PRÁCTICA 6: LA VENTANA INDISCRETA[1]

1. Introducción

Un problema clásico en desarrollo software es el de la *propagación de los cambios*, consistente en conseguir que cuando un elemento de un sistema software cambie, todos los elementos que dependen o están relacionados con dicho elemento modificado también cambien. Este problema recurrente ha dado recientemente lugar a la aparición de un nuevo paradigma de programación, conocido como *Reactive Programming*, y cuyo principal objetivo es dar solución a dicho problema.

Dentro de la orientación a objetos este problema se ha abordado mediante el patrón de diseño *Observer*, uno de los patrones quizás más populares y presentes en multitud de *frameworks* y librerías, como pueden ser librerías para la creación de interfaces gráfica (e.g., Swing).

El objetivo general de esta práctica es que el alumno adquiera familiaridad con dicho patrón, mediante su aplicación a un problema concreto. La siguiente sección describe los objetivos de esta práctica de manera más detallada.

2. Objetivos

Los objetivos concretos de esta práctica son:

(1) Comprender el funcionamiento general del Patrón *Observer*.
(2) Comprender cómo el Patrón *Observer* contribuye a la creación de interfaces gráficas.

Para alcanzar dichos objetivos, el alumno deberá realizar de forma satisfactoria una serie de actividades relacionadas con el problema que se describe a continuación.

3. Visualizador del Sistema de Archivos Sparrow.

Por alguna razón que se desconoce, dentro de la *Isla Muerta*, localización donde se deberá ejecutar el sistema de archivos *Sparrow*, si cuando accedemos a un enlace para recuperar su nombre, la clase correspondiente a los enlaces accede al elemento referenciado para recuperar su nombre, el rendimiento del sistema se ve gravemente afectado. Mientras los tripulantes de *La Perla Negra* tratan de resolver este misterio, se solicita a los ingenieros del sistema de archivos *Sparrow* que almacenen los nombres de los elementos referenciados dentro del propio enlace. Ello crea un nuevo problema que los ingenieros del sistema *Sparrow* tendrán que solucionar: cuando se cambia el nombre de un elemento, se han de cambiar los nombres almacenados en todos los enlaces que apuntan a dicho elemento.

[1] En homenaje a la magnífica película de Alfred Hitchcock del mismo nombre.

Además, para hacer más fácil la gestión del sistema de archivos Sparrow, se ha creado una interfaz gráfica que permite la visualización de un sistema de archivos Sparrow. La Figura 1 muestra dicha interfaz.

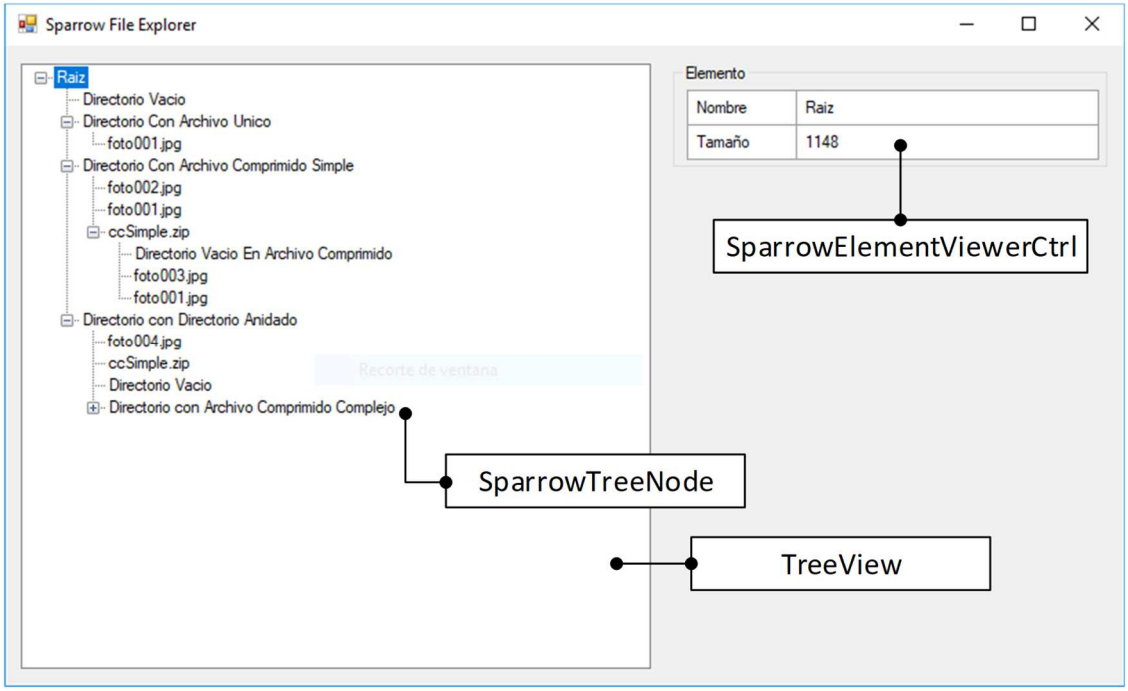

Figura 1. Interfaz Gráfica del Explorar de Archivos Sparrow

La interfaz gráfica está compuesta por un visualizador de árboles (a la izquierda de la Figura 1) más un visualizador de elementos del sistema de archivos (a la derecha de la Figura 1). El visualizador de árboles se implementa mediante la clase *TreeView*, proporcionada por la plataforma .NET. Esta clase Esta clase visualiza una serie de elementos de la clase *Node*. Para el caso concreto del sistema *Sparrow* se ha creado una subclase de esta clase *Node* denominada *SparrowNode*. Esta clase básicamente añade a la clase *Node* una referencia a un elemento concreto de un sistema de archivos Sparrow. De esta forma cada elemento de un sistema de archivos es visualizado como un *SparrowNode*.

El visualizador de elementos se implementa mediante un control propio definido mediante la clase *SparrowElementViewerCtrl*. Esta clase contiene una referencia a un elemento del sistema de archivos *Sparrow*, más un método interno *displayElement* que es el que se encarga de refrescar los valores del componente gráfico.

Actualmente tanto los *SparrowNode* como la clase *SparrowElementViewerCtrl* no se actualizan cuando el elemento referenciado cambia. Se solicita por tanto a lo ingenieros del sistema que, mediante la aplicación del patrón Observer, resuelvan este problema.

4. Actividades.

El alumno, para alcanzar los objetivos perseguidos, deberá completar las siguientes actividades:

1. Añadir los enlaces al diseño creado para la práctica del patrón *Composite* en caso de que dicho diseño no los contenga actualmente. Si el alumno no sabe cómo hacerlo, éste deberá consultar con el profesorado de la asignatura, el cual le explicará con detalle como hacerlo.
2. Hacer que los enlaces guarden dentro su clase, en un atributo, el nombre del elemento al que apuntan.
3. Como consecuencia de la modificación del nombre de un elemento del sistema de archivos deberán actualizarse los nombres almacenados en todos los enlaces que apuntan a dicho elemento. Para ello, hacer que los enlaces sean observadores de los elementos a los cuales apuntan.
4. Por compatibilidad con la interfaz gráfica a integrar, hacer que los enlaces permitan modificar su nombre. Cuando se modifica el nombre de un enlace, se modifica realmente el nombre del elemento al que apuntan.
5. Descargar de la plataforma *moodle* el proyecto que contiene el esqueleto de la interfaz gráfica para el sistema de archivos *Sparrow*. Realizar las modificaciones necesarias en dicho proyecto para que compile y se pueda ejecutar correctamente.
6. Hacer que la clase *SparrowNode* y *SparrowElementViewerCtrl* sean observadores de los elementos del sistema de archivos que corresponda, de manera que cuando estos elementos del sistema de archivos se cambios, estos controles también cambien de manera automática.

NOTA: Junto con el proyecto *moodle* se proporciona otro formulario, implementado por la clase *FileNameEditor*, cuyo objeto es permitir de manera cómoda y amigable la modificación de los nombres actualmente existentes en el sistema de archivos. Por limitaciones de este editor, se recomienda no dar el mismo nombre a dos elementos del sistema de archivos, aunque estén sitos en directorios diferentes, ya que esto puede provocar comportamientos anómalos.

5. Pasos para Aplicar el Patrón *Observer*

1. Identificar la clase o clases a ser observadas.
2. Crear una interfaz común para los observadores.
3. Añadir métodos a dicha interfaz que permitan enviar notificaciones a los observadores relativas a cambios realizados en la clase observada. Se recomienda no ser escueto en los parámetros proporcionados a través de dichos métodos.
4. Añadir una colección de observadores en las clases a ser observadas.
5. Añadir, si fuese necesario, mecanismos para añadir y eliminar observadores de dicha colección.
6. Añadir un método *notify* a las clases observadas. Este método simplemente recorrerá la colección de observadores y les enviará la notificación que sea pertinente.

7. Hacer que siempre que se cambie algo en las clases observadas que deba ser notificado, se invoque el correspondiente método notify.
8. Hacer que las clases que deban cambiar cuando las clases observadas cambien implementen la interfaz de observación.

Pablo Sánchez Barreiro