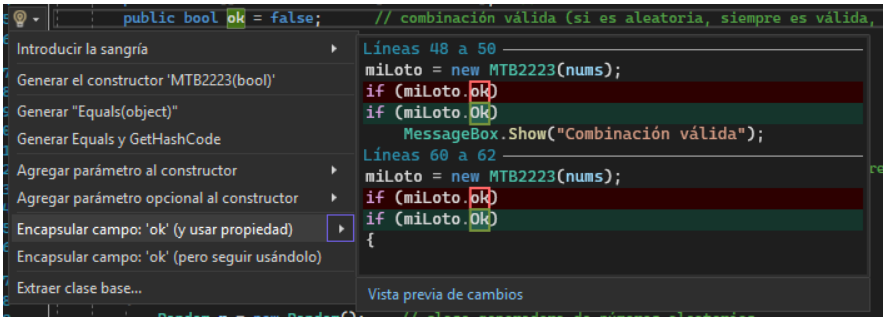
- Encontrar cinco errores de normas de estilo en el fichero *loto.cs,* indicando número de línea, error encontrado y solución.
    - Llaves (líneas 77 a 79): las llaves deben estar incluidas en el código (aunque el compilador no lo detecta como error). Se deben incluir incluso si hay una sola línea. La solución sería añadir llaves.
    - Salto de línea (línea 79): el if y el bloque interior deberían estar en líneas distintas.
    - Nombres no autodescriptivos (líneas 76 y 74): premi y a deberían tener nombres mas sencillos que no necesiten un comentario para especificar qué es, les cambio el nombre a combinacionGanadora y aciertos respectivamente.
    - Espaciado (líneas 77 y 78): después de punto y coma y entre operadores debería haber espacios. Como excepción se pueden dejar juntos los operadores unarios como i++. Se escribiría correctamente: for (int i = 0; i < MAX_NUMEROS; i++)
    - Nomenclatura (línea 74): el método comprobar debería seguir la nomenclatura PasCal, no camelCase, por lo que le cambio el nombre como en las anteriores; usando Ctrl + R, R a Comprobar. De esta forma se actualiza el nombre seleccionado y todas sus referencias.

  Estos errores no son los únicos que he encontrado, se repiten estos mismos y aparecen otros como el de tabulación de la línea 75 pero únicamente señalo los ya mencionados arriba. (En las líneas 14 y 15 los nombres podrían ser más descriptivos también)

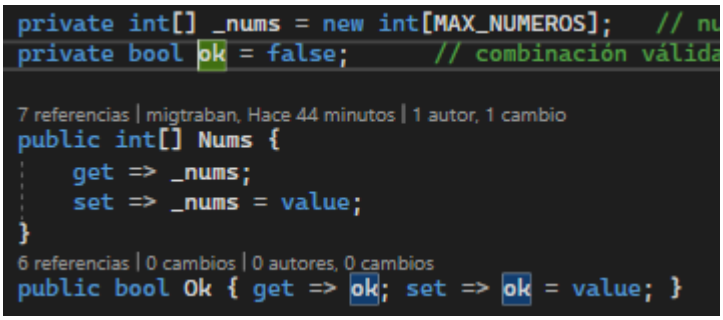
- Si existen, detectar y aplicar al menos tres patrones de refactorización (tanto en el fichero *Loto.cs* como en el fichero *Form1.cs*), indicando el patrón que se y, si es posible aplicarlo con Visual Studio, la opción que se usa.
    - Numero mágico: el numero 6 aparece muchas veces en el Form1.cs haciendo referencia al máximo de números (igual que en el loto.cs). Visual studio NO ofrece posibilidad de refactorización automática por el momento en este patrón.
    - Método largo (en este caso el constructor del formulario): dedica 6 lineas con dos puntos y coma cada una y hace que el constructor no se

quede limpio, por lo que extraemos el código del constructor. Visual studio nos permite hacerlo automáticamente con una opción llamada extraer método.

- Encapsulacion: la línea 15 de loto.cs contiene un miembro público y siguiendo el principio de encapsulación de la POO deberían ser privados. En caso de que se necesite acceder a ellos desde fuera se crea una propiedad. Visual studio nos permite una refactorización automática



Nótese que no solo genera la propiedad y la usa, si no que el propio miembro cambia a private:



- Realizar el diseño de pruebas (caja negra) para el constructor con parámetro de la clase *loto*.

Antes de nada, refactorizo levemente parte del código para que me quede más claro todo y pueda entender mejor el código.

El constructor recibe un parámetro int[] que debe tener números entre 1 y 49 y y no repetir números. Si se dan esas condiciones, se considera un valor valido. De no ser asi se considera un valor no válido, no se asigna el valor y la variable originalmente llamada ok pasa a ser false. Primero identifico los dominios. Para la primera condición,

al tratarse de un rango entre dos números hay 3 opciones (menor que 1, entre 1 y 49, mayor que 49). Para la segunda condición; podemos resumirla en una condición (booleano) si se repite o no. Para la realización de este ejercicio asumo que el array de entrada tiene 6 siempre elementos. De no asumirlo habría que contar con ello también en los casos de equivalencia e implementar pruebas en torno a ello.

- Constructor(int[] entrada)
  - 1 <= entrada[valor] <= 49 && no_repetidos -> único caso válido
  - valor_repetido en entrada -> caso no válido, no asigna
  - entrada < 1 || entrada > 49 -> caso no válido, no asigna

Para la realización de pruebas comprobaré si es válido o no basándome en la propiedad que encapsulé anteriormente.

Valores representativos

nombre	entrada	salida
ValorRepetido	1,1,2,3,4,5	No valido (valor repetido)
FueraRangoArriba	1, 2,4, 6,90	No válido (fuera rango por arriba)
FueraRangoAbajo	0, 1,2,3,4,5	No valido (fuera rango por abajo
ValorNormal	1,2,3,4,5,6	Valido (no implemento código)

Valores limite

A la hora de seleccionar valores límite para rangos, nos centramos en los dos limites (1 y 49) y a cada uno de ellos le hacemos lo siguiente: a-1, a, a+1.

Esto significa que tenemos que probar con los valores 0, 1, 2 y también con los valores 48, 49 y 50.

En la tabla anterior ya he probado los valores 0, 1 y 2 por lo que nos faltaría el segundo conjunto de valores.

nombre	entrada	salida
ValoresLimiteOK	1,2,4,48,49, 6	Valido (no implemento código
FueraRangoLimite	50, 1, 2, 3, 4, 0	No válido (fuera rango por arriba)