



PRÁCTICA 3: LOS LADRONES DE CUERPOS¹

1. Introducción

Desde el punto de vista de la calidad externa, los productos software creados mediante la utilización de *patrones de diseño* no se distinguirían de los productos software creados sin la utilización de los mismos. Por ejemplo, supongamos que se creasen dos productos software con exactamente la misma funcionalidad, pero donde un producto se ha creado utilizando patrones de diseño y el otro producto sin utilizarlos. En este caso, ambos productos ofrecerían tanto unas capacidades como un rendimiento similar. En adelante denominaremos al producto que utiliza los patrones de diseño como el producto *patronizado* y al que los utiliza como el producto *sin patronizar*.

Estos productos no serían sólo iguales desde un punto de visto externo, sino que incluso, en algunas situaciones muy particulares², el producto *patronizado* podría ser ligeramente más lento y/o consumir más memoria que el producto *sin patronizar*. Es decir, en todo caso, el producto *patronizado* es peor que el que *sin patronizar*.

Por tanto, si, desde el punto de vista de la calidad externa, ambos productos son prácticamente indistinguibles y la utilización de los patrones de diseño en todo caso sólo sirve para empeorar, ¿cuál es entonces la ventaja aportada por su utilización?

La principal ventaja que suelen aportar los *Patrones de Diseño* está relacionada con la calidad interna de un producto software y es la de facilitar la su evolución y adaptación. Para ello los patrones de diseño suelen introducir en un producto software *puntos de variación* bien definidos que permiten extender de forma cómoda, sencilla y elegante un producto software con nuevas funcionalidades, sin necesidad además de modificar el producto ya existente.

Por tanto, volviendo a nuestro ejemplo, aunque externamente el producto *patronizado* pueda parecer exactamente igual que el producto *sin patronizar*, el primero es mucho más fácil de adaptar a nuevas situaciones y extender con nuevas funcionalidades. Es decir, en resumen, el producto *patronizado* es mucho más barato de mantener.

El primer objetivo de esta práctica es que el alumno entienda cómo y por qué los patrones de diseño ayudan a mejorar la calidad interna de un producto software. Para alcanzar este objetivo, el alumno deberá aplicar el patrón *Visitor* en una situación concreta. Se ha escogido el patrón *Visitor* por un ser un claro ejemplo de la situación descrita con anterioridad.

¹ En homenaje al clásico de la ciencia ficción "*La invasión de los ladrones de cuerpos*" (Don Siegel, 1956), donde unas extrañas criaturas se introducen en los cuerpos de los habitantes de un pequeño pueblo estadounidense, los cuales mantienen su apariencia externa pero cambian su comportamiento.

² Dadas las características del hardware actual, este fenómeno es prácticamente inapreciable en casi la totalidad de situaciones.



Siguiendo los principios del patrón *Visitor* podemos conseguir que sea posible incorporar nuevas funcionalidades a una jerarquía o estructura concreta de clases, sin necesidad de tener que modificar ninguna de esas clases.

Sin embargo, para aplicar el patrón *Visitor* debemos utilizar de un complejo y exótico mecanismo de *double dispatching*. Este mecanismo puede ser complejo de entender, sobre todo por parte de programadores noveles poco familiarizados con los patrones de diseño. Además, el *double dispatching* introduce una serie de niveles de indirección que ralentizarían la ejecución de la aplicación, aunque este fenómeno es inapreciable dadas las características del hardware actual.

Resumiendo, el objetivo de esta práctica es que el alumno comprenda parte de las ventajas e inconvenientes asociados a los patrones de diseño mediante la aplicación del patrón *Visitor*. Para alcanzar dicho objetivo, el alumno deberá satisfacer los subobjetivos que se detallan en la siguiente sección.

2. Objetivos

Los objetivos concretos de esta práctica son:

1. Comprender el funcionamiento del patrón *Visitor*.
2. Comprender el concepto de *punto de variación* y *variante*.
3. Ser capaz de, utilizando el patrón *Visitor*, introducir *puntos de variación* dentro de una estructura de clases de manera que se puedan incorporar de nuevas funcionalidades a dicha estructura.
4. Ser capaz de, utilizando el patrón *Visitor*, extender la funcionalidad de una estructura de clases *visitable* mediante la implementación de nuevas variantes.

Para alcanzar dichos objetivos, el alumno deberá aplicar el patrón *Visitor* en la situación que se describe a continuación.

5. Visualización del Sistema de Archivos *Sparrow*

El sistema de archivos *Sparrow* (ver Práctica 2) debe permitir visualizar su estructura en forma arbórea de acuerdo a dos formatos distintos: el formato *compacto* y el formato *extendido*.

En ambos casos, sólo se debe mostrar un elemento por línea. Por cada elemento, además se debe mostrar su nombre. Además, cuando se muestre un directorio, deberá mostrarse debajo su contenido. Dicho contenido se mostrará tabulado con tres espacios con respecto al nombre del directorio.

Si se utiliza el formato compacto, el contenido de los archivos comprimidos no se visualizaría. Por el contrario, cuando se utiliza el formato desplegado, si se mostraría el contenido de un archivo comprimido. Al igual que en el caso de los directorios, este contenido estará tabulado con tres espacios con respecto al nombre del archivo comprimido.



Además, en el formato extendido, delante el nombre de cada elemento se mostrará una letra que indique de qué tipo de elemento se trata. Como letra se utilizará la “d” para los directorios, la “e” para los enlaces directos, la “c” para los archivos comprimidos, la “e” para los enlaces directos y la “f” para los archivos.

La Figura 1 muestra un ejemplo de visualización para el formato desplegado.

```
d Raiz
  d Directorio Vacio
  d Directorio Con Archivo Unico
    f foto001.jpg
  d Directorio Con Archivo Comprimido Simple
    f foto002.jpg
    e foto001.jpg
    c ccSimple.zip
      d Directorio Vacio En Archivo Comprimido
        f foto003.jpg
        e foto001.jpg
  d Directorio con Directorio Anidado
    f foto004.jpg
    e ccSimple.zip
    e Directorio Vacio
  d Directorio con Archivo Comprimido Complejo
    f foto005
    f foto006
    c ccComplejo.zip
      c ccAnidada.zip
        f foto007.jpg
        f foto008.jpg
```

Figura 1. Ejemplo de visualización de Sistema de Archivos.

Para aplicar el patrón *Visitor* a este problema concreto, se deberán realizar las actividades que se describen en la siguiente sección.

3. Actividades

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

1. Hacer que jerarquía de clases creada en la práctica dedicada al patrón *composite* sea *visitable* por las funciones de impresión descritas. Para hacer que la jerarquía sea *visitable*, el alumno deberá crear un visitante abstracto y añadir los métodos para aceptar visitantes, con su correspondiente implementación, a la jerarquía existente de clases. Las funciones de impresión deberán serializar un sistema de archivos como una cadena de caracteres, de manera que dicha cadena se pueda imprimir en diferentes salidas, tales como un monitor o una impresora. Por tanto, las funciones de impresión o visualización deben retornar una cadena de caracteres.
2. Implementar un visitante concreto para el formato compacto, sin tener en cuenta la necesidad de tabular el contenido de los directorios.



3. Crear un programa de prueba que verifique el correcto funcionamiento del visitante implementado para el sistema de archivos de la Figura 1.
4. Implementar un visitante concreto para el formato extendido, sin tener en cuenta la necesidad de tabular el contenido de los directorios.
5. Crear un programa de prueba que verifique el correcto funcionamiento del visitante implementado para el sistema de archivos de la Figura 1.
6. Modificar los visitantes implementados para que tengan en cuenta el requisito relativo a la tabulación de los elementos contenidos en directorios o archivos comprimidos.
7. Con los programas de prueba implementados anteriormente, verificar el correcto funcionamiento de las modificaciones realizadas para soportar las tabulaciones.

La realización de estas actividades se evaluará y calificará de acuerdo a los criterios que se definen en la siguiente sección.



5. Criterios de Evaluación y Aclaraciones

La práctica se entregará a través de la plataforma *moodle* siguiendo las instrucciones en ella proporcionadas.

La calificación de la práctica vendrá determinada por la calificación de los siguientes elementos:

- (1) Correcta implementación del visitante abstracto (0.5 puntos).
- (2) Correcta implementación de los métodos para aceptar visitantes (0.5 puntos).
- (3) Correcta implementación del visitante para el formato compacto (2 puntos).
- (4) Correcta implementación del visitante para el formato extendido (2 puntos).
- (5) Correcta implementación de los programas de prueba (0.5 puntos).
- (6) Ausencia de redundancias y código clonado (1 punto).
- (7) Correcta implementación de la funcionalidad para la tabulación (2.5 puntos).
- (8) Cumplimiento de la *Guía de Buenas Prácticas en Programación* (1 punto).

Pablo Sánchez Barreiro