Unidad

3

DIPLOMATURA EN PROGRAMACION .NET

Ejercicios

Capítulo 6

Excepciones y aserciones

Ejercicio 1

En este ejercicio se utilizará el bloque `try-catch` o `Try-Catch` (C# o VB) para manejar una simple rutina de excepción.

El programa que se encuentra en la solución Capitulo6\ejercicio1 imprime por pantalla los elementos con que se inicializa el vector vec, pero tiene un pequeño error, intenta acceder a los elementos más allá del último (el elemento 8, con subíndice 7) que es el límite del vector.

C#

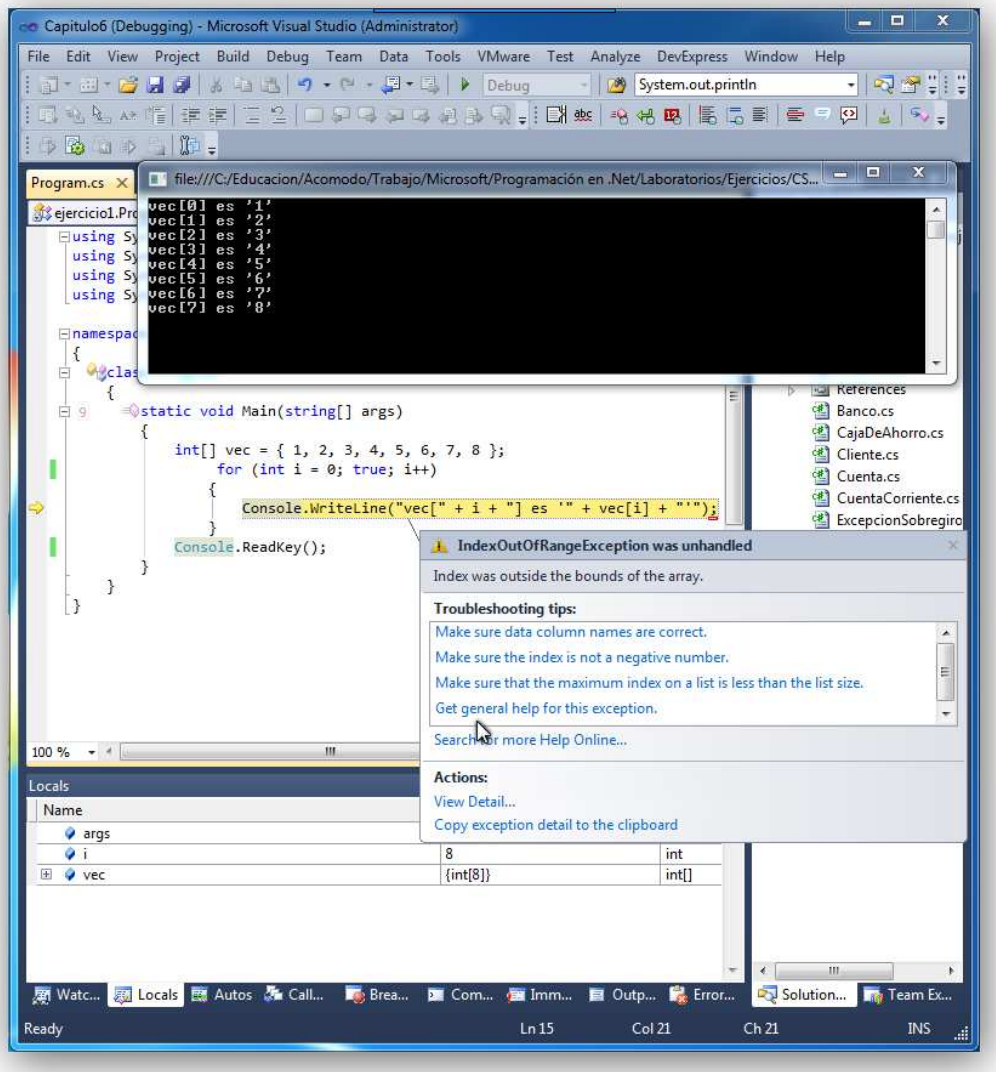
```
namespace ejercicio1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] vec = { 1, 2, 3, 4, 5, 6, 7, 8 };
                 for (int i = 0; true; i++)
                {
                    Console.WriteLine("vec[" + i + "] es '" + vec[i] + "'");
                }
            Console.ReadKey();
        }
    }
}
```

VB

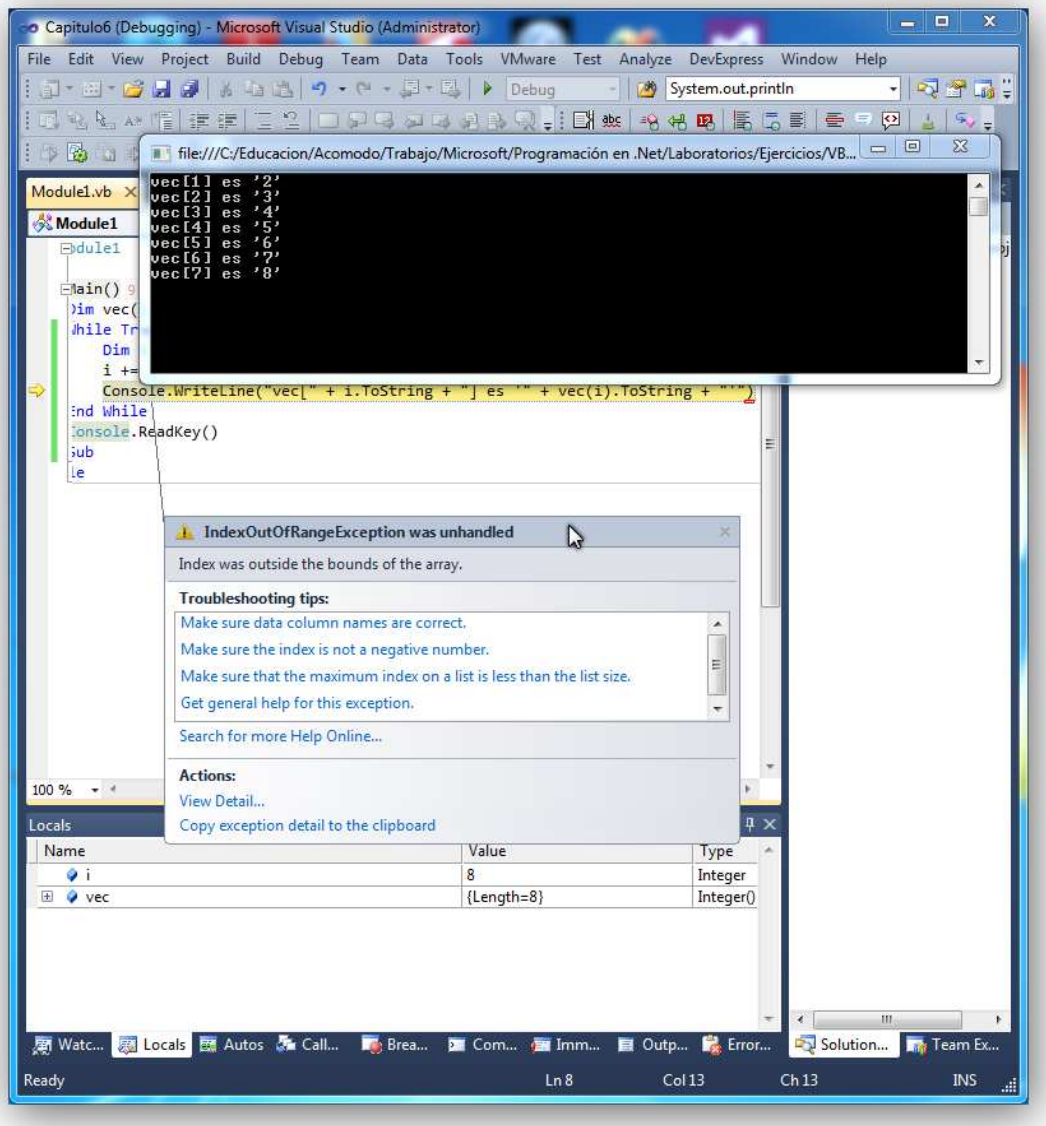
```
Module Module1

    Sub Main()
        Dim vec() As Integer = {1, 2, 3, 4, 5, 6, 7, 8}
        While True
            Dim i As Integer
            i += 1
            Console.WriteLine("vec[" + i.ToString + "] es '" + _
                     vec(i).ToString + "'")
        End While
        Console.ReadKey()
    End Sub
End Module
```


Nota: la mala implementación del código es intencional. Recordar que los vectores se recorren siempre verificando no exceder su largo máximo con la propiedad vec.length (en este caso). El propósito es experimentar con bloques que atrapen (catch) excepciones (exceptions).

1. Abrir la solución en el directorio Capítulo 6. Los errores que figuran luego de importar los archivos se deben a las modificaciones que deberán realizarse.
2. Abrir el proyecto ejercicio 1 también
3. Compilar y ejecutar el programa.
4. Verificar que la salida es:


Capitulo6 (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Data Tools VMware Test Analyze DevExpress Window Help
Debug
System.out.println
Program.cs
file:///C:/Educacion/Acomodo/Trabajo/Microsoft/Programación en .Net/Laboratorios/Ejercicios/CS...
vec[0] es '1'
vec[1] es '2'
vec[2] es '3'
vec[3] es '4'
vec[4] es '5'
vec[5] es '6'
vec[6] es '7'
vec[7] es '8'
static void Main(string[] args)
{
int[] vec = { 1, 2, 3, 4, 5, 6, 7, 8 };
for (int i = 0; true; i++)
{
Console.WriteLine("vec[" + i + "] es '" + vec[i] + "'");
}
Console.ReadKey();
}
References
Banco.cs
CajaDeAhorro.cs
Cliente.cs
Cuenta.cs
CuentaCorriente.cs
ExcepcionSobregiro
IndexOutOfRangeException was unhandled
Index was outside the bounds of the array.
Troubleshooting tips:
Make sure data column names are correct.
Make sure the index is not a negative number.
Make sure that the maximum index on a list is less than the list size.
Get general help for this exception.
Search for more Help Online...
Actions:
View Detail...
Copy exception detail to the clipboard
Locals
Name
args
i 8 int
vec {int[8]} int[]
Watc... Locals Autos Call... Brea... Com... Imm... Outp... Error... Solution... Team Ex...
Ready
Ln 15 Col 21 Ch 21 INS

VB



5. Modificar el programa para manejar la excepción IndexOutOfRangeException colocando el ciclo for o For (C# o VB) dentro de un bloque try o Try (C# o VB)
6. Escribir el bloque catch asociado con una referencia al objeto del tipo IndexOutOfRangeException como argumento.
7. Dentro del bloque catch declarar Console.WriteLine(e.Message); para obtener e imprimir por pantalla el mensaje que devuelve la excepción e imprimir por pantalla una cadena que indique el nombre de la excepción ocurrida y que el programa va a terminar.
8. Compilar y ejecutar el programa
9. Verificar que la salida sea

```
vec[0] es '1'
vec[1] es '2'
vec[2] es '3'
vec[3] es '4'
vec[4] es '5'
vec[5] es '6'
vec[6] es '7'
vec[7] es '8'
Index was outside the bounds of the array.
Se produjo la excepción IndexOutOfRangeException
El programa finalizó por esta causa
```

Ejercicio 2

En este ejercicio se utilizarán aserciones para comparar su funcionamiento con el manejo de excepciones realizado en el ejercicio 1.

1. Abrir el proyecto llamado ejercicio 2
2. Verifique que el código que se encuentra en Main es como el inicio del ejercicio 1:

C#

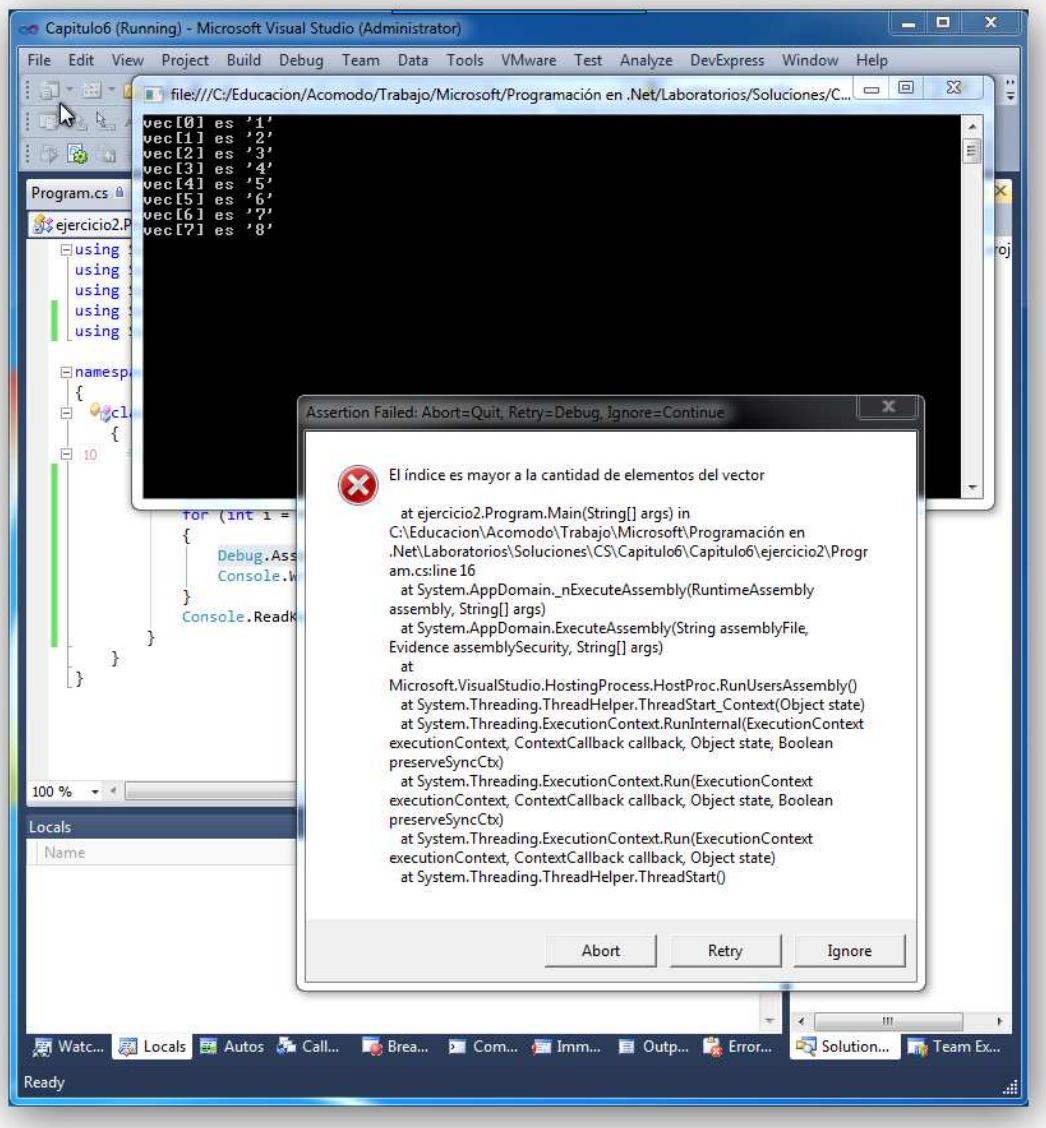
```
namespace ejercicio1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] vec = { 1, 2, 3, 4, 5, 6, 7, 8 };
                 for (int i = 0; true; i++)
                {
                    Console.WriteLine("vec[" + i + "] es '" + vec[i] + "'");
                }
            Console.ReadKey();
        }
    }
}
```

VB

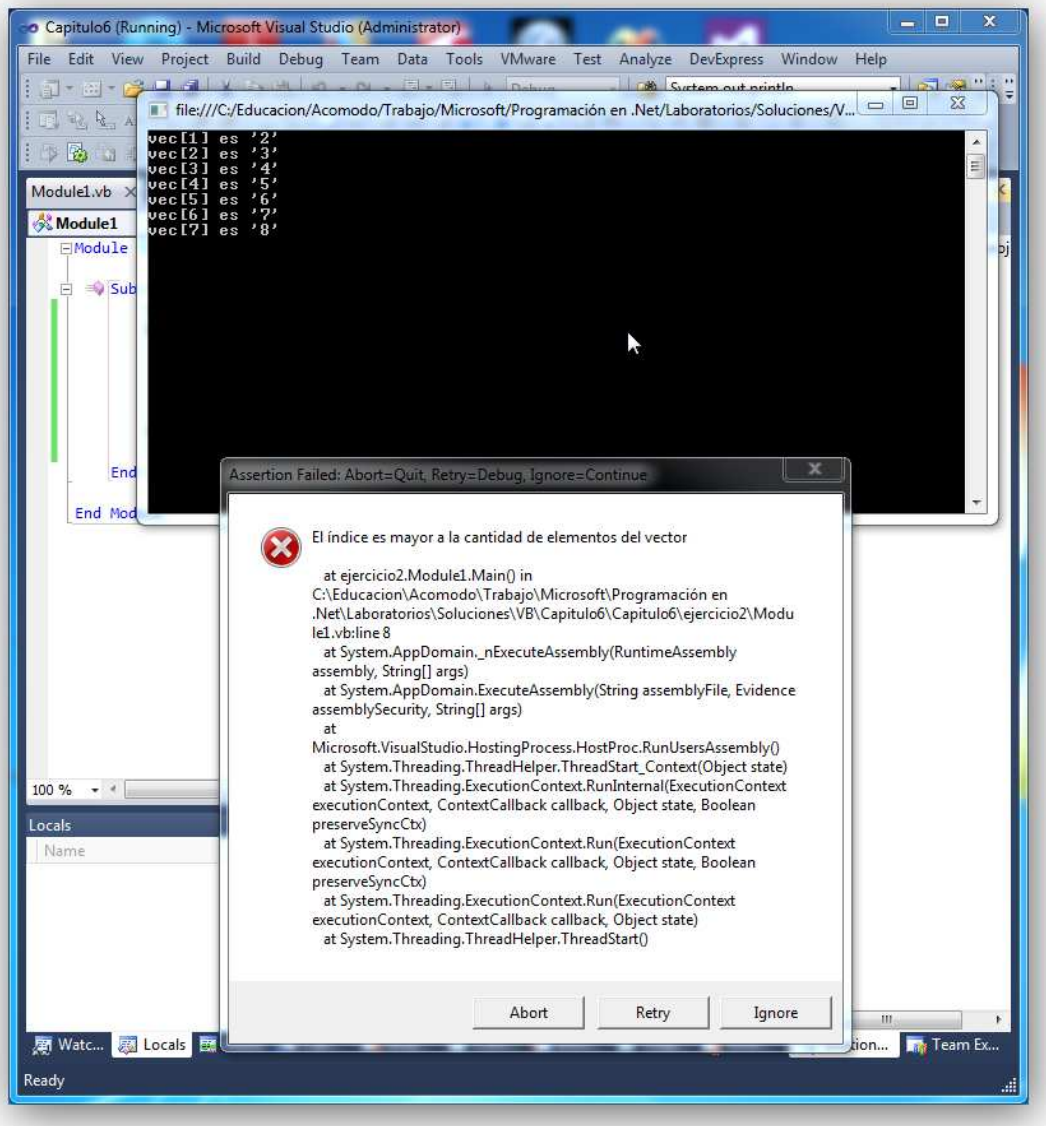
```
Module Module1

    Sub Main()
        Dim vec() As Integer = {1, 2, 3, 4, 5, 6, 7, 8}
        While True
            Dim i As Integer
            i += 1
            Console.WriteLine("vec[" + i.ToString + "] es '" + _
                    vec(i).ToString + "'")
        End While
        Console.ReadKey()
    End Sub
End Module
```

3. Diseñe una aseveración que finalice el programa antes que se exceda el límite del vector. Es decir, el programa deberá finalizar por un error de Assert y no por el lanzamiento de la excepción IndexOutOfRangeException.
4. Ejecutar el programa normalmente y verificar que la salida sea similar a la siguiente:

C#



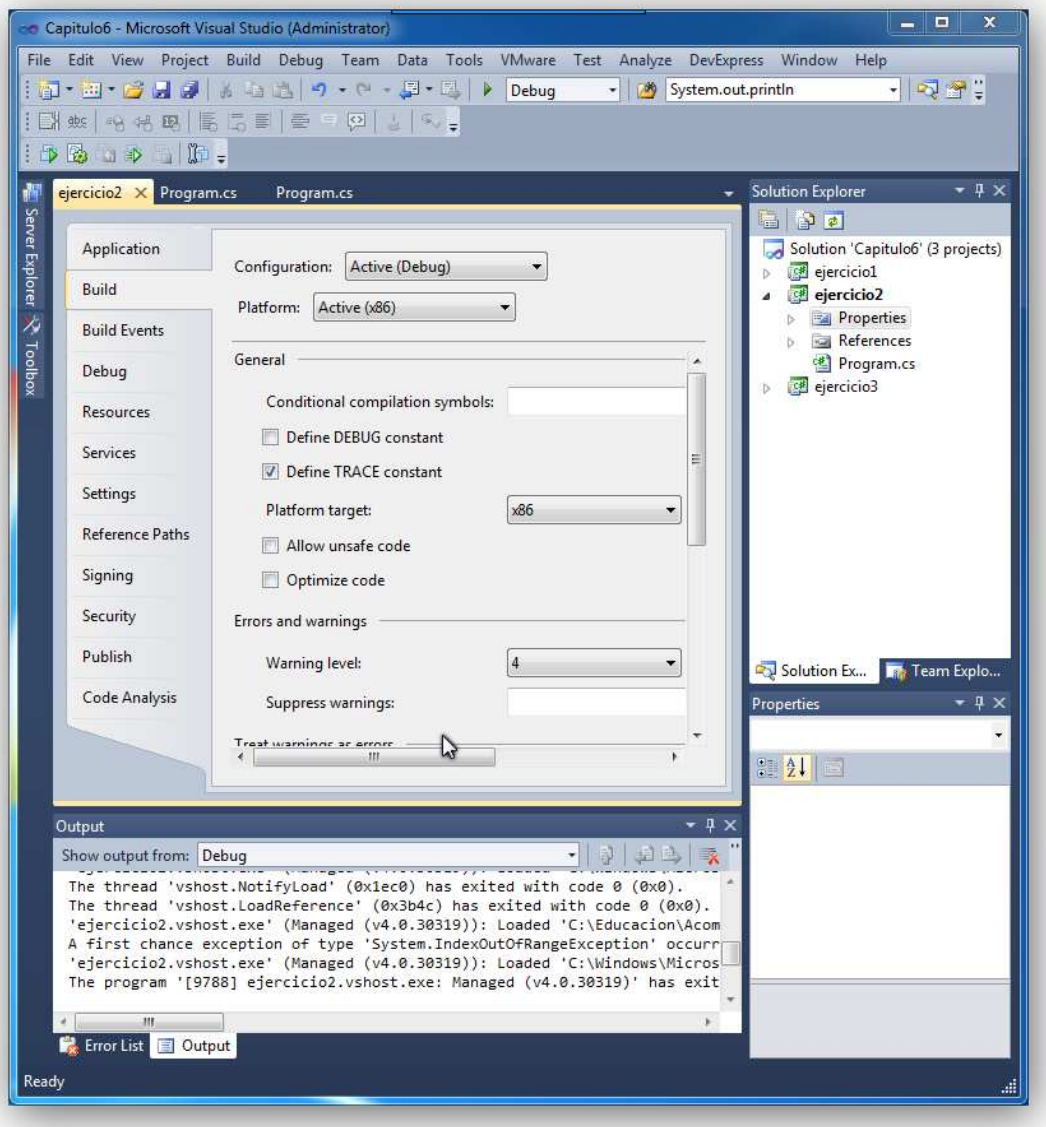
VB



5. Deshabilitar la opción de verificación de afirmaciones en tiempo de depuración. Para ello deshabilitar la marca que define la constante DEBUG en tiempo de compilación. Ir a las propiedades del proyecto y según el lenguaje, realizar el procedimiento adecuado:

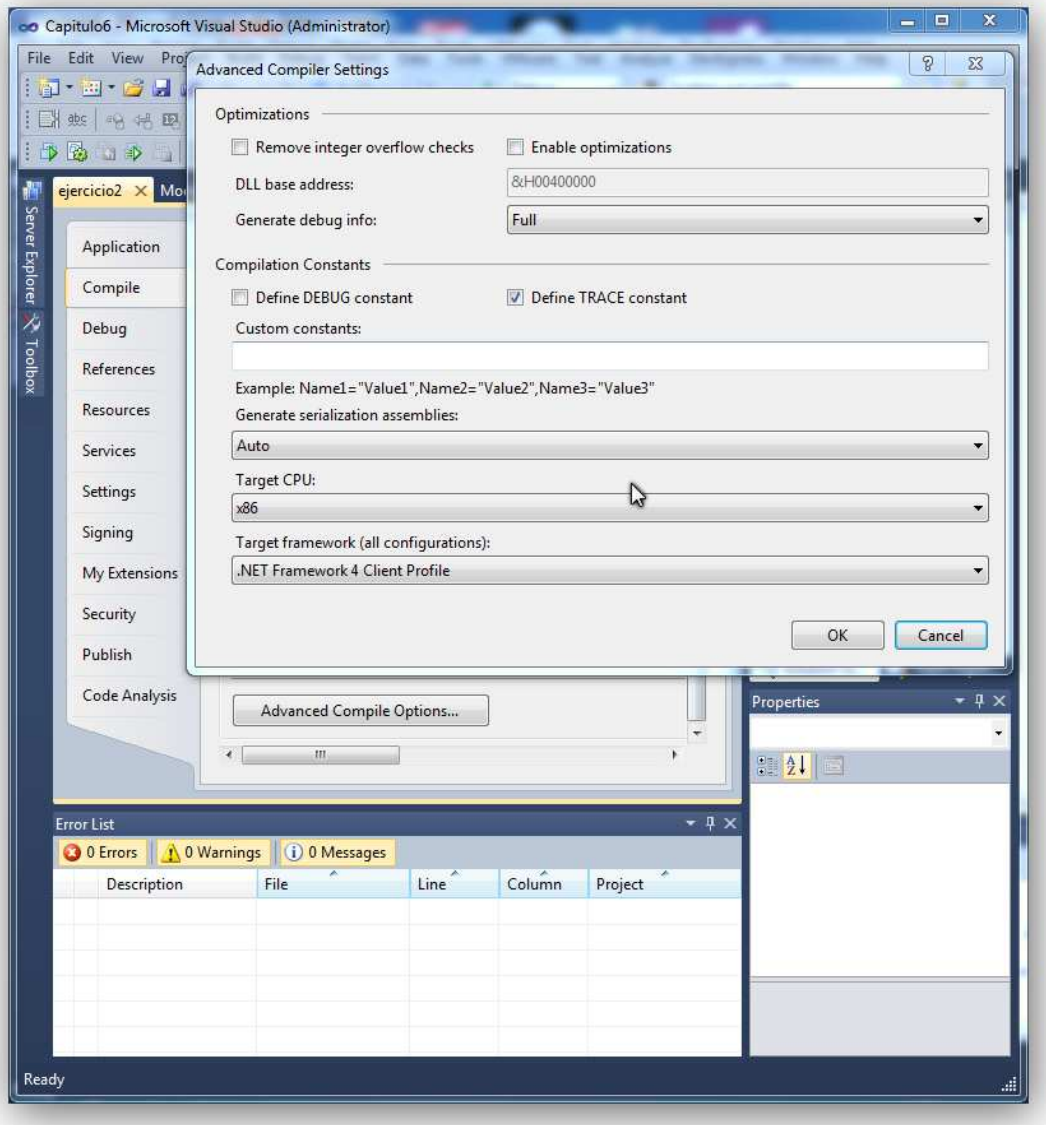
C#

Ir a la pestaña Build y desmarcar la casilla “Define DEBUG constant” como se muestra en la imagen:



VB

Ir a la pestaña Compile y presionar el botón Advanced Compile Options. En la ventana que se despliega, desmarcar la casilla “Define DEBUG constant” como se muestra en la imagen:

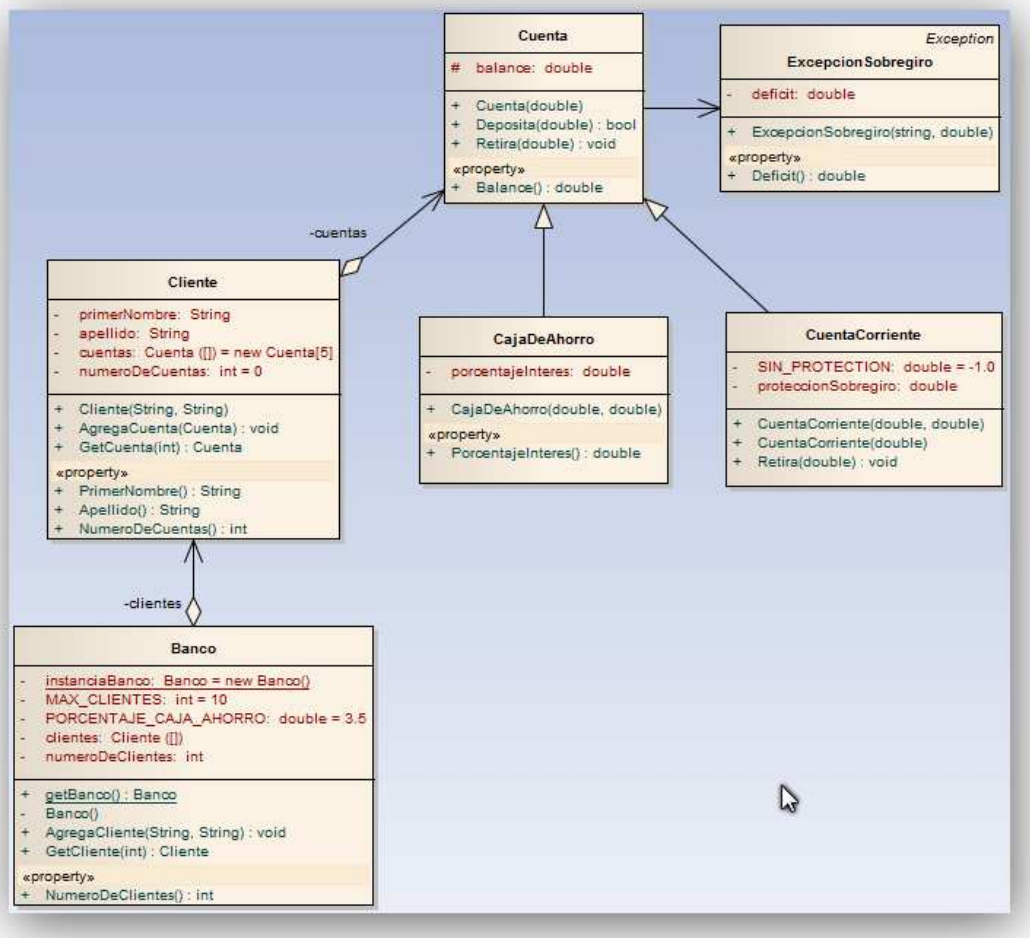


6. Ejecutar nuevamente el programa y verificar que su comportamiento es el mismo que tenía antes de incluir el método Assert.

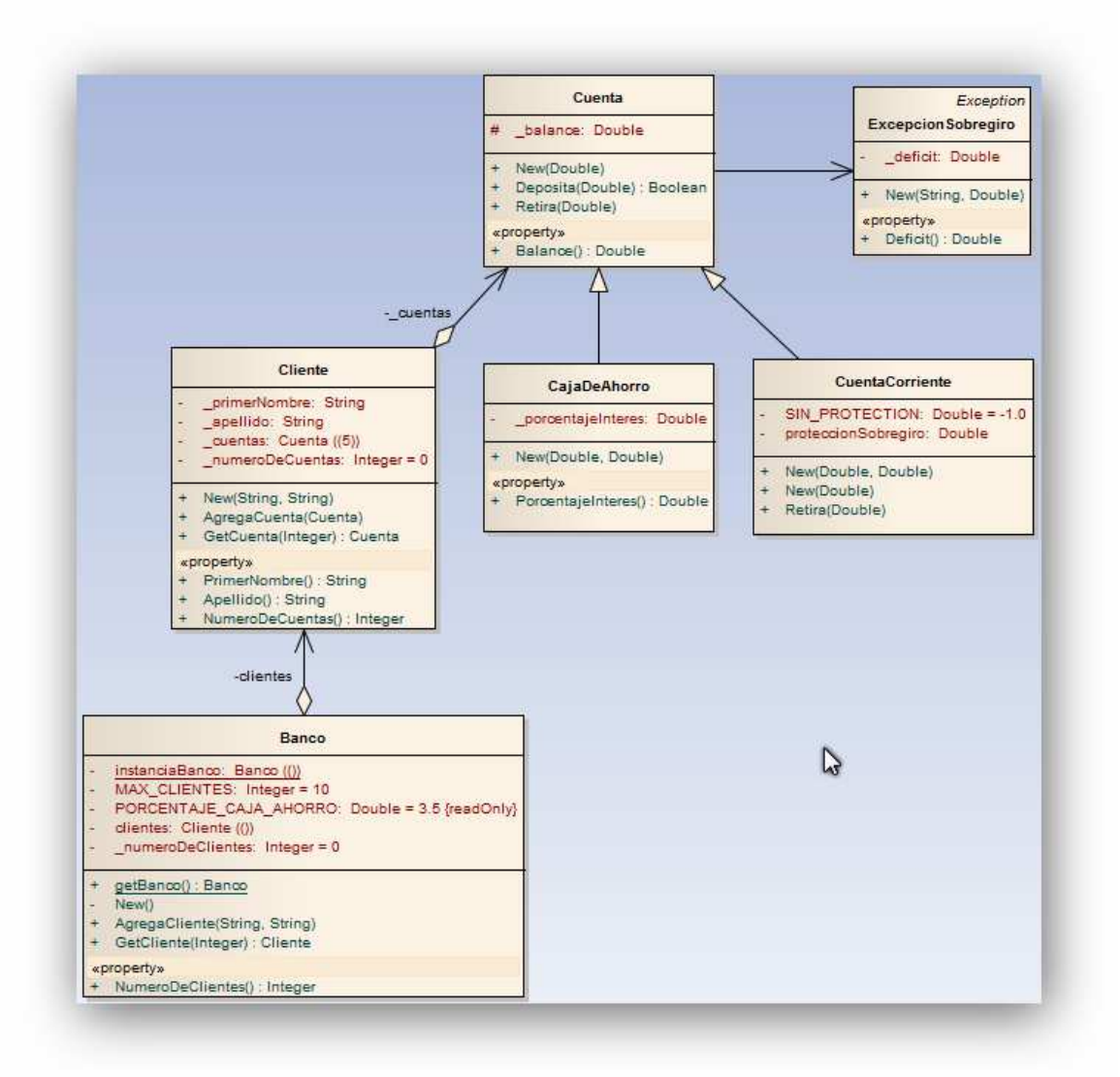
Ejercicio 3

En este ejercicio se creará una excepción llamada ExcepcionSobregiro la cual es lanzada por el método retira de la clase Cuenta. Las clases a obtener figuran en el siguiente diagrama UML

C#



VB



1. Abrir la solución del directorio Capitulo6 si no la tiene abierta ya.
2. Crear la clase ExcepcionSobregiro
3. Crear una clase pública llamada ExcepcionSobregiro en el paquete operacionesBancarias.dominio. Esta clase hereda de la clase Exception.
9. Agregar un atributo privado llamado déficit que pueda almacenar un `double`
4. Agregar una propiedad de sólo lectura y acceso público llamada Deficit.
5. Agregar un constructor público que reciba dos parámetros: mensaje y deficit. El parámetro mensaje deberá pasarse al constructor de la superclase. El parámetro déficit inicializa el atributo deficit.

Modificar la clase Cuenta

10. Modificar el método retira() :

   a. Volver a escribir el método para que no devuelva valores, (esto es, que sea `void` o `Sub` – C# o VB).
   b. Declarar que este método lanza una excepción llamada ExcepcionSobregiro.
11. Modificar el código para lanzar la nueva excepción especificando "Fondos Insuficientes" en el mensaje y el déficit (lo que le falta para estar dentro del acuerdo de giro descubierto).

Modificar la clase CuentaCorriente

12. Modificar el método retira() :
   a. Volver a escribir el método para que no devuelva valores, (esto es, que sea `void` o `Sub` – C# o VB).
   b. Declarar que este método lanza una excepción llamada ExcepcionSobregiro.
13. Modificar el código para lanzar una excepción si es necesario. Hay dos casos que necesitan ser manejados.
   a. Primero, si hay un déficit pero no se puede cubrir con la protección de giro en descubierto para la cuenta de ahorro. En este caso utilizar un mensaje "No hay protección por sobregiro".
   b. Segundo, la cantidad que almacena proteccionSobregiro es suficiente para cubrir el déficit, utilizar el mensaje "Fondos insuficientes para proteger el sobregiro"
14. Compilar y ejecutar el programa
15. Verificar que la salida sea:

```
El Cliente [Perez, Juan] tiene un balance en cuenta corriente de 200.0
con una protección por sobregiro de 500.00.
Cuenta Corriente [Juan Perez]: retira 150.00
Cuenta Corriente [Juan Perez]: deposita 22.50
Cuenta Corriente [Juan Perez]: retira 147.62
Cuenta Corriente [Juan Perez]: retira 470.00
Excepción: Fondos insuficientes para proteger el sobregiro   Deficit: -
470.0
El Cliente [Perez, Juan] tiene un balance en cuenta corriente de 0.0

El Cliente [Toma, Oscar] tiene un balance en cuenta corriente de 200.0
Cuenta Corriente [Oscar Toma]: retira 100.00
Cuenta Corriente [Oscar Toma]: deposita 25.00
Cuenta Corriente [Oscar Toma]: retira 175.00
Excepción: No hay protección por sobregiro   Deficit: -50.0
El Cliente [Toma, Oscar] tiene un balance en cuenta corriente de 125.0
```