Proyecto 3 evaluación

**Refactorizacion**:

* 3) Cambiados los nombres de controles gráficos que tenían nombre por defecto para aumentar la comprensión, el resto de variables, clases etc... Estaban correctos.
* 4) Retirados los inputbox .
  + Usadas cajas de texto usando la propiedad **TextBox.MaxLength**,
  + Validando los datos de entrada con **int.TryParse**. y usando las características del **int.Maxvalue**, no usado el **MinValue** porque en ninguno de los casos quiero que se inserten números negativos.
  + Usado el s **tring.IsNullOrEmpty()** para controlar las cadenas vacías.
* 5)Crear clases con la lógica de negocio de cada subsistema
  + A)Clases en el mismo namespace del susbistema
  + B)El nombre de la clase incluye la palabra lógica
  + C)En el caso de **Numeros Fibonacci y Numeros Primos** los métodos son static en el caso de **Numeros Primitiva** y **Numeros TorresDeHanoi**. Las clases son instanciables para poder acceder a sus métodos.
  + D)Usa excepciones para comunicar errores de la lógica al interfaz y try/catch para recuperarlos en los formularios.
  + E)Usados el out para devolver resultados en ningua de mis clases he necesitado usar el out para devolver resultados de los métodos, aunque en el caso de **Torres de Hanoi** uso un **ref** para devolver un texto que va variando, también usado out en los int.TryParse .
  + F)Diseña nuevas excepciones personalizadas para facilitar el control de errores o usa alternativas como la vista en clase , en la ultima parte del tema de test unitarios.
* 6)Documentar pruebas de caja negra con sus correspondientes casos de prueba.

**Pruebas de caja negra:**

* Relizamos pruebas de caja negra sobre el formulario con la **clase lógica de números Fibonacci**

1. Usando el textchanged le aplico restricciones para los datos que no podía controlar y eran no deseados.
2. Le añado también un maxlength de 10 ya que el máximo número permitido mide 10.
3. Uso el botón.enabled.false para no dejar introducir caracteres no desados y asi evitar problemas.
4. Con una condición controlo todos las clases de equivalencia de números negativos de esta forma no es necesario realizar la prueba de minValue.
5. Cualquier numero comprendido entre el 1 y maxValue será la clase de equivalencia de la **prueba numero 4**
6. Con los apartados 4 y 5 tambien conseguimos controlar la introducción de decimales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1 | “” | “” | “” | OK | Con la condición de que el botón este deshabilitado evitamos que dicha prueba falle porque si no se ingresa nada no deja que puedas usar la función de la aplicación |
| 2 | A | Debe introducir un número entero positivo no valen caracteres | Debe introducir un número entero positivo no valen caracteres | OK | Conforme introduces un carácter que no sea un número entero te envía un msBox para decirte que no es posible realizar y limpia el textbox |
| 3 | 0 | Debe introducir un numero positivo que no exceda el valor máximo | Debe introducir un numero positivo que no exceda el valor máximo | OK | La condición establecida es que siempre sea un número mayor que 0 por lo tanto no deja introducir un 0 |
| 4 | 1 | Los números Fibonacci hasta tu numero es :0, 1 | Los números Fibonacci hasta tu numero es :0, 1 | OK |  |
| 5 | 2 | Los números Fibonacci hasta tu numero es :0, 1,1,2 | Los números Fibonacci hasta tu numero es :0, 1,1,2 | OK | Caso especial dado que tiene que sacar dos veces el 1 como se muestra. |
| 6 | -1 | Debe introducir un número entero positivo no valen caracteres | Debe introducir un número entero positivo no valen caracteres | OK | No deja introducir números negativos con este caso realizamos la prueba de todos los números negativos |
| 7 | 2147483646 | Muestra todos los números Fibonacci comprendidos entre 0 y numintroducido | Muestra todos los números Fibonacci comprendidos entre 0 y numintroducido | OK | Este caso es el maxValue-1 |
| 8 | 2147483647 | Debe introducir un numero positivo que no exceda el valor máximo | Debe introducir un numero positivo que no exceda el valor máximo | OK | Este caso es el MaxValue.  Al introducir el maxvalue salte mbox y deshabilita el botón para no dar excepciones |
| 9 | 1.1 | Debe introducir un número entero positivo no valen caracteres | Debe introducir un número entero positivo no valen caracteres | OK | Controlados los números con decimales con la condición que no deje meter caracteres de ningún tipo solo números enteros. |

* Relizamos pruebas de caja negra sobre el formulario con la **clase lógica de La Primitiva**

1. Usando el textchanged le aplico restricciones para los datos que no podía controlar y eran no deseados.
2. Le añado también un maxlength de 2 ya que el máximo número es 49
3. Uso el botón.enabled.false para **probar suerte** para que no puedas jugar mientras no tengas los 6 numeros introducidos.
4. Con una condición controlo todos las clases de equivalencia de números negativos de esta forma no es necesario realizar la prueba de minValue.
5. Con la condición de que el numero no sea mayor de 49 controlamos todos los positivos hasta el 49 (CLASES DE EQUIVALENCIA)
6. Controlo todas las clases de equivalencia para todo lo que sea distinto a números enteros positivos con la condición que el número y sea un entero

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1 | “” | “” | “” | OK | Con la condición de que el botón este deshabilitado evitamos que dicha prueba falle porque si no se ingresa nada no deja que puedas usar la función de la aplicación |
| 2 | A | Inserte un valor valido | Inserte un valor valido | OK | Cuando intentas insertar algún careater prohibido en el array te da error y tienes que volver a ingrearlo. |
| 3 | 0 | Inserte un valor valido | Inserte un valor valido | OK | No deja ingresar cualquier número que no este comprendido entre 1 y 49 |
| 4 | 1 | El número ha sido añadido | El número ha sido añadido | OK |  |
| 5 | 1,1,2,3,4,5 | El número esta repetido ingrese otro | El número esta repetido ingrese otro | OK | Controlamos que no se repitan números |
| 6 | -1 | Inserte un valor valido | Inserte un valor valido | OK | No deja introducir números negativos con este caso realizamos la prueba de todos los números negativos |
| 7 | 49 | El número ha sido añadido | El número ha sido añadido | OK | Caso máximo de número permitido |
| 8 | 50 | Inserte un valor valido | Inserte un valor valido | OK | Caso máximo número permitido mas 1 |
| 9 | 1. | Inserte un valor valido | Inserte un valor valido | OK | Dado que el textLength esta limitado a 2 no deja ingresar números con decimales. |
| 10 | 1,2,3,4,5,6 | Ya has ingresado todos los números | Ya has ingresado todos los números | OK | Se activa el botón prueba suerte |

* Realizamos las pruebas de caja negra sobre **números primos Logica**

1. Usando el text.length =10 controlo el número máximo permitido.
2. Controlo que el número sea mayor de 0 y menor que minvalue de lo contrario el botón estará desactivado asi controlo las clases de equivalencia de números negativos.
3. Cuando el número no es válido vacía la matriz y no deja ingresar
4. Controlo que el textbox no este vacío de lo contrario el botón esta desactivado.
5. Con cualquier número comprendido entre 1 y max value controlo las clases de equivalencias de números positivos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1 | “” | “” | “” | OK | Con la condición de que el botón este deshabilitado evitamos que dicha prueba falle porque si no se ingresa nada no deja que puedas usar la función de la aplicación |
| 2 | A | El botón esta deshabilitado | El botón esta deshabilitado | OK | El botón esta desactivado con caracteres no pemitidos |
| 3 | 0 0  0 0 | El botón esta deshabilitado | El botón esta deshabilitado | OK | Tiene que ser mayor de 0 para que se active el boton |
| 4 | 1 4  4 3 | Numeros primos en posición 0,1 y posición 1,1 | Numeros primos en posición 0,1 y posición 1,1 | OK | Te da los números primos en las posiciones con eso controlamos la clase de equivalencia de números positivos |
| 5 | 2147483646  En cuanquiera | Devuelve los números primos en sus posiciones | Devuelve los números primos en sus posiciones | OK | Controlamos el rango por arriba del número máximo permitido |
| 6 | 2147483647  En cualquiera | El botón esta deshabilitado | El botón esta deshabilitado | OK | Controlamos el rango por arriba del número máximo no permitodo |
| 7 | 1.1 | El botón esta deshabilitado | El botón esta deshabilitado | OK | Controlamos que no se pueda meter ningún número decimal |
| 8 | -1 -1  -1 -1 | El botón esta deshabilitado | El botón esta deshabilitado | OK | Controlamos que en el momento que un número de la matriz sea negativo el botón este deshabilitado |

* Realizamos pruebas de caja negra sobre **Torres De Hanoi lógica**
  1. El número de discos con los que se juega estran entre 1 y 9
  2. El maxlength del texto esta limitado a 1, con eso controlamos cualquier número con decimal y,negativos y números superiores a 9.
  3. Tambien controla que cuando el textbox este vacio el botón esta desactivado.
  4. Cuando el numero ingresado es 0 da aviso y vacia el textbox.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1 | “” | “” | “” | OK | Con la condición de que el botón este deshabilitado evitamos que dicha prueba falle porque si no se ingresa nada no deja que puedas usar la función de la aplicación |
| 2 | A | Para jugar solo puede ser un número entre 1 y 9 | Para jugar solo puede ser un número entre 1 y 9 | OK | Vacía el textbox y te da aviso de no permitido |
| 3 | 0 | Para jugar solo puede ser un número entre 1 y 9 | Para jugar solo puede ser un número entre 1 y 9 | OK | Vacía el textbox y te da aviso de no permitido |
| 4 | 1 | Devuelve el número necesarios de movimientos | Devuelve el número necesarios de movimientos | OK | Controlamos el rango por abajo |
| 5 | 9 | Devuelve el número necesarios de movimientos | Devuelve el número necesarios de movimientos | OK | Controlamos el rango por arriba del número máximo permitido |
| 6 | -1 | Para jugar solo puede ser un número entre 1 y 9 | Para jugar solo puede ser un número entre 1 y 9 | OK | Vacía el textbox y te da aviso de no permitido |
| 7 | 1.1 | Para jugar solo puede ser un número entre 1 y 9 | Para jugar solo puede ser un número entre 1 y 9 | OK | Vacía el textbox y te da aviso de no permitido |

**Después de realizar las pruebas de caja negra me he dado cuenta que con el int.tryParse controlas el int.maxValue y he decidido retirarlos porque en algunos casos podían llevar a confusión**

7)Pruebas unitarias en las clases de lógica de negocio

Para **[TestClass] public class testFibonacci**

Datos:  
Se controlan todos los datos introducidos en el formulario ya sean negativos, números decimales o cualquier carácter, también se controla que el número introducido no supere el maxvalue.

Al limitar la entrada el único caso de prueba es alguno comprendido entre 1 y maxvalue.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1  **testFibo** | **111** | **"Estos son los números fibonacci hasta el número introducido: 0,1,1,2,3,5,8,13,21,34,55,89,"** | **"Estos son los números fibonacci hasta el número introducido: 0,1,1,2,3,5,8,13,21,34,55,89,"** | OK | Prueba global para cualquier caso |

Para **[TestClass] public class testPrimitiva**

Datos:

En estas pruebas comprobamos los números acertados tanto cuando es 0, cuando es entre 1 y 5 que es el mismo caso y cuando son 6, también comprueba si el número contenido esta en la lista para que no se repita.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1  **numPrimiRepetido** | **int numInicial = 1;**  **listaContiene.Add(1);** | **El número esta repetido ingrese otro** | **El número esta repetido ingrese otro** | OK | Se le añade el número para que sea el mismo que se intenta ingresar y devuelva repetido |
| 2  **numPrimiNoRepetido** | **int numInicial = 1;**  **listaContiene.Add(2);** | **El número ha sido añadido** | **El número ha sido añadido** | OK | Se le añade un número distinto al que ya hay añadido para ver si lo ingresa |
| 3  **sinAciertosPrimitiva** | **listaDondeJuegas.Add(2);**  **listaPremiada.Add(1);** | **No has acertado ninguno** | **No has acertado ninguno** | OK | Comprueba los números acertados y devuelve ninguno |
| 4  **seisAciertosPrimitiva** | **listaDondeJuegas.Add(1); listaDondeJuegas.Add(13); listaDondeJuegas.Add(21); listaDondeJuegas.Add(42); listaDondeJuegas.Add(33); listaDondeJuegas.Add(49);**  **listaPremiada.Add(1); listaPremiada.Add(13); listaPremiada.Add(21); listaPremiada.Add(42); listaPremiada.Add(33); listaPremiada.Add(49);** | **Has acertado todas YOU WIN!!!** | **Has acertado todas YOU WIN!!!** | OK | Devuelve dicho string cuando aciertas todas |
| 5  **aciertoMasDeCeroMenosDeSeis** | **istaDondeJuegas.Add(1);**  **listaPremiada.Add(1);** | **has acertado:**  **1,** | **has acertado:**  **1,** | OK | Cuando aciertas algún número siempre que sea menos de 6 devuelve cuales fueron los aciertos |

Para **[TestClass] public class testNumerosPrimos**

Al igual que en los otras aplicaciones también controlamos que no metan números negativos ni con decimales ni caracteres especiales y que este comprendido entre 0 y 1.

Con ello solo debemos controlar que sea primo o no y después que devuelve el string que queremos que con ese caso de prueba controlas el resto

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1  **testPrimosEsPrimo** | 11 | esPrimo=true | esPrimo=true | OK | Devuelve que si es primo |
| **2**  **testPrimosNoEsPrimo** | 8 | esPrimo=false | esPrimo=false | OK | Devuelve que no es primo |
| 3  **testMatrizPrimos** | **matrizPrimos[0, 0] = 2;**  **matrizPrimos[0, 1] = 8;**  **matrizPrimos[1, 0] = 11;**  **matrizPrimos[1, 1] = 22;** | **Los números primos dentro de la matriz son: \nde la posicion 0,0el valor: 2\n de la posicion 1,0el valor: 11\n** | **Los números primos dentro de la matriz son: \nde la posicion 0,0el valor: 2\n de la posicion 1,0el valor: 11\n** | OK | Coje los números que sean primos de la matriz y los muestra |

Para **[TestClass] public class testTorresDeHanoi**

Esta limitado para jugar entre 1 y 9 discos por lo tanto todos los casos de prueba entre 1 y 9 son comunes, el resto controlamos que no ingresen números negativos ni decimales ni caracteres especiales al igual que el 0.

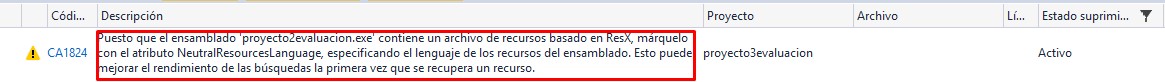
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número**  **prueba** | **Entrada** | **Salida** | **Esperado** | **Resultado** | **Comentario** |
| 1  **testTorresHanoi** | **3** | **Muevo el disco 1 de Origen a Destino\n Muevo el disco 2 de Origen a Auxiliar\n Muevo el disco 1 de Destino a Auxiliar\n Muevo el disco 3 de Origen a Destino\n Muevo el disco 1 de Auxiliar a Origen\n Muevo el disco 2 de Auxiliar a Destino\n Muevo el disco 1 de Origen a Destino\n** | **Muevo el disco 1 de Origen a Destino\n Muevo el disco 2 de Origen a Auxiliar\n Muevo el disco 1 de Destino a Auxiliar\n Muevo el disco 3 de Origen a Destino\n Muevo el disco 1 de Auxiliar a Origen\n Muevo el disco 2 de Auxiliar a Destino\n Muevo el disco 1 de Origen a Destino\n** | OK | Prueba global para cualquier caso  Las \n son saltos de linea |

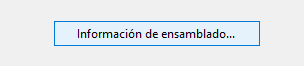
**9. Prueba las funcionalidades de análisis de código de Visual Studio**

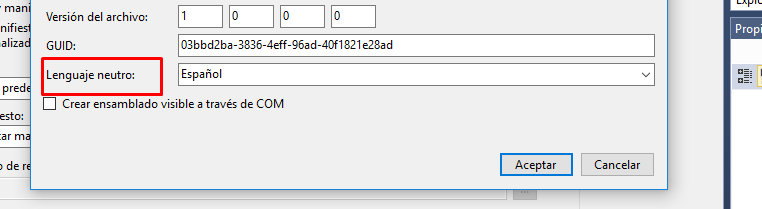
**para mejorar el código.**

Tenia un error por revisar textos vacíos que arregle con el string.isnullorempty que mandaste en el pdf.

CA1824: solucionando el análisis de código de el código mencionado







Solucionada agregando el idioma neutro español en la infromacion de ensamblado.

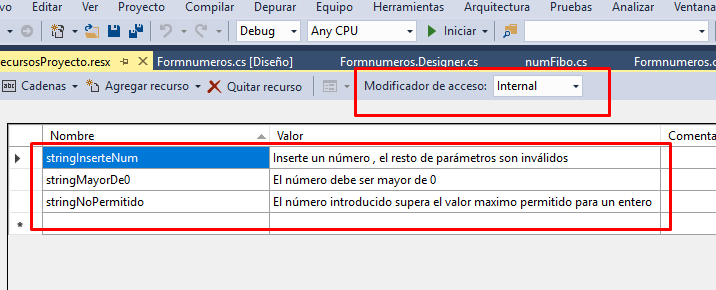
Siguiendo la guía de msdn:

<https://msdn.microsoft.com/library/bb385967.aspx>

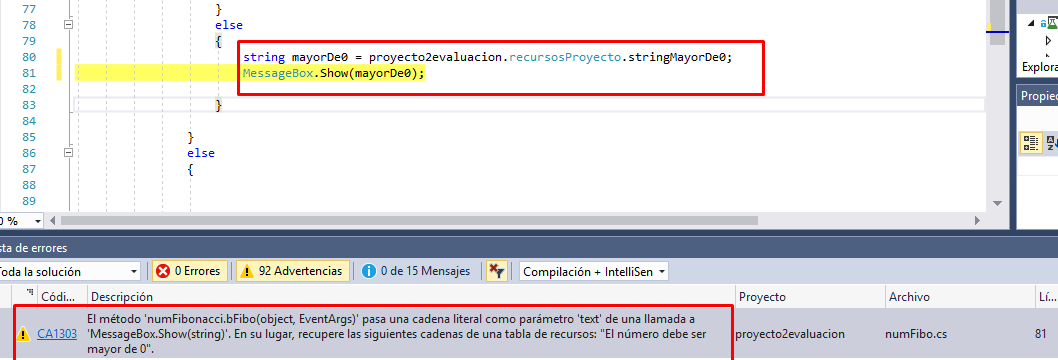
Analisis de código ca1303:

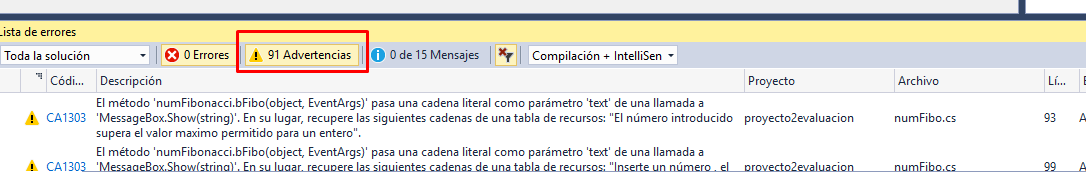
Solucionando agregando un elemento de recursos y metiendo ahí el string,

**Realizado sobre toda la clase de números Fibonacci.**



Se ha solucionado como se puede comprobar.





Prueba con intelliTest para probarlo:

Aquí el enlace que nos pasaste en castellano:

https://msdn.microsoft.com/es-es/library/dn823749.aspx

