Programación
PRACTICA 5
Sudoku

1. OBJETIVO

La práctica consiste en la elaboración de un programa que permita al usuario completar un "sudoku". Los objetivos son:

- Trabajar con programas diseñados en varias capas: la interfaz o vista por un lado y el modelo que dirige la vista por otro.
- Poner en práctica los conceptos vistos sobre matrices, recorrido y búsqueda.

2. EL JUEGO (FUENTE: WIKIPEDIA)

El sudoku se presenta normalmente como una tabla de 9 × 9, compuesta por subtablas de 3 × 3 denominadas "regiones" (también se le llaman "cajas" o "bloques").

Algunas celdas ya contienen números, conocidos como "números dados" (o a veces "pistas"). El objetivo es rellenar las celdas vacías, con un número en cada una de ellas, de tal forma que cada columna, fila y región contenga los números 1–9 solo una vez.

Además, cada número de la solución aparece solo una vez en cada una de las tres "direcciones", de ahí el "los números deben estar solos" que evoca el nombre del juego.

3. EL PROGRAMA.

El programa está compuesto de dos clases:

- La clase InterfazSudoku:
    - Contiene el interfaz gráfico que utilizará el usuario para completar el juego.
    - Se proporciona ya implementado.
    - Se apoya en la clase Sudoku.
- La clase Sudoku:
    - Ésta es la clase que hay que completar.
    - Contiene la tabla de números que conforman el juego durante su realización, así como una serie de métodos para actuar sobre ellos. El interfaz de usuario realiza llamadas a los métodos de esta clase para ir completando los números y hacer comprobaciones.

4. INTERFAZ GRÁFICO Y FUNCIONAMIENTO DEL PROGRAMA.

- El programa se entrega completamente funcional para que se pueda observar cuál es su comportamiento en respuesta a las acciones del usuario. No se detalla aquí su funcionamiento porque resulta bastante intuitivo y obvio, pero se puede probar el programa si se tiene alguna duda.
- Para que el programa sea funcional y al mismo tiempo no desvelar cuál es el código que resuelve la práctica se ha hecho lo siguiente:
    - Se ha incorporado al proyecto la clase Sudoku ya resuelta. De esta clase, el proyecto contiene el archivo binario (compilado) pero no el archivo fuente con el código Java. La clase Sudoku resuelta se incorpora en un paquete distinto, llamado "resuelto"

- En la clase InterfazSudoku se puede indicar que se use la clase ya resuelta o la que tienes que resolver tú cambiando la siguiente línea:
    - Para usar la versión ya resuelta y poder probar:
        - `import resuelto.Sudoku;`
    - Para usar tu versión:
        - `import modelo.Sudoku;`

5. LA CLASE SUDOKU

Consta de en una matriz de 9 x 9 números enteros. Las celdas vacias se representan por un 0. Además la clase tiene los métodos que se describen a continuación:

- **`public Sudoku()`**

  Crea el sudoku vacio, es decir, con ceros en todos sus elementos.

- **`public Sudoku(int datos[][]){`**

  Crea sudoku a partir de los datos de una matriz que se recibe como parámetro. Se supone que la matriz recibida tiene las dimensiones adecuadas y contiene valores correctos (entre 0 y 9).

- **`public Sudoku(String nombreFichero) throws FileNotFoundException{`**

  Crea sudoku a partir de los datos que se encuentran en el fichero cuyo nombre se indica como parámetro.

  El fichero contendrá 81 datos correctos, separados por espacios en blanco o saltos de línea.

  Para leerlos se utilizará la clase Scanner (asociándola al fichero) y el método nextInt().

  Se supone que el contenido del fichero es correcto.

  Si el fichero indicado no existe se lanzará la excepción que aparece en la cabecera del método.

- **`public void setDatos(int datos[][]){`**

  Método setter. Asigna al sudoku los datos de la matriz que se recibe como parámetro. Se supone que la matriz tiene las dimensiones adecuadas y contiene valores correctos (0 a 9)

- **`public int[][] getDatos()`**

  Método getter: Devuelve la matriz con los números del sudoku.

- **`public int getDato(int fila, int columna)`**

  Devuelve un dato del Sudoku: el de la fila y columna que se indica. La primera fila y primera columna son la 0.

- **`public void setDato(int fila, int columna, int valor){`**

  Modifica el valor del dato cuya fila y columna se indica. La primera fila y columna son la 0.

- `public boolean validarElemento(int fila, int columna){`

  Devuelve true si el elemento de la fila y columna indicados es válido y false en caso contrario.

  Un elemento es válido si:

  - No está repetido dentro de la misma fila.
  - No está repetido dentro de la misma columna.
  - No está repetido dentro de la misma región de 3x3 elementos.

  Si el valor del elemento es 0 siempre se considera válido, puesto que está sin completar.

- `public boolean resuelto(){`

  Devuelve true si el Sudoku se encuentra resuelto y false en caso contrario.

  El sudoku se encuentra resuelto si todos los elementos contienen algún valor (distintos de 0) y además todos los elementos son válidos (no hay repeticiones). Éste método tendrá, por tanto, que hacer uso del método validarElemento descrito anteriormente.

- `public void reiniciar(){`

  Devuelve el sudoku al estado en que se encontraba inmediatamente después de ejecutarse el constructor, sea cual sea el constructor que se ha usado para crearlo.

6. AMPLIACIÓN

La ampliación consiste en modificar el constructor public Sudoku() para que, en lugar de generar un sudoku vacío, genere un sudoku con números dados o pistas.

Parece demostrado que para que un sudoku con pistas tenga una única solución es necesario que el número de pistas sea al menos 17. Por tanto, generaremos 17 pistas con valores aleatorios y en posiciones aleatorias.

Las pistas generadas deben ser, por supuesto, correctas. Es decir, en el sudoku inicial generado no deben repetirse números en la misma fila, columna o región. y los números deberán estar entre 1 y 9.