Programación III. I.T. Informática de Sistemas y Gestión

Enunciado de la práctica

Curso 2010–11

1. Enunciado del problema

Sudoku es un rompecabezas matemático de colocación que se popularizó hace unos años en España.

El objetivo es rellenar una cuadrícula de 9×9 celdas dividida en cajas de tamaño 3×3 utilizando las cifras del 1 al 9, partiendo de algunos números ya dispuestos en determinadas celdas. Las reglas especifican que no se debe repetir ninguna cifra en una misma fila, columna o caja.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

En estos momentos podemos encontrar sudokus planteados en las secciones de pasatiempos de gran cantidad de periódicos, incluso asociados a varios tipos de concursos, o en páginas especializadas en internet.

El objetivo de la práctica es elaborar una aplicación que ayude a resolver un sudoku.

Fase1

Elaborar una clase *SUDOKU* que modele el tablero de juego. La clase debe ajustarse a la siguiente forma corta:

```
class interface SUDOKU
creation
  make_vacio
    -- Creación sin cifras
    -- Todas las posiciones serán modificables por el usuario
  make_from_file (f: TEXT_FILE_READ)
    -- Creación con las cifras indicadas en `f'
    -- El fichero debe contener 9 filas de 9 cifras cada una, separadas
    -- por espacios. La cifra 0 indica que la casilla está vacía
    -- Sólo las posiciones vacías serán modificables por el usuario
    require
      f /= Void;
      f.is_connected
feature(s) from SUDOKU
  -- Métodos de creación/Inicialización
  make_from_file (f: TEXT_FILE_READ)
    -- Creación con las cifras indicadas en `f'
    -- El fichero debe contener 9 filas de 9 cifras cada una, separadas
    -- por espacios. La cifra 0 indica que la casilla está vacía
    -- Sólo las posiciones vacías serán modificables por el usuario
    require
      f /= Void;
      f.is_connected
feature(s) from SUDOKU
  -- Consultas sobre el estado
  esta_correcto: BOOLEAN
    -- con las cifras actuales
```

```
  esta_completo: BOOLEAN
    -- con las cifras actuales
  cifras: INTEGER
    -- Número de cifras insertadas en el tablero
  cifra (f, c: INTEGER): INTEGER
    -- Cifra almacenada en la fila `f', columna `c', cero
    -- si la posición está vacía
    require
      indices_correctos(f,c)
    ensure
      Result >= 0 and Result <= 9;
      Result = 0 implies cifra_vacia(f,c)
  cifra_correcta (f, c: INTEGER): BOOLEAN
    -- ¿Es correcta la cifra de esta posición?
    -- Una cifra puede no ser correcta incluso cuando no es
    -- modificable por el usuario
    require
      indices_correctos(f,c)
  cifra_vacia (f, c: INTEGER): BOOLEAN
    -- ¿Está vacía esta posición?
    require
      indices_correctos(f,c)
    ensure
      Result implies cifra(f,c) = 0;
      Result implies cifra_modificable(f,c)
  cifra_modificable (f, c: INTEGER): BOOLEAN
    -- ¿El usuario puede cambiar esta cifra?
    require
      indices_correctos(f,c)
  indices_correctos (f, c: INTEGER): BOOLEAN
    -- ¿Corresponden los indices a una posición del tablero?
feature(s) from SUDOKU
  -- Modificación del estado
  mete (n, f, c: INTEGER)
    -- Inserta `n' en la posición (`f',`c')
    require
      indices_correctos(f,c);
      cifra_vacia(f,c);
      cifra_modificable(f,c);
      n_es_cifra: n >= 1 and n <= 9
    ensure
      not cifra_vacia(f,c)
  borra (f, c: INTEGER)
    -- Deja libre la posición (`f',`c')
    require
      indices_correctos(f,c);
      not cifra_vacia(f,c);
      cifra_modificable(f,c)
    ensure
      cifra_vacia(f,c)
end of SUDOKU
```

Los asertos que aparecen en la forma corta no son obligatorios, pero pueden facilitar la prueba de esta clase. Puesto que la forma corta sólo muestra las características públicas de la clase, en la implementación pueden utilizarse tantas características privadas como se considere oportuno.

Fase 1.1: Sólo para alumnos de Gestión

Los alumnos de I.T.I. Gestión deberán diseñar una batería de pruebas para la clase `SUDOKU`, aplicando las técnicas de prueba adecuadas, documentarla y automatizar la ejecución de los juegos de pruebas. Dicha automatización deberá estar contenida en una clase `PRUEBAS`.

Fase 2

Elaborar una interfaz basada en eGTK que permita jugar al sudoku, utilizando para ello el patrón observador. La clase `SUDOKU` de la fase anterior será utilizada como base para el sujeto, pero no pueden realizarse modificaciones sobre su código

La aplicación resultante debe permitir:

- Cargar un sudoku. Se puede optar por que el usuario deba introducir el nombre del fichero como argumento en la llamada al programa o por disponer en la interfaz gráfica de algún elemento que permita seleccionar el fichero
- Insertar una cifra en la posición modificable deseada por el usuario
- Borrar la cifra de una posición modificable indicada por el usuario

Además cuando en el sudoku aparezcan cifras repetidas en una misma fila, columna o cuadro, la interfaz mostrará claramente al usuario que existen cifras incorrectas, marcando su posición de una forma especial.

La clase lanzadora de esta fase se denominará obligatoriamente `APLICACION1`

Fase 3

En ocasiones, al resolver un sudoku deseamos borrar uno o varios de los movimientos que acabamos de realizar. Elaborar una nueva versión de la aplicación anterior, cuya clase lanzadora se denominará en este caso `APLICACION2` y cuya interfaz incorporará un botón «deshacer» que permita tantos «pasos atrás» como el usuario desee.

Para implementar esta fase es necesario encapsular los comandos de inserción y borrado del usuario en objetos, tal y como indica el patrón comando.

En la implementación de esta fase se pueden utilizar las clases de las fases anteriores, en particular `SUDOKU` de la fase 1, pero no pueden realizarse modificaciones sobre las mismas.

2. Condiciones de entrega y calificación de la práctica

Para la entrega de la práctica el código fuente será almacenado el directorio apropiado de `jair` conforme a la siguiente tabla:

Grupo	Directorio
Sistemas	`$HOME/p3sis1011`
Gestión	`$HOME/p3ges1011`

La clase `SUDOKU` a que hace referencia la fase uno deberá almacenarse en un fichero `sudoku.e`. Las clases lanzadoras de las dos aplicaciones deberán llamarse `APLICACION1` y `APLICACION2` y deberán estar almacenadas en sendos ficheros `aplicacion1.e` y `aplicacion2.e`. La clase `PRUEBAS`, lanzadora de la batería de pruebas para I.T.I. Gestión, deberá almacenarse en un fichero `pruebas.e`.

El sistema recogerá además todos los ficheros con extensión `.e`(adicionalmente los ficheros con extensión `.pdf` para alumnos de gestión) pero no otros tipos de ficheros como los que contengan iconos. Se recomienda pues, no utilizar iconos en la interfaz de las fases dos y tres.

Las prácticas que no se ajusten estrictamente[1] a estas condiciones de entrega, o que no compilen, serán consideradas como no presentadas.

[1] Atención a las minúsculas en los nombres de los ficheros

Aunque las prácticas tienen carácter individual, se admite el trabajo en parejas. Todo grupo de tres o más prácticas iguales será considerado fraudulento. En este sentido, el artículo 11.5 del Reglamento de Ordenación Académica determina que la realización fraudulenta de alguno de los ejercicios o trabajos exigidos para la evaluación de una asignatura supondrá una nota de cero, suspenso, en la convocatoria.

3. Fechas límite de entrega

La práctica será recogida el día del examen de la asignatura a partir de las ocho de la tarde.

4. La práctica en la convocatoria extraordinaria

Esta práctica es válida para la convocatoria extraordinaria, en las mismas condiciones y con límite de entrega en el día del examen extraordinario de la asignatura. Si en esta convocatoria no se entrega una nueva práctica la nota considerada será la obtenida en la convocatoria ordinaria.