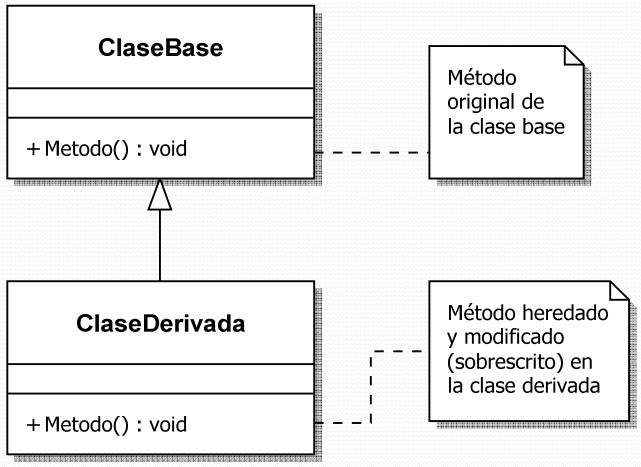
9

Herencia



10

Herencia y sobrescritura



11

Sobrescritura [Overriding]

- Sucede cuando una clase "B" hereda comportamiento de una clase "A", pero la clase "B" re-define este comportamiento
- Propiedades y métodos pueden heredarse de una superclase. Si estas propiedades y métodos son re-definidos en la clase derivada, se dice que han sido "sobrescritos".

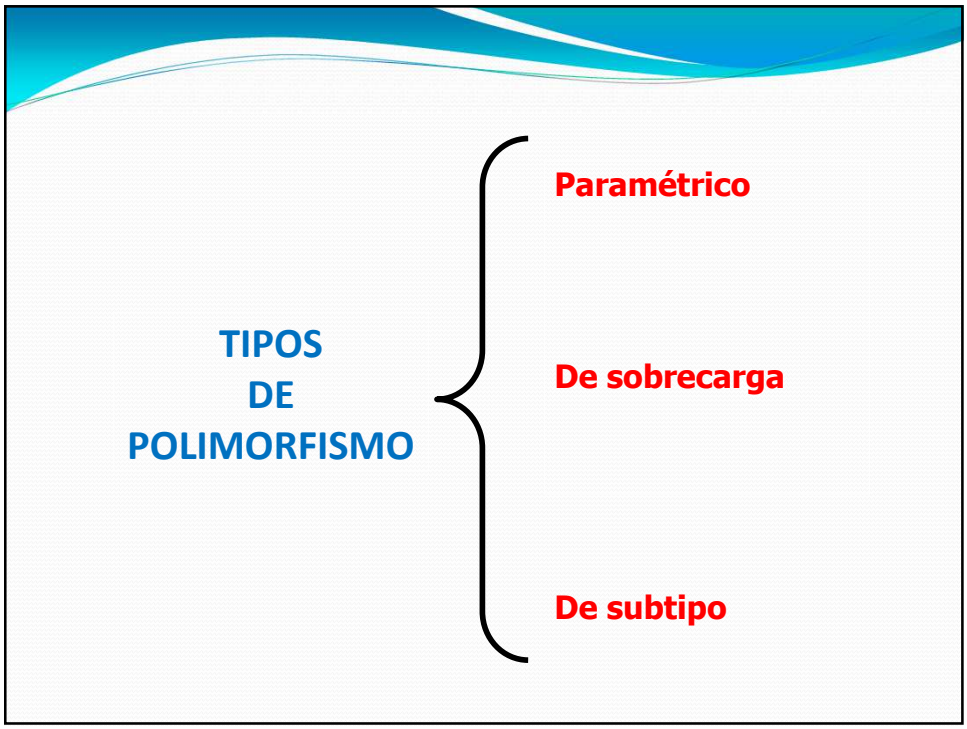
12

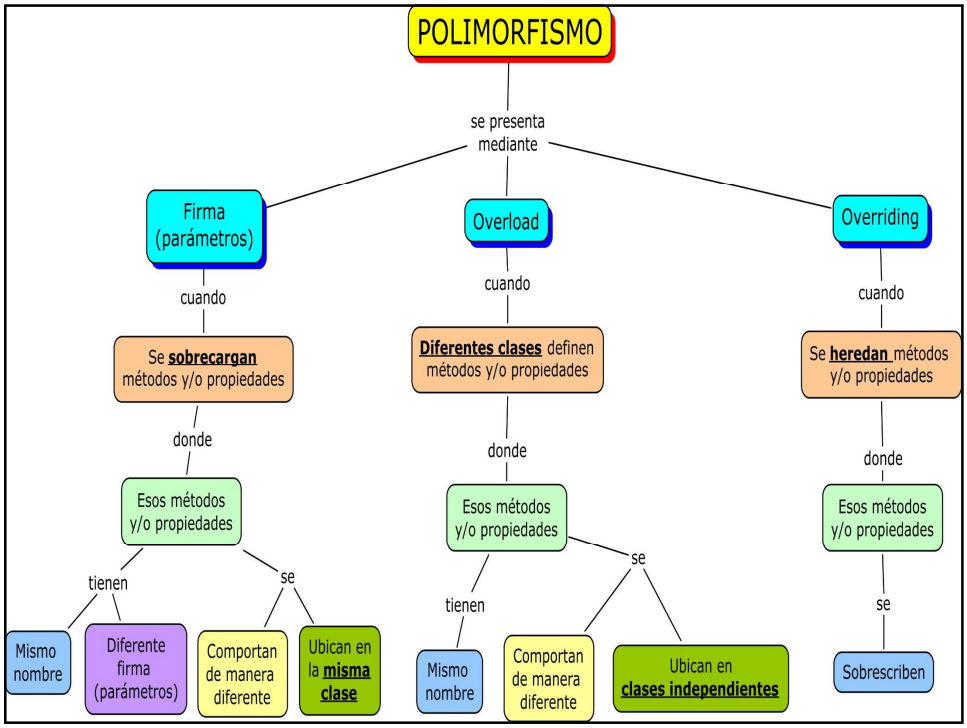
Sobrescritura [Overriding]

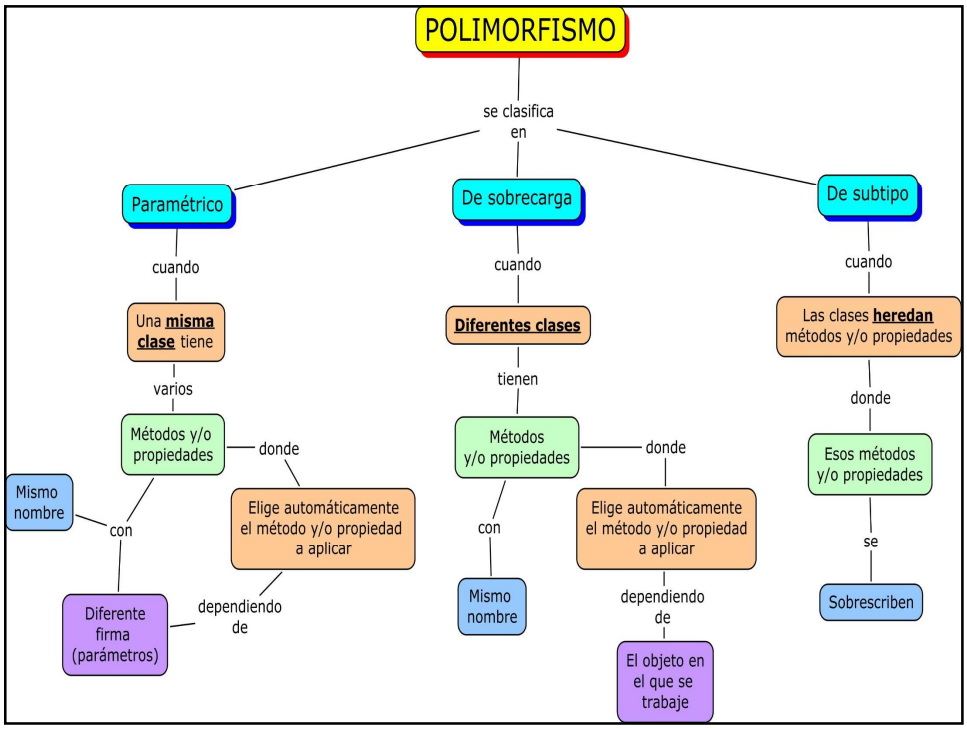
miLibro y **miLaptop** heredan el método **Abrir ( )** pero NO lo utilizan; sino que cada uno lo implementa nuevamente de manera distinta.

Un libro y una Laptop se abren de diferente manera.

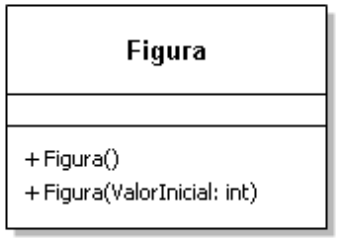
13




POLIMORFISMO
se presenta mediante
Firma (parámetros)
Overload
Overriding
cuando
Se sobrecargan métodos y/o propiedades
donde
Esos métodos y/o propiedades
tienen
se
Mismo nombre
Diferente firma (parámetros)
Comportan de manera diferente
Ubican en la misma clase
cuando
Diferentes clases definen métodos y/o propiedades
donde
Esos métodos y/o propiedades
tienen
se
Mismo nombre
Comportan de manera diferente
Ubican en clases independientes
cuando
Se heredan métodos y/o propiedades
donde
Esos métodos y/o propiedades
se
Sobrescriben


POLIMORFISMO
se clasifica en
Paramétrico
De sobrecarga
De subtipo
cuando
Una misma clase tiene
varios
Métodos y/o propiedades
donde
Elige automáticamente el método y/o propiedad a aplicar
dependiendo de
Mismo nombre
con
Diferente firma (parámetros)
cuando
Diferentes clases
tienen
Métodos y/o propiedades
donde
Elige automáticamente el método y/o propiedad a aplicar
dependiendo de
El objeto en el que se trabaje
con
Mismo nombre
cuando
Las clases heredan métodos y/o propiedades
donde
Esos métodos y/o propiedades
se
Sobrescriben

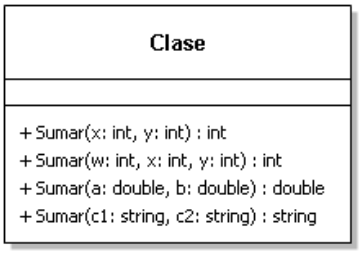
Ejemplo de polimorfismo paramétrico



- Una clase define varios métodos con el mismo nombre pero diferente firma (sobrecarga)
- Se elige el método de acuerdo a la firma aplicada
- La sobrecarga del constructor es un ejemplo de ello

17

Otro ejemplo de polimorfismo paramétrico

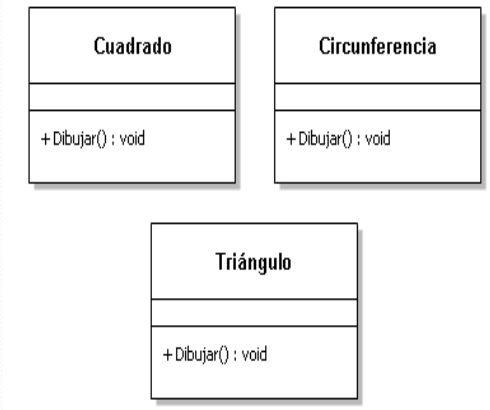


La misma clase tiene varios métodos con el mismo nombre pero diferentes firmas con diferentes tipos de datos

La sobrecarga de métodos **NO** provoca polimorfismo de sobrecarga, sino polimorfismo paramétrico

18

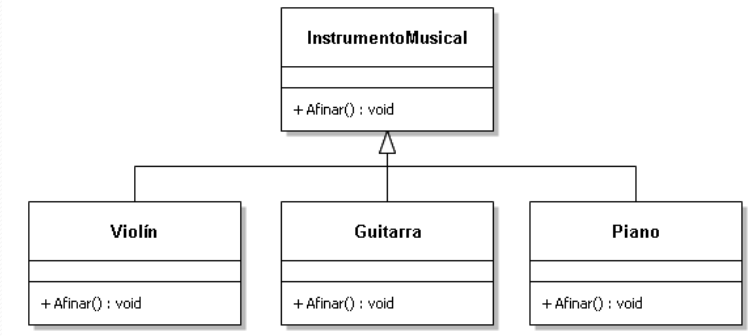
Ejemplo de polimorfismo de sobrecarga (overload)



- Diferentes clases tienen un método con el mismo nombre, pero comportamiento diferente
- Se aplica el método de acuerdo al objeto en que se trabaje

19

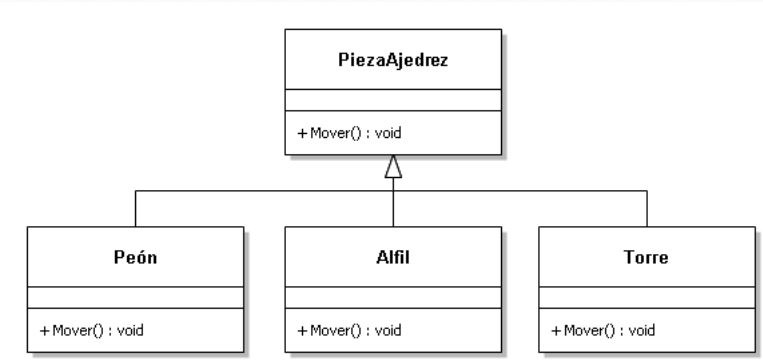
Ejemplo de polimorfismo de subtipo (override)



- Las clases derivadas redefinen los métodos y/o propiedades heredados mediante la sobrescritura (override)

20

Otro ejemplo de polimorfismo de subtipo (override)



- Se sobrescribe el método heredado **`Mover()`** según lo requiera la pieza del ajedrez

21

Diferencia entre Polimorfismo y Sobrecarga

- Un método está sobrecargado si dentro de una clase existen dos o más declaraciones de dicho método con el mismo nombre pero con parámetros distintos.

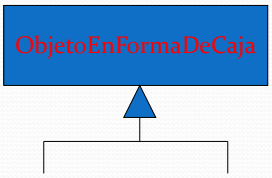
- En definitiva: La sobrecarga se resuelve en tiempo de compilación utilizando los nombres de los métodos y los tipos de sus parámetros; el polimorfismo se resuelve en tiempo de ejecución del programa, esto es, mientras se ejecuta, en función de la clase a la que pertenece el objeto.

22

Polimorfismo

POLI = Múltiples MORFISMO = Formas

```
miRefrigerador.Abrir( "Puerta de Abajo" )
miRefrigerador.Abrir( "Puerta de Arriba" , "Mitad" )
```



miRegalo.Abrir( ) miCofre.Abrir( )

23

Métodos virtuales

- Son métodos en la clase base pensados para ser sobrescritos por subclases.
- Para declararlos, se utiliza la palabra reservada "`virtual`"; para sobrescribirlos, en la subclase se utiliza la palabra reservada "`override`".
- Un método virtual "PUEDE" ser sobrescrito, o utilizarse tal como está.
- Solo se puede utilizar "`override`" si el método en la clase base está marcado como "`virtual`", "`abstract`" u "`override`".
- El método "`override`" debe mantener el mismo nivel de acceso que el método "`virtual`" correspondiente

24

Método virtual

```
class ClaseBase
{
   // Método virtual (preparado para ser modificado
en una clase derivada)
   public virtual void Metodo()
   {
      . . . .
   }
}
```

25

Método sobrescrito

```
class ClaseDerivada : ClaseBase
{
   // Sobrescritura del método heredado
   public override void Metodo()
   {
      . . . .
   }
}
```

26

Ejemplo virtual…override

```
class Vehiculo
{    public virtual void Arrancar()
     {  System.Console.WriteLine("Arrancar...Clase Vehiculo");
     }
}

class Carro: Vehiculo
{   public override void Arrancar()
    {   System.Console.WriteLine("Arrancar....Clase Carro");
    }
}

class Programa
{   static void Main()
    {   Carro miCarro = new Carro();
        miCarro.Arrancar();
        System.Console.ReadLine();
    }
}
```

Ejecución del programa…
Arrancar....Clase Carro

27

Ejemplo virtual…override (Polimorfismo en Tiempo de ejecución)

```
class Vehiculo
{   public virtual void Arrancar()
    {   System.Console.WriteLine("Arrancar...Clase Vehiculo");
    }
}

class Carro : Vehiculo
{   public override void Arrancar()
    {   System.Console.WriteLine("Arrancar....Clase Carro");
    }
}

class Programa
{   static void Main()
    {   Vehiculo v;
        v = new Vehiculo();
        v.Arrancar();
        v = new Carro();
        v.Arrancar();
        System.Console.ReadLine();
    }
}
```

En una variable tipo "Vehiculo" se almacenan objetos tipo "Vehiculo" y tipo "Carro". Al invocar el mismo método para el mismo objeto, se observa una conducta diferente, apropiada para cada objeto.

Ejecución del programa…
Arrancar...Clase Vehiculo
Arrancar....Clase Carro

28

override sealed

- Agregar "`sealed`" a un método "`override`" impide la futura sobrescritura de ese método, proporcionando una implementación final.

```
class Aparato
{   public virtual void Prender()
    {  System.Console.WriteLine(" Prendiendo el Aparato ");
    }
}

class TV : Aparato
{   public override sealed void Prender()
    {  System.Console.WriteLine(" LA TV SE ESTA PRENDIENDO ");
    }
}
class TVColor : TV
{   public override void Prender()
    {  System.Console.WriteLine(" La tele a color se esta prendiendo");
    }
}
```

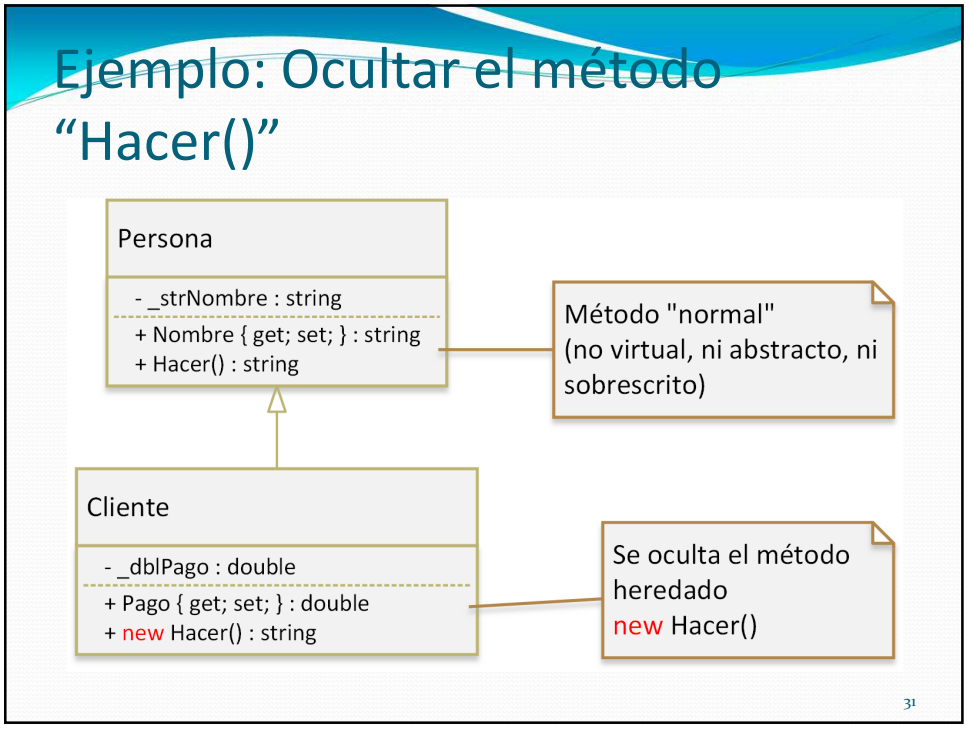
ERROR!!!...El método ya no se puede sobrescribir.

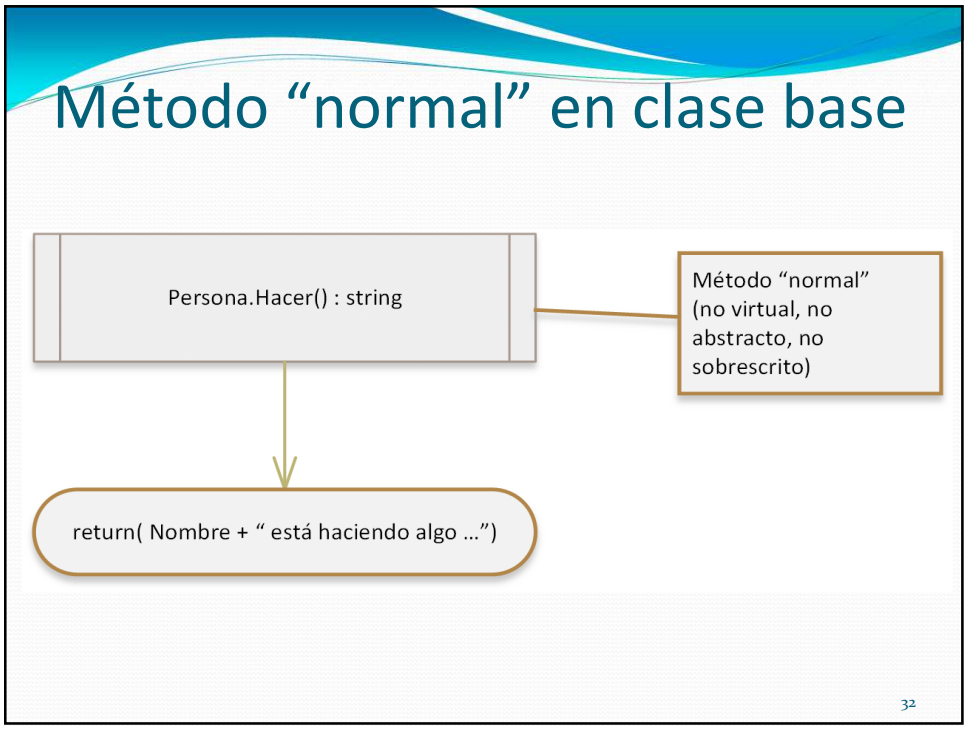
Ocultar métodos heredados

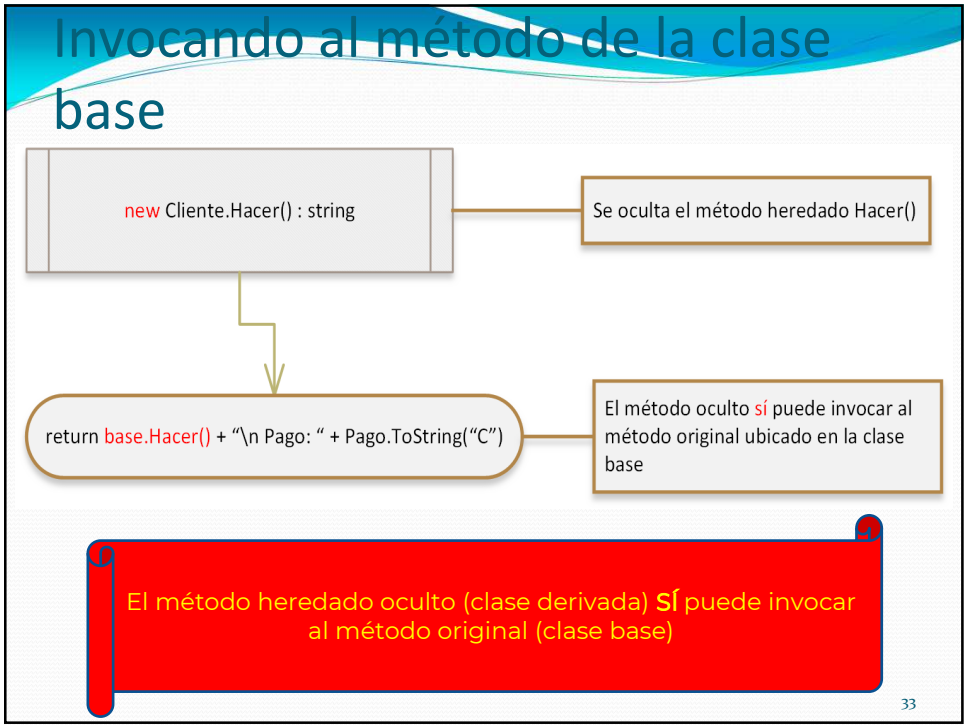
- Es posible ocultar un método heredado e introducir uno nuevo a la jerarquía de clases. El método antiguo (heredado) es reemplazado por otro nuevo, diferente, pero con el mismo nombre y la misma firma.

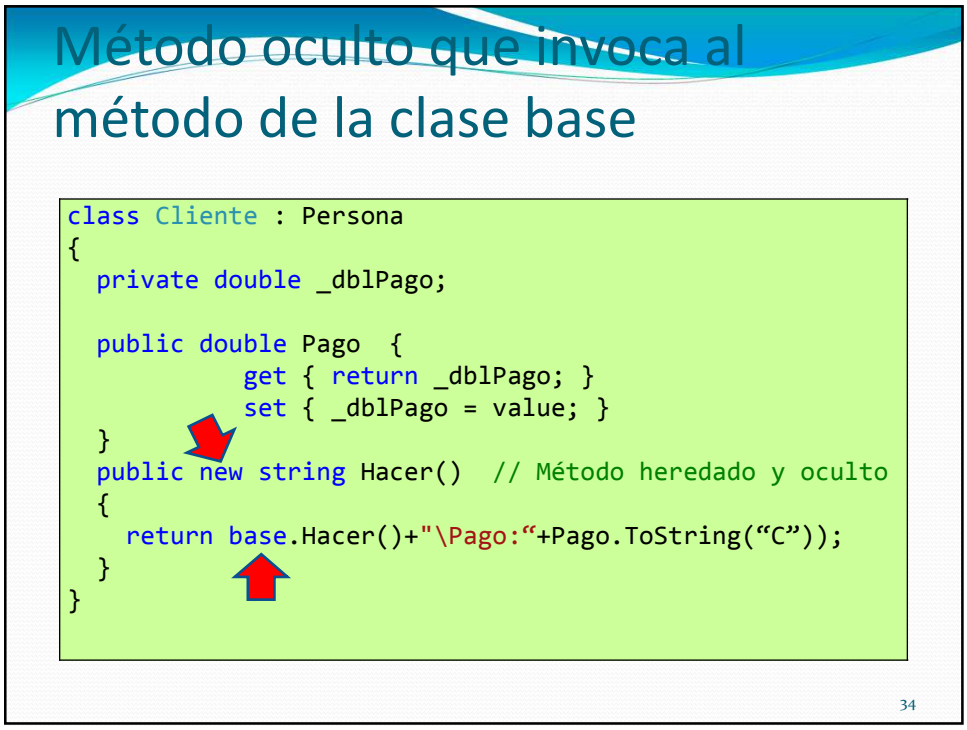
```
class Vehiculo
{
    public void Arrancar()
    {
        System.Console.WriteLine(" Clase Vehiculo. Metodo Arrancar ");
    }
}
class Automovil : Vehiculo
{
    public new void Arrancar()
    {
        System.Console.WriteLine(" Clase Automovil. Metodo Arrancar ");
    }
}
```

30


Ejemplo: Ocultar el método "Hacer()"
Persona
- _strNombre : string
+ Nombre { get; set; } : string
+ Hacer() : string
Método "normal"
(no virtual, ni abstracto, ni sobrescrito)
Cliente
- _dblPago : double
+ Pago { get; set; } : double
+ new Hacer() : string
Se oculta el método heredado
new Hacer()
31


Método "normal" en clase base
Persona.Hacer() : string
Método "normal"
(no virtual, no abstracto, no sobrescrito)
return(Nombre + " está haciendo algo ...")
32


Invocando al método de la clase base
new Cliente.Hacer() : string
Se oculta el método heredado Hacer()
return base.Hacer() + "\n Pago: " + Pago.ToString("C")
El método oculto sí puede invocar al método original ubicado en la clase base
El método heredado oculto (clase derivada) SÍ puede invocar al método original (clase base)
33

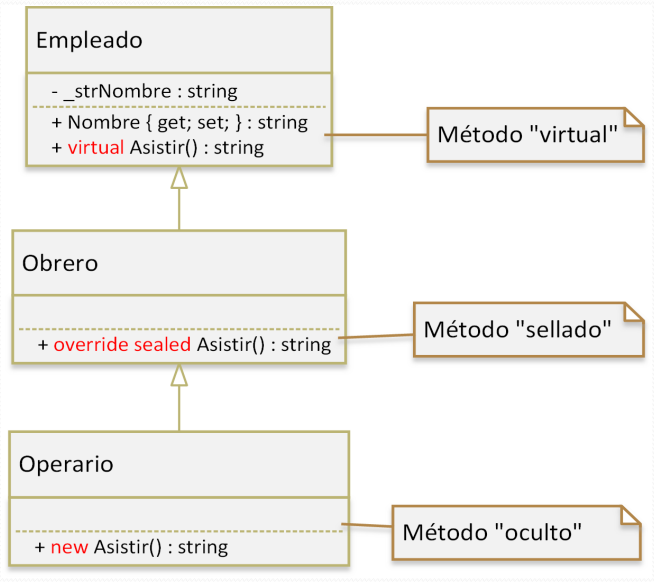

Método oculto que invoca al método de la clase base
class Cliente : Persona
{
private double _dblPago;
public double Pago {
get { return _dblPago; }
set { _dblPago = value; }
}
public new string Hacer() // Método heredado y oculto
{
return base.Hacer()+"\Pago:"+Pago.ToString("C"));
}
}
34

Ocultar un método heredado

Ocultar un método heredado es un "truco" para poder "sobrescribir" un método "normal"

35

Ejemplo: Ocultar un método "sellado"



36

Método “virtual” (clase base)

```
class Empleado
    {
        private string _strNombre;

        public string Nombre
        {
            get { return _strNombre; }
            set { _strNombre = value; }
        }

        public virtual string Asistir()
        {
            return Nombre + " asiste a trabajar ...";
        }
    }
```

37

Método “sellado” (1ª. clase derivada)

```
class Obrero : Empleado
{
   public override sealed string Asistir()
   {
     return Nombre + " asiste a su depto...";
   }
}
```

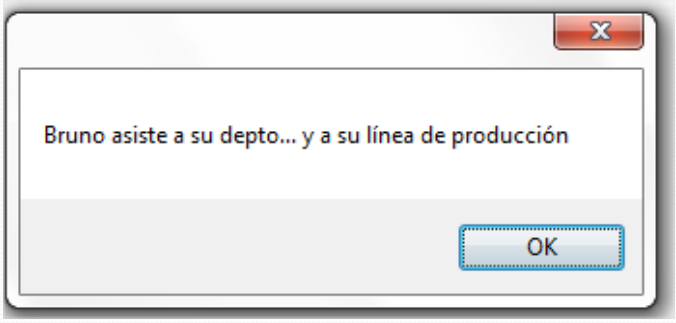
38

Método "oculto" (2ª. clase derivada)

```
class Operario : Obrero
{
  public new string Asistir()
  {
    return base.Asistir()+ " y a su línea de
producción";
  }
}
```

39

Salida



El método "oculto" `Operario.Asistir()` invoca al método "sellado" de su clase base `Obrero.Asistir()`

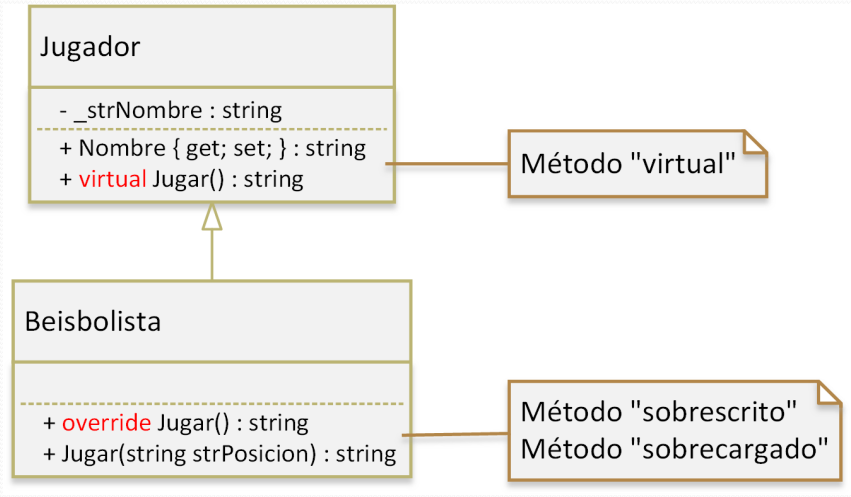
40

Ocultar un método “sellado”

Ocultar un método “sellado” es un “truco” para poder “sobrescribirlo” nuevamente

41

Ejemplo: Sobrecargar un método “override”



42

Método "virtual" (clase base)

```
class Jugador
{
   private string _strNombre;

   public string Nombre  {
            get { return _strNombre; }
            set { _strNombre = value; }
   }

   public virtual string Jugar()  {
      return Nombre + " juega ...";
   }
}
```

43

Método "override" y sobrecargado (clase derivada)

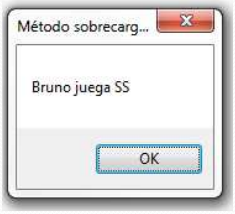
```
class Beisbolista: Jugador
{
   public override string Jugar()   {
            return base.Jugar()+ " beisbol";
   }

   public string Jugar(string strPosicion)  {
     return Nombre + " juega " + strPosicion;
   }
}
```

44

Salida


```
MessageBox.Show( unBeisbolista.Jugar() );
```




```
MessageBox.Show( unBeisbolista.Jugar("SS") );
```

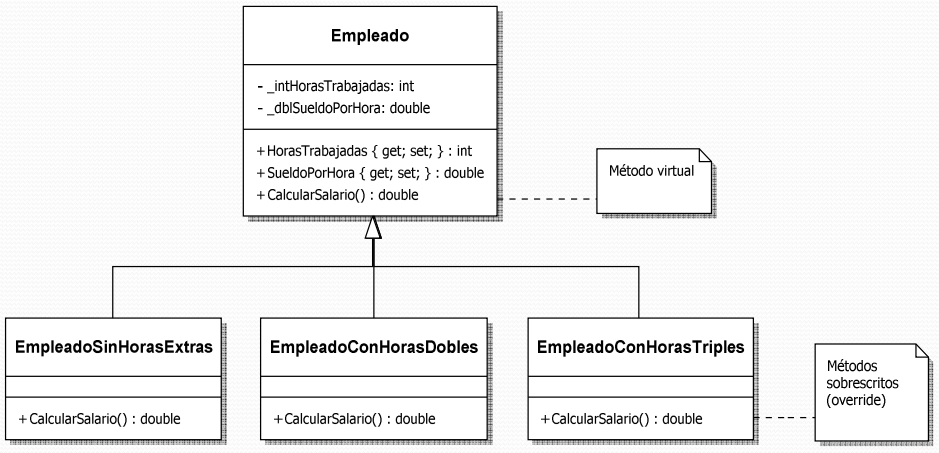
45

Ejercicio

- Una empresa contrata a varios empleados y les calcula su salario semanal de acuerdo a las horas trabajadas y al sueldo por hora considerando las horas trabajadas del empleado y tomando en cuenta que la base de trabajo es de 40 horas a la semana:
  - *Si trabaja 40 horas o menos, entonces simplemente calcula el sueldo normal*
  - *Si trabaja entre 41 y 45 horas, cada hora extra se le paga al doble*
  - *Si trabaja más de 45 horas, cada hora extra se le paga al triple*

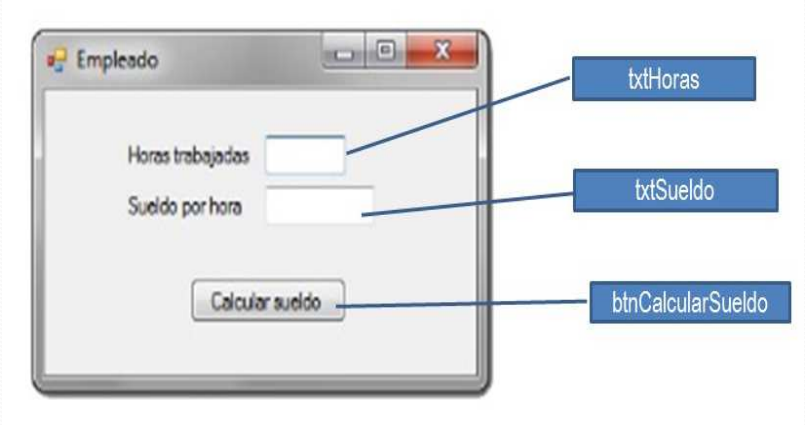
46

Ejercicio

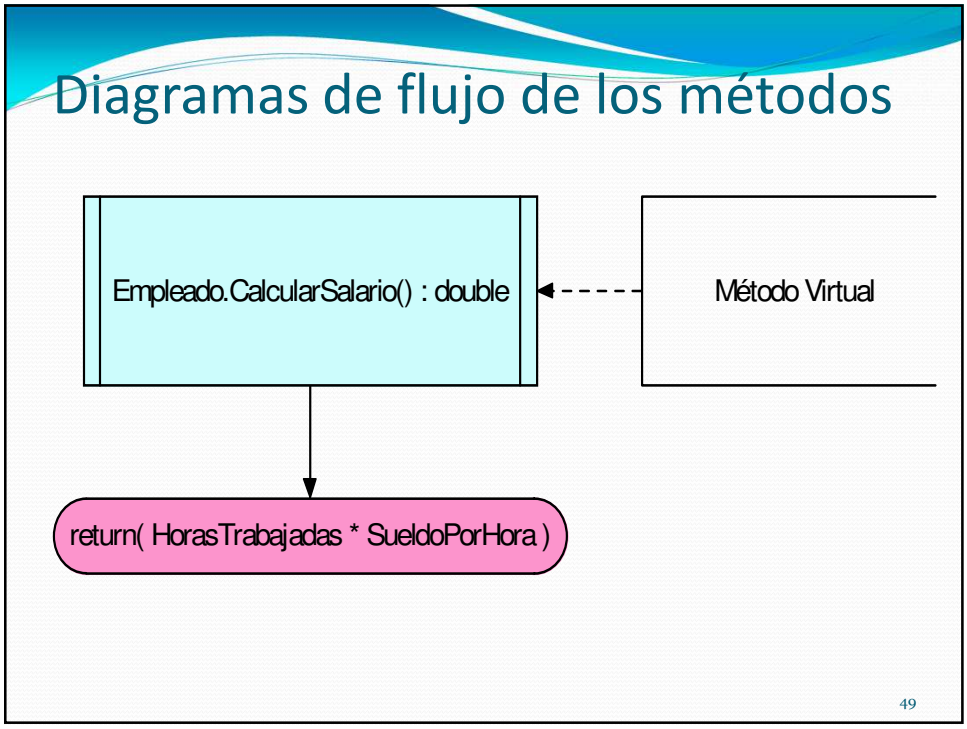


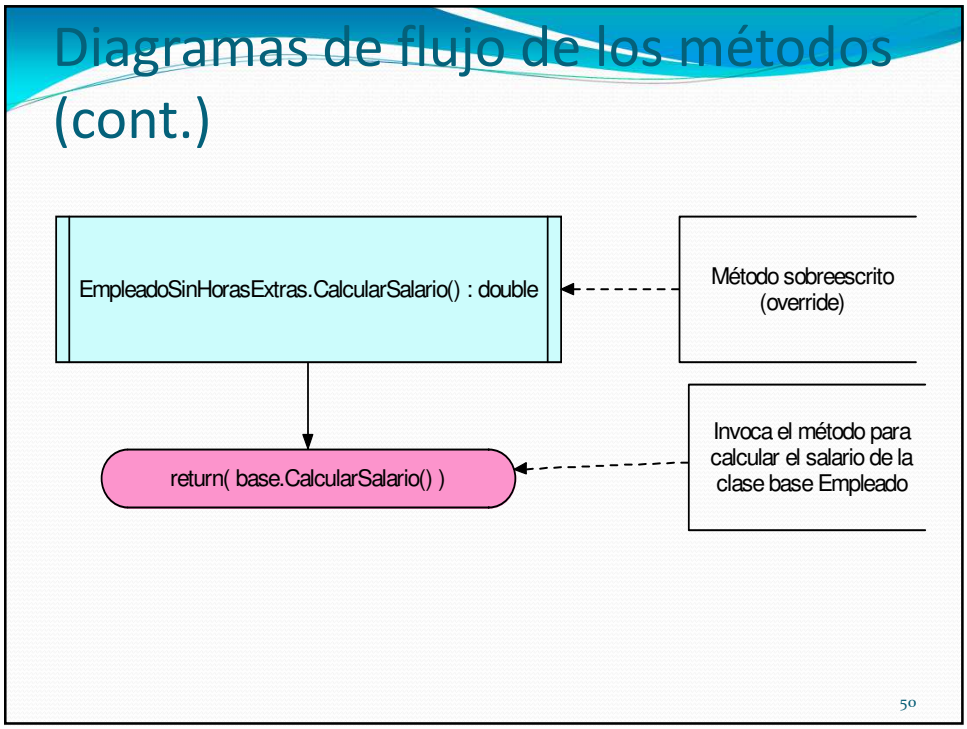
47

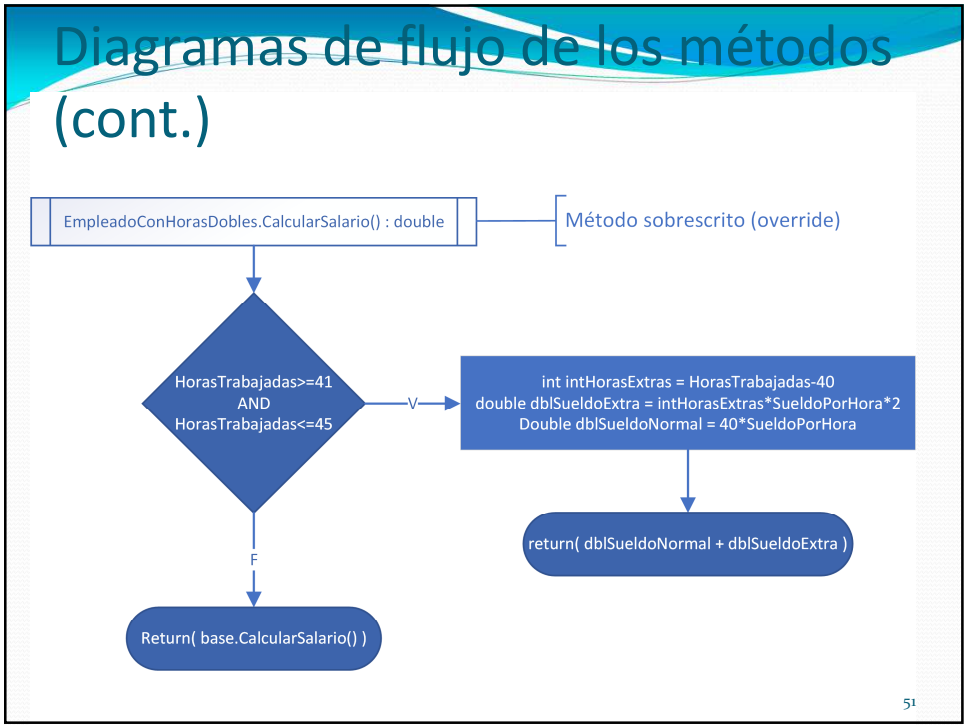
Diseño de la forma



48


Diagramas de flujo de los métodos
Empleado.CalcularSalario() : double
Método Virtual
return(HorasTrabajadas * SueldoPorHora)
49


Diagramas de flujo de los métodos (cont.)
EmpleadoSinHorasExtras.CalcularSalario() : double
Método sobreescrito (override)
return(base.CalcularSalario())
Invoca el método para calcular el salario de la clase base Empleado
50


Diagramas de flujo de los métodos (cont.)
EmpleadoConHorasDobles.CalcularSalario() : double
Método sobrescrito (override)
HorasTrabajadas>=41 AND HorasTrabajadas<=45
V
int intHorasExtras = HorasTrabajadas-40
double dblSueldoExtra = intHorasExtras*SueldoPorHora*2
Double dblSueldoNormal = 40*SueldoPorHora
return(dblSueldoNormal + dblSueldoExtra)
F
Return(base.CalcularSalario())
51

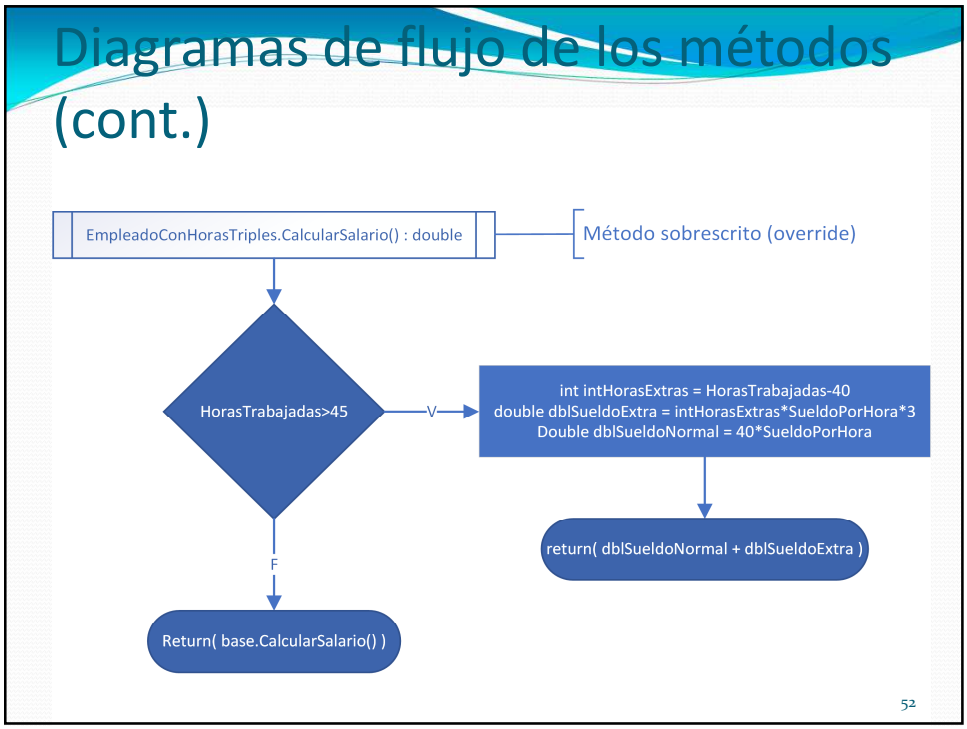
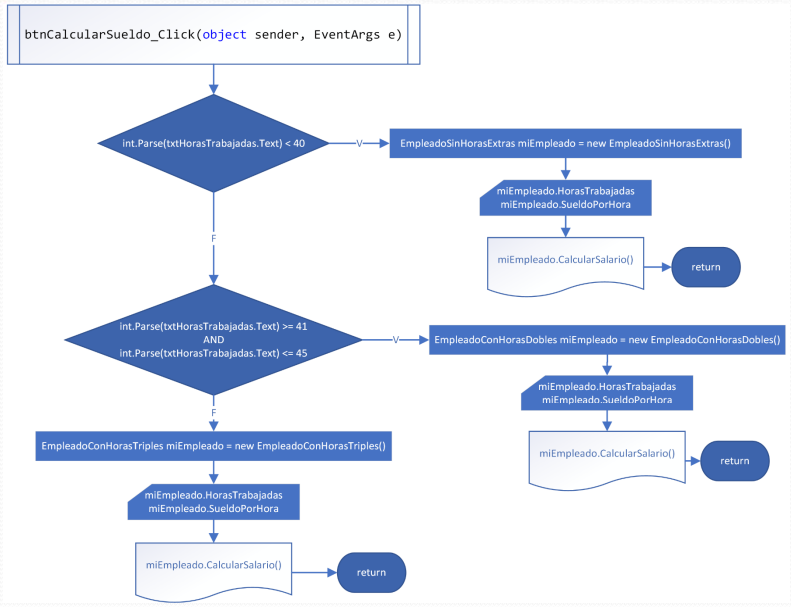

Diagramas de flujo de los métodos (cont.)
EmpleadoConHorasTriples.CalcularSalario() : double
Método sobrescrito (override)
HorasTrabajadas>45
V
int intHorasExtras = HorasTrabajadas-40
double dblSueldoExtra = intHorasExtras*SueldoPorHora*3
Double dblSueldoNormal = 40*SueldoPorHora
return(dblSueldoNormal + dblSueldoExtra)
F
Return(base.CalcularSalario())
52

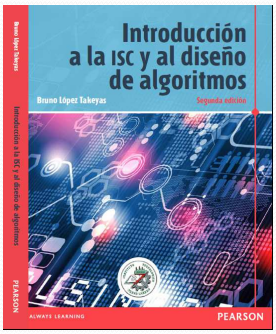
Diagrama de flujo del botón

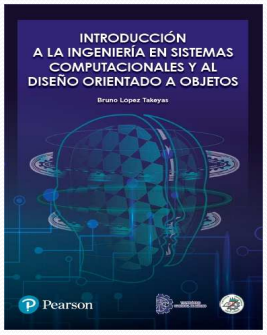


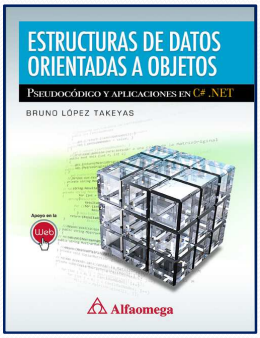
53

Otros títulos del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas