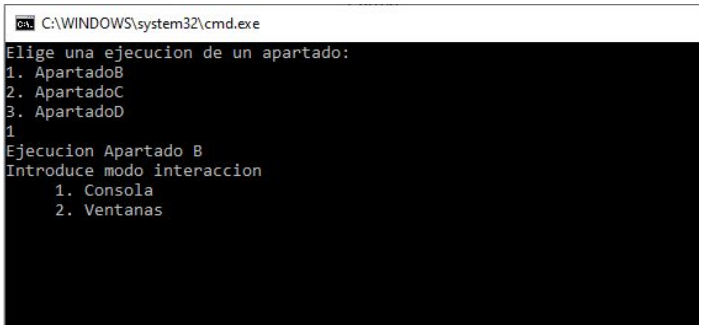
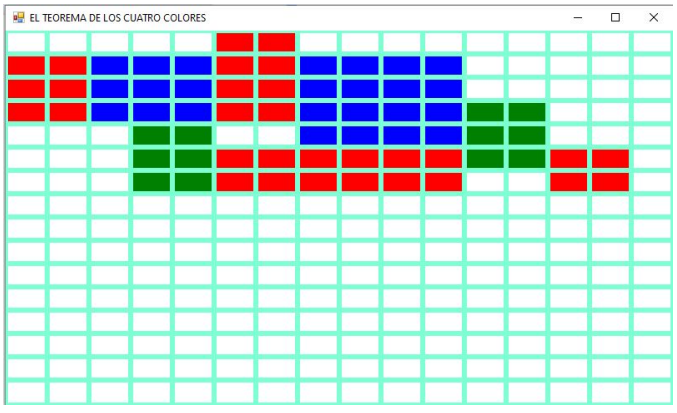
Práctica 2b
Salvador Martín Barcia
Borja Pérez Bernardos

a)

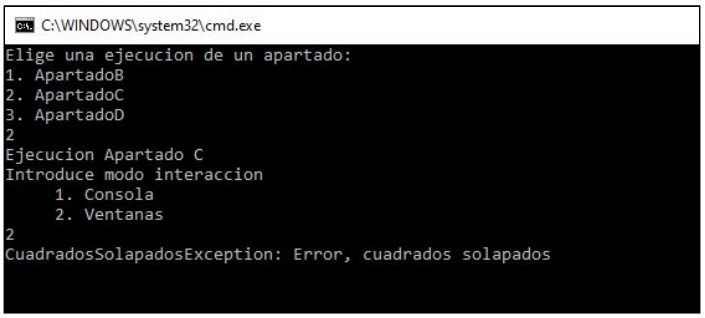
- Hemos decidido usar como base el código de los apartados b, c y d inmutables porque los apartados b y c los teníamos implementados como inmutables y los queríamos reutilizar.
- Nos hemos basado en algunas partes del código de los ejercicios de Windows Forms, en concreto el ejemplo de la calculadora, donde se usa una clase abstracta *Interfaz* y las clases *InterfazConsola* e *InterfazVentana*.
- Al ejecutar el programa, podemos elegir qué apartado queremos ejecutar y seguidamente si queremos la parte de consola (que es igual a la de la práctica 2b) o la parte de Windows Forms.



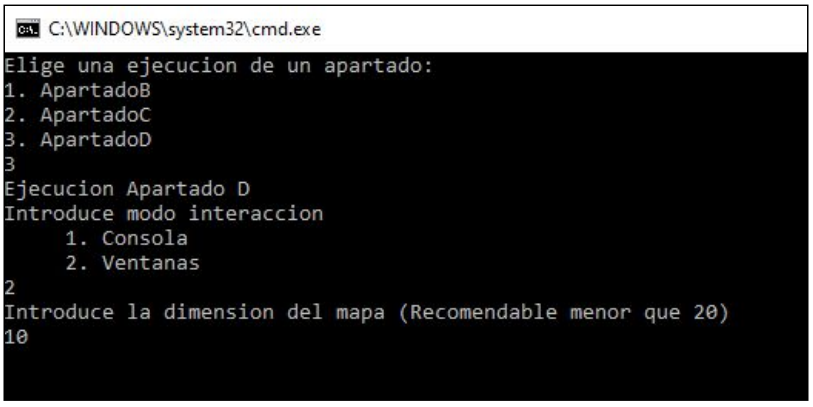
- **Apartado B(WF):**
  - Como en la práctica anterior, tenemos una lista de provincias y una lista de colores con las cuales haremos una ejecución estática del programa. Al ejecutarse se mostrará una ventana donde veremos la solución.
  - En este apartado y en el C, hemos fijado la dimensión a 16x16 por comodidad.



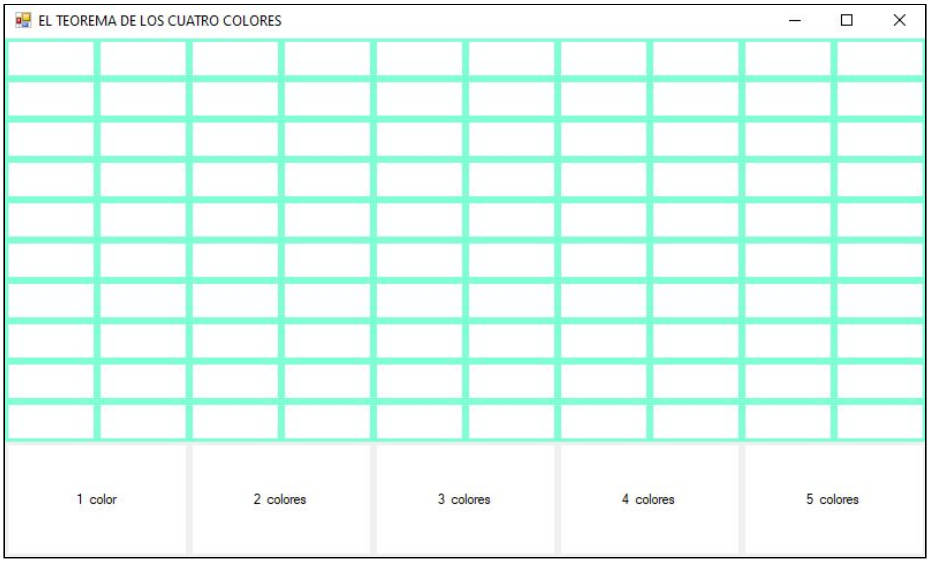
- **Apartado C(WF):**
    - En el apartado C hemos controlado el error de Cuadrados Solapados, cuando se produce el error se muestra por terminal.



- **Apartado D(WF):**

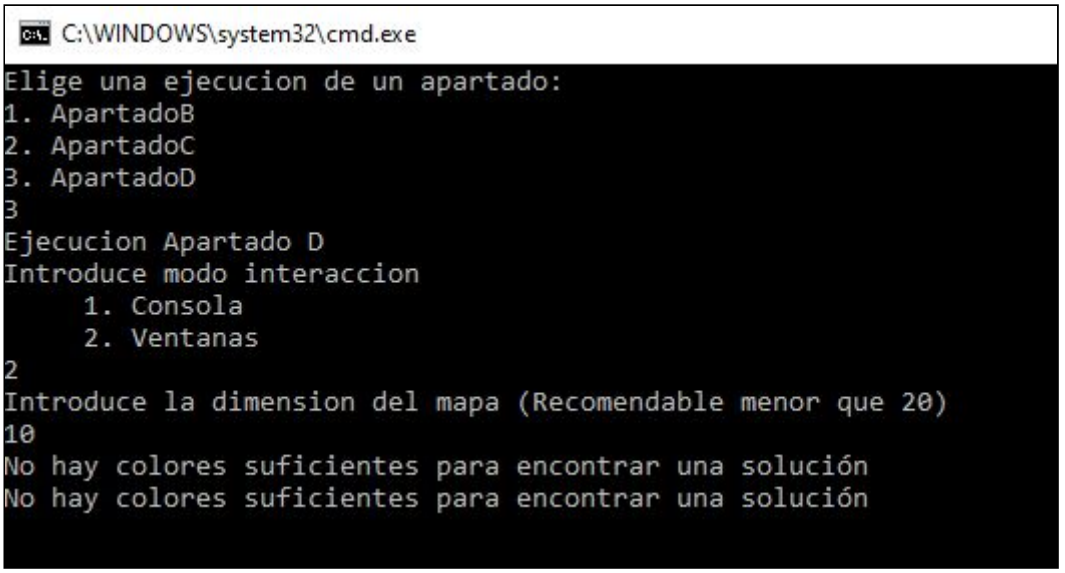


    - Después de haber elegido la dimensión, se mostrará una ventana como la siguiente:

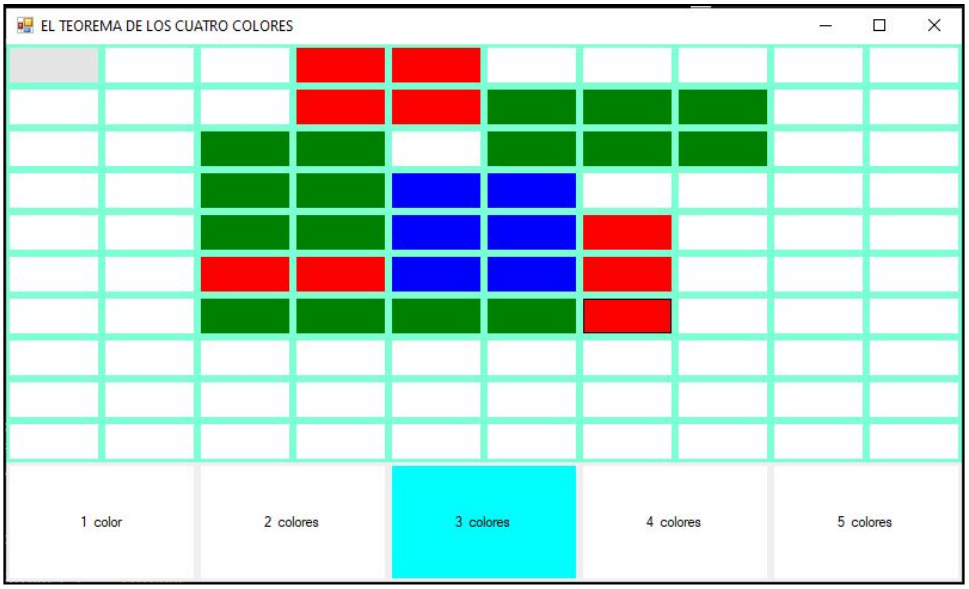


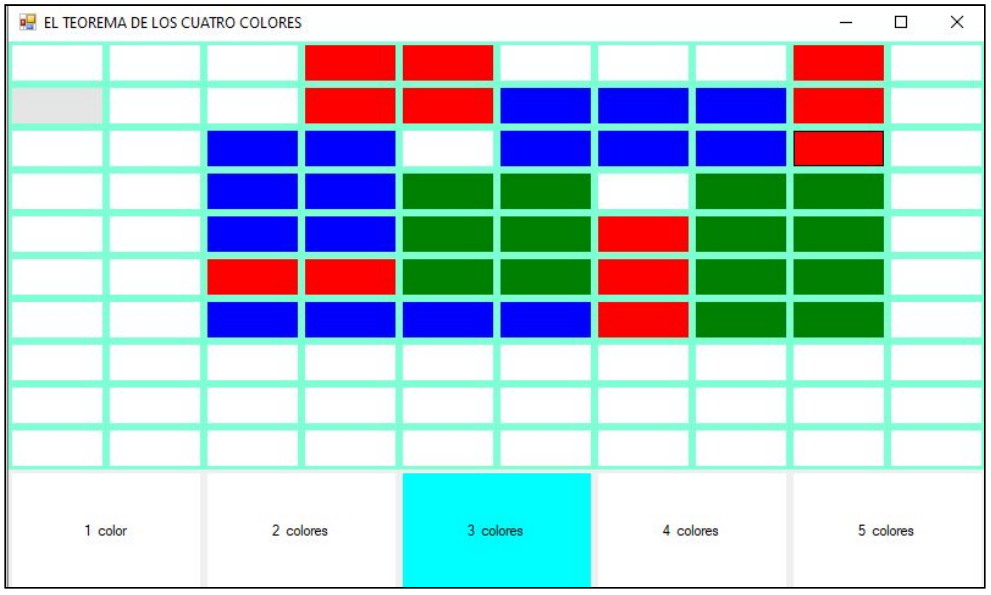
    - Para añadir provincias, el usuario deberá hacer un primer click en cualquiera de los cuadrados de la cuadrícula para elegir lo que sería la coordenada superior izquierda, y después debe seleccionar la coordenada inferior derecha, que deberá cumplir la condición de ser inferior y a la derecha de la primera coordenada para que esta se añada.
    - Automáticamente se ejecutará el algoritmo y el programa calculará una solución con la nueva provincia y la imprimirá.

- El usuario puede elegir la cantidad de colores a través de los botones que aparecen en la parte inferior. Si elige una cantidad insuficiente, en la Terminal se mostrará un mensaje indicando que no se pudo obtener una solución con esa cantidad de colores elegida.



- Si se añade una provincia y no hay un número suficiente de colores para encontrar una solución, no mostrará la solución pero si lo añadirá a la lista de provincias, y cuando se seleccione un número adecuado de colores, la solución se imprimirá.
- Cada vez que se introduce una provincia, se comprueba si está solapada con alguna otra.





b)

- La funcionalidad de este apartado la hemos hecho sólo en el apartado B de los B-C-D iniciales que había que implementar, ya que esta sería la misma para los apartados C y D.
- Hemos usado la función Except() de la librería Linq para sustituir la función diferenciaDosListas() que usábamos en la función Coloreados().
- Hemos creado un delegado llamado Fronteras para asemejar lo que hacíamos en Haskell cuando el data Mapa tenía una función como parámetro y al crear la variable andalucia de tipo Mapa, al constructor le pasábamos una lista de provincias y la función encontrarFronteras, de forma que conseguimos que el constructor de Mapa y la función coloresFrontera sean de orden superior.
- Hemos cambiado la función *MosaicoInicial* para que devuelva un *IEnumerable<List<Char>>* ya que utilizamos las sentencias yield return y yield break. Todos los que reciben un mosaico lo hemos por *IEnumerable<List<Char>>*. Y en *incluirProvincia* hemos cambiado la forma de acceder al mosaico para que funcione con *IEnumerable*.