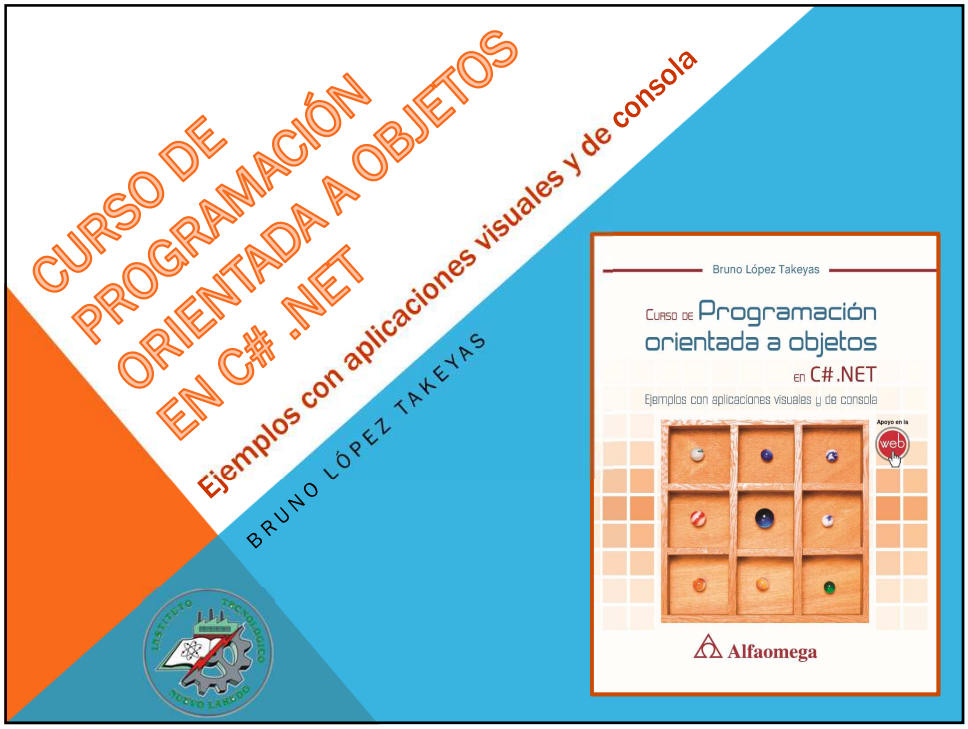

CURSO DE PROGRAMACIÓN ORIENTADA A OBJETOS EN C# .NET
Ejemplos con aplicaciones visuales y de consola
BRUNO LÓPEZ TAKEYAS
Bruno López Takeyas
Curso de Programación orientada a objetos en C#.NET
Ejemplos con aplicaciones visuales y de consola
Alfaomega


CAPÍTULO 8
EXCEPCIONES
2

Preguntas detonadoras

- ¿Cómo se controlan los posibles errores que ocurran durante la ejecución de una aplicación?
- Un programador, ¿puede disparar sus propias excepciones?
- ¿Cómo se prepara a una aplicación para que maneje errores?

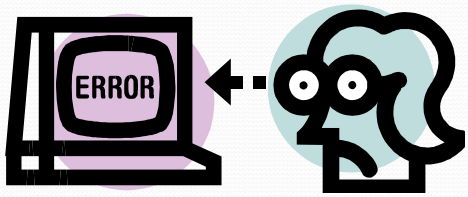
3

¿Qué es una excepción?

- *Es un evento que ocurre durante la ejecución de un programa y que interrumpe el flujo normal de operación*

Esperando lo inesperado !!

- Ocurrencia de sucesos que se consideran excepcionales.
- Cómo manejar situaciones anómalas
- Pueden ocurrir durante la ejecución de un programa



Control de excepciones

- *Debe ser simple de usar y de entender*
- *Debe separar el código del tratamiento de excepciones del código normal*
- *Implementar un tratamiento uniforme de las excepciones*
- *Permitir que las acciones de recuperación sean programadas*

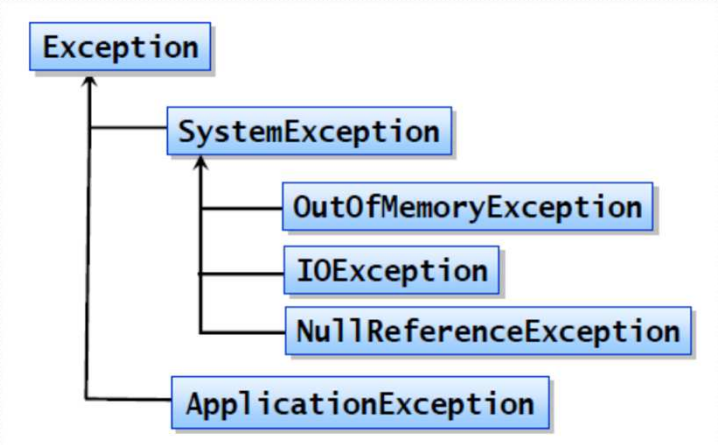
Tipos de excepciones

Tipos de excepciones

- *Implícitas.- Definidas por el lenguaje*
- *Explícitas.- Definidas por el programador*

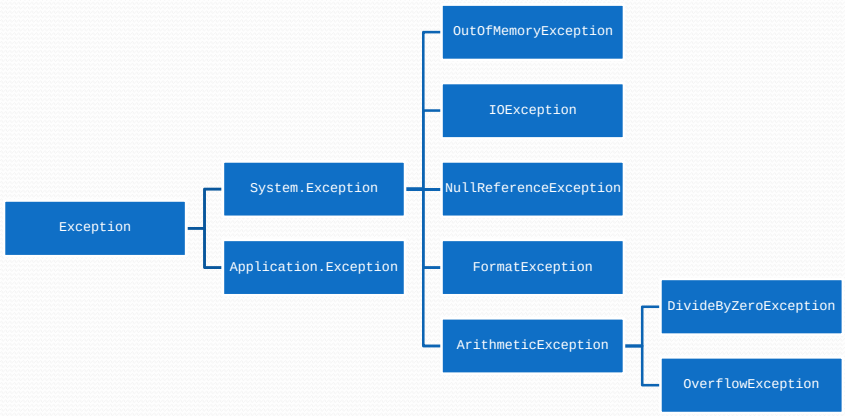
Excepciones en C#

- **Todas las excepciones derivan de `System.Exception`**



El espacio de nombres System.Exception

- *Incluido en la librería de clases (BCL)*



Algunas excepciones

Excepción	Origen (causa del error)
FormatException	El formato de un dato no corresponde con sus especificaciones
DivideByZeroException	Cuando se intenta calcular una división por cero (el denominador es cero)
OverflowException	Cuando una operación aritmética produce un resultado que está fuera del intervalo de datos permitido.
OutOfMemoryException	No hay suficiente espacio de memoria para crear un objeto
IndexOutOfRangeException	Cuando se intenta acceder a una celda de un arreglo cuyo índice está fuera del rango permitido

Introducción al manejo de excepciones

- Excepción
- Manejador de excepción
- Levantamiento de una excepción

Manejo de excepciones en C#

C# envía una excepción cuando ocurre un error en el programa y detiene su ejecución.

Si se desea que la aplicación siga ejecutándose después del error, entonces se usa:

✓ **`try`** para poner en alerta al programa sobre el código que puede lanzar una excepción.

✓ **`catch`** para capturar y manejar cada excepción que se lance.

✓**`finally`** código que se ejecutará haya o no excepciones.

Manejo de excepciones en C# (*cont.*)

```
try
{
        [Bloque de código que puede causar errores]
}

catch
{
        [Qué hacer si sucede un error]
}

finally
{
        [De cualquier manera, hacer lo siguiente…]
}
```

No deben declararse variables dentro de un bloque de sentencias try, ya que serían consideradas locales y únicamente dicho bloque podría utilizarlas

¿Qué hacer después de manejar la excepción?

Cuando el manejador termina se pueden hacer dos cosas:

- **Reanudar** la ejecución del bloque
- **Terminar** la ejecución del bloque y devolver el control al punto de invocación.

Tratamiento de excepciones

- Tratamiento de excepciones orientado a objetos :

```
try {
        ...     // bloque normal de código
}
catch {
        ...     // bloque que controla la excepción
}[ finally {
        ...     // bloque de finalización que siempre se ejecuta
}]
```

- Ejemplo:

```
try {
  Console.WriteLine("Escriba un número");
  int i = int.Parse(Console.ReadLine());
}
catch (OverflowException capturada)
{
  Console.WriteLine(capturada);
}
```

Tratamiento de excepciones

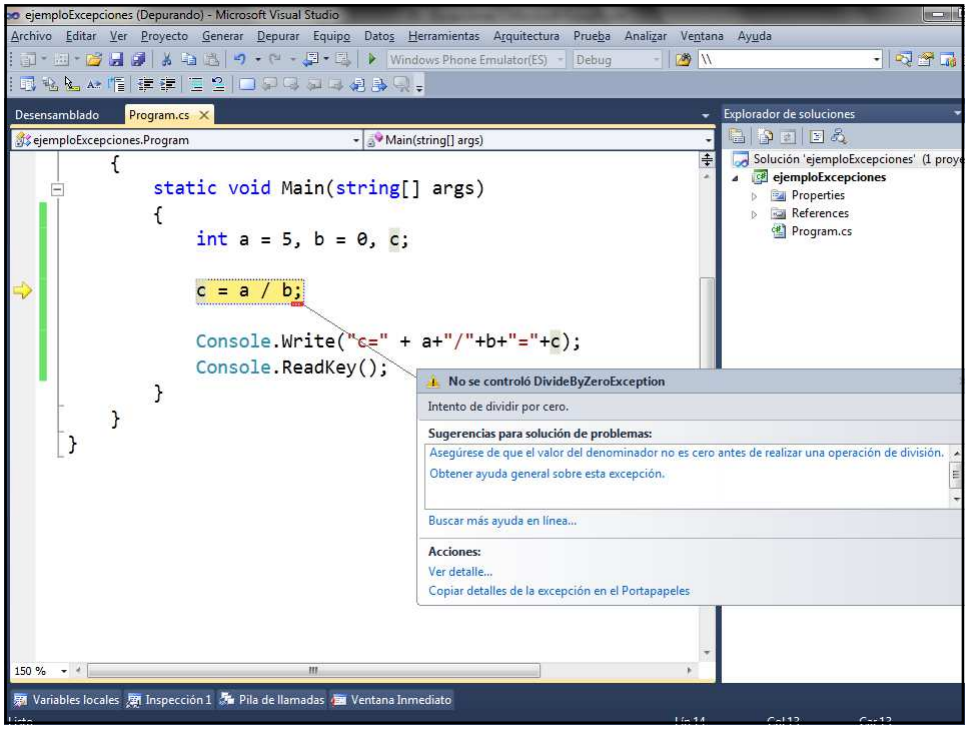
- Cada bloque *catch* captura una clase de excepción
- Un bloque *try* puede tener un bloque *catch* general que capture excepciones no tratadas (uno solo y el último de los bloques *catch*)
- Un bloque *try* no puede capturar una excepción derivada de una clase capturada en un bloque catch anterior

```
…
try
{
  Console.WriteLine("Escriba el primer número");
  int i = int.Parse(Console.ReadLine());
  Console.WriteLine("Escriba el segundo número");
  int j = int.Parse(Console.ReadLine());
  int k = i / j;
}
catch (OverflowException capturada) {
  Console.WriteLine(capturada); }
catch (DivideByZeroException capturada)
  {Console.WriteLine(capturada); }
catch {…}  // también: catch (Exception x) { … }
```

Ejemplo sin manejo de excepciones

```
static void Main(string[] args)
        {
            int a = 5, b = 0, c;

            c = a / b;

            Console.Write("c=" + a+"/"+b+"="+c);
            Console.ReadKey();
        }
```

Intenta ejecutar una división por cero (b = 0)



DivideByZeroException

```
static void Main(string[] args)
        {
            int a = 5, b = 0, c;

            try
            {
                c = a / b;
            }
            catch (DivideByZeroException x)
            {
                Console.WriteLine(x.Message);
                Console.ReadKey();
                return;
            }

            Console.Write("c=" + a+"/"+b+"="+c);
            Console.ReadKey();
        }
```

Intenta ejecutar una división por cero (b = o)

Captura la excepción DivideByZeroException

Propiedad con el mensaje de la excepción

Mensajes de excepciones

Ejemplo con manejo de excepciones

Ejemplo de división por cero

Codificación del botón para hacer la división

```
private void button1_Click(object sender, EventArgs e)
{
   int dividendo=0, divisor=0, resultado=0;
   try
   {
      dividendo = int.Parse(textBox1.Text);
      divisor = int.Parse(textBox2.Text);
      resultado = dividendo / divisor;
   }
   catch (Exception x)
   {
      MessageBox.Show("ERROR: "+x.Message);
   }
   finally
   {
      textBox3.Text = resultado.ToString();
   }
}
```

Captura cualquier excepción que se dispare

Propiedad con el mensaje de la excepción

Otro ejemplo para validar la captura de datos

```
int a=0;
bool Bandera = true;
do
{
    Bandera = false;
    try
    {
       Console.Write("Capture un número entero: ");
       a = int.Parse(Console.ReadLine());
    }
    catch (Exception x)
    {
       Console.WriteLine("ERROR: " + x.Message);
       Console.ReadKey();
       Bandera = true;
    }
} while (Bandera);
```

Captura cualquier excepción que se dispare

Propiedad con el mensaje de la excepción

Ejemplo con try-catch

```
public static void Main(string[] args)
       {
         int dato1 = 0, dato2 = 0, dato3;
         System.Console.WriteLine("Se inicia la aplicacion");
         try
              {
                dato1++;
                dato3 = dato1 / dato2;
                dato2++;
              }
         catch (System.DivideByZeroException e) {
              Console.WriteLine("Error: " + e.Message);
              dato3 = dato1;
              }
        //Otras sentencias
         Console.WriteLine(dato1 + " " + dato2 + " " + dato3);
       }
```

CATCH atrapará solamente excepciones de tipo DIVIDEBYZERO EXCEPTION

OverflowException

Por defecto, **NO** se verifica el desborde aritmético

```
checked {
    int number = int.MaxValue;
    Console.WriteLine(++number);
}
```

OverflowException

Dispara una excepción
No se ejecuta la impresión.

```
unchecked {
    int number = int.MaxValue;
    Console.WriteLine(++number);
}
```

¿ MaxValue + 1 es negativo?

-2147483648

Ejemplo sin verificación

```
static void Main(string[] args)
{
  int x = int.MaxValue;

  Console.WriteLine("x=" + x);
  // 2, 147, 483, 647

  x++;

  Console.WriteLine("x=" + x);
  // -2, 147, 483, 648 ??????
}
```

ERROR:
No verifica
OverflowException !!!

Ejemplo con verificación

```
static void Main(string[] args)
{
  int x = int.MaxValue;
  Console.WriteLine("x=" + x);
  // 2, 147, 483, 647
  try
  {
     checked
     {
        x++;
     }
  }
  catch(OverflowException)
  {
    Console.Write("Número demasiado grande !!!");
    return;
  }
}
```

Solución:
Verificar
OverflowException

Tipos de excepciones

Excepciones de sistema:

Cuando se realiza alguna operación no válida se lanza automáticamente.

Acceso a algún objeto que no existe, división por cero...

Excepciones de programador:

Se define una clase que herede de *Throwable* o de *Exception*.

Excepciones de usuario:

Gestiona la excepción mediante los bloques de código *try*, *catch*, *finally*.

Indica que el código producirá una excepción que no se tratará dentro de él y se pasará al método superior utilizando ***throw***.

La sentencia throw

- La instrucción **throw** se utiliza para señalizar la aparición de una situación anómala (excepción) durante la ejecución del programa.
- Se puede utilizar una instrucción throw en el bloque **catch** para volver a producir la excepción, la cual ha sido capturada por la instrucción **catch**.
- El programador puede disparar una excepción mediante:
  - `throw new Exception("Error: …….");`

Ejemplo de la sentencia throw

```
static void Main(string[] args)
{
    int a=3, b=0, c=0;

    try
    {
        c = CalcularDivision(a, b);
    }
    catch (Exception x)
    {
        Console.WriteLine(x.Message);
    }
    finally
    {
        Console.WriteLine(a+"/"+b+"="+c);
    }
    Console.ReadKey();
}
```

Ejemplo de la sentencia throw

```
static int CalcularDivision(int numerador, int denominador)
{
   if (denominador == 0)
    throw new Exception("El denominador NO debe ser cero");
   else
    return (numerador / denominador);
}
```

Ejemplo 1: Excepción general

```
static void Main( )
{
   try
      { System.Console.WriteLine(" Introduce un número: ");
      int a = System.Convert.ToInt32
      (System.Console.ReadLine() );
      }
   catch ( Exception e )
      { System.Console.WriteLine(" Ha habido un error..." +
      e.Message);
      }
   finally
      { System.Console.WriteLine(" Con error y Sin error, este
      mensaje aparece. ");
      System.Console.ReadLine();
      }
}
```

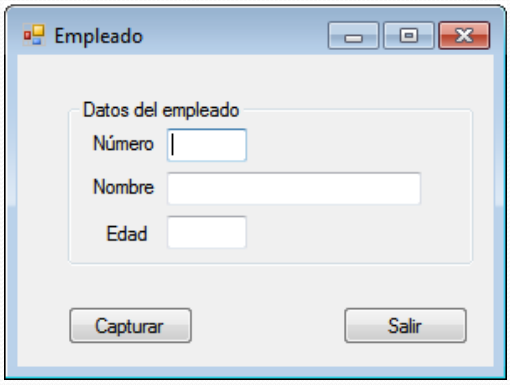
Ejemplo 2: Disparar una excepción mediante la sentencia throw

```
 using System;
class MainClass {
   static void ProcesarCadena(string s) {
   if (s == null)
   { throw new ArgumentNullException(); }
}
static void Main()
{
   try
   {  string s = null;
      ProcesarCadena(s);
   }
   catch (Exception e)
   {  Console.WriteLine("{0} Excepcion capturada.", e);
   }
}
}
```

Ejemplo 3: Secuencia de sentencias catch

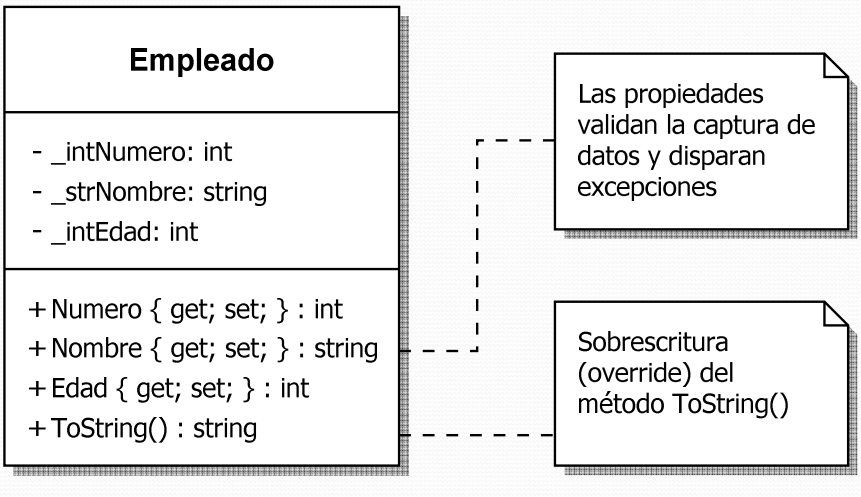
```
static void ProcesarCadena (string s)
{
   if (s == null)
   { throw new ArgumentNullException(); }
}
static void Main() {
try {
       string s = null;
       ProcesarCadena(s);
}
//Mas específico
catch (ArgumentNullException e)
{      Console.WriteLine("{0} First exception caught.", e); }
//Menos específico
catch (Exception e)
{      Console.WriteLine("{0} Second exception caught.", e);
} }
```

Uso de propiedades para validar la captura de datos



- Capturar datos y validar que se tecleen correctamente.
- No dejar datos en blanco
- Rango de valores permitido

Ejemplo

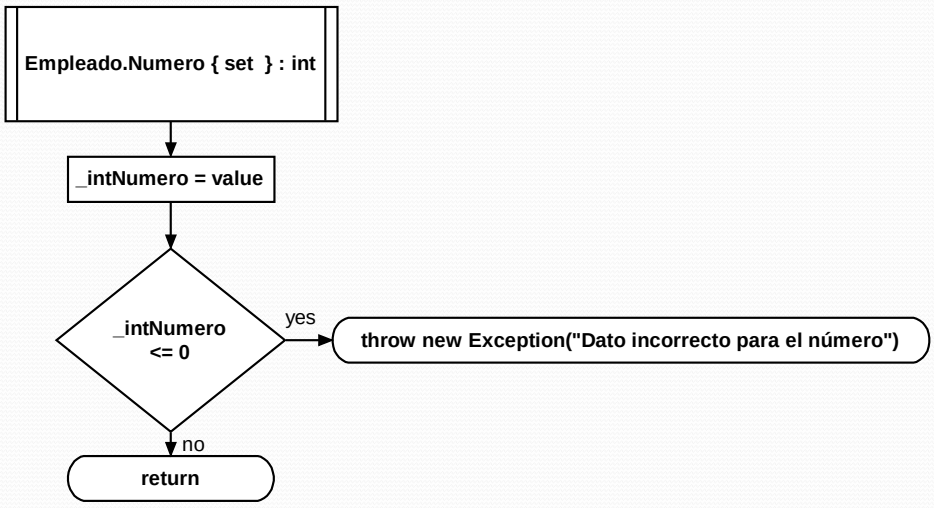


37

Uso de propiedades para validar la captura de datos (declaración de atributos)

```
class Empleado
{
  // Atributos privados
  private int _intNumero;
  private string _strNombre;
  private int _intEdad;

        :
        :
        :
```

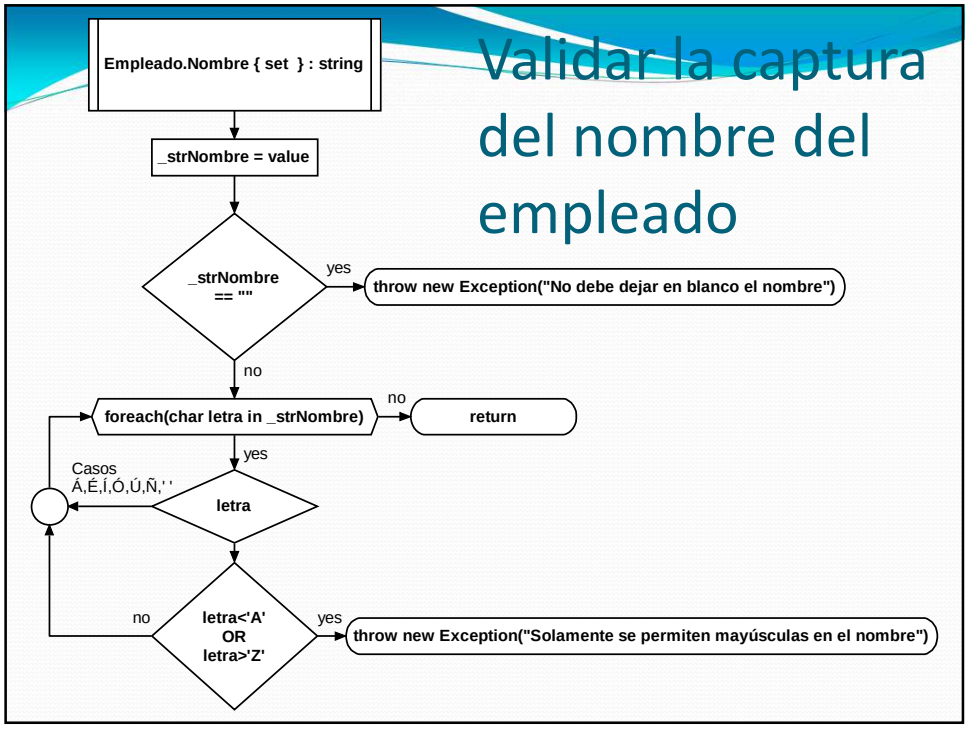
Validar la captura del número del empleado



Validar la captura del número del empleado

```
// Propiedad pública del número
public int Numero
{
   get { return _intNumero; }
   set
   {
     _intNumero = value;
     if (_intNumero <= 0)
      throw new Exception("Dato incorrecto para
el número");
   }
}
```

Validar la captura del nombre del empleado



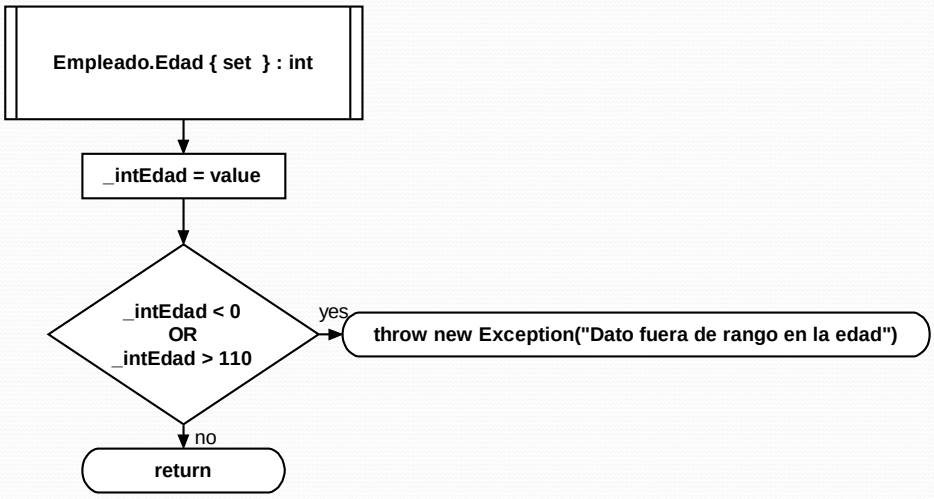
Validar la captura del nombre del empleado

```
public string Nombre
{
  get { return _strNombre; }
  set
  {
    _strNombre = value;
    if (_strNombre == "")
     throw new Exception("No debe dejar en blanco el nombre");

    foreach (char letra in _strNombre) {
      // Caracteres permitidos
      switch (letra)
      {
        case 'Á': continue;
        case 'É': continue;
        case 'Í': continue;
        case 'Ó': continue;
        case 'Ú': continue;
        case 'Ñ': continue;
        case 'Ü': continue;
        case ' ': continue;
      }

      if (letra < 'A' || letra > 'Z')
       throw new Exception("Solamente se permiten mayúsculas en el nombre (no capturar números ni
otros caracteres)");
  }
 }
}
```

Validar la captura de la edad del empleado

```
// Propiedad pública de la edad
public int Edad
{
  get { return _intEdad; }
  set
  {
    _intEdad = value;
    if (_intEdad < 0 || _intEdad > 110)
     throw new Exception("Dato fuera de rango en
la edad");
  }
}
```

Validar la captura de la edad del empleado



Validar la captura de datos desde el botón

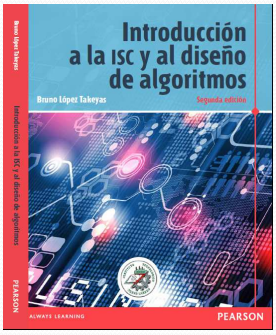
```
private void btnCapturar_Click(object sender, EventArgs e)
{
  Empleado miEmpleado = new Empleado();

  try
  {
     miEmpleado.Numero = int.Parse(txtNumero.Text);
  }
  catch (Exception x)
  {
     MessageBox.Show(x.Message);
     txtNumero.Text = "";
     txtNumero.Focus();
     return;
  }
```

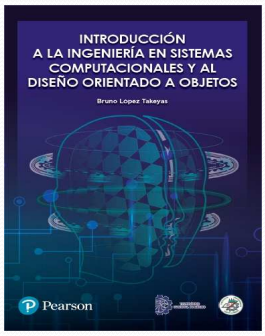
Ejemplo de salida

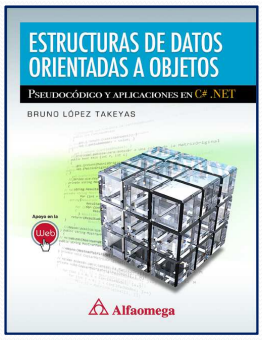
46

Otros títulos del autor

http://www.itnuevolaredo.edu.mx/Takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas