PRÁCTICA 7: SIGUE EL CAMINO DE LAS BALDOSAS AMARILLAS[1]

1. Introducción

Los objetos de un sistema software no normalmente son entidades aisladas e independientes unas de otras. Por el contrario, estas entidades, como es natural, suelen conformar complejas redes de elementos que constituyen estructuras de diversos tipos con significados concretos en el mundo real. Por tanto, se hace necesario, para diversos fines, poder inspeccionar o recorrer dichas estructuras de manera sistemática y ordenada. Por ejemplo, podría ser necesario recorrer dichas estructuras para encontrar objetos concretos dentro de ellas o realizar ciertos cálculos a partir de la información contenida en sus elementos.

Resolver el problema anteriormente descrito es el objetivo principal del patrón *Iterator*. Dado que este problema es bastante común y la solución proporcionada por el patrón *Iterator* es elegante y está comúnmente aceptada, dicho patrón se ha ido integrando en los lenguajes de programación actuales, como Java, C# o Python. Dichos lenguajes proporcionan interfaces predefinidas para la especificación de iteradores e instrucciones especiales, como los bucles *foreach*, para una más cómoda manipulación de los diferentes iteradores que se generen.

El objetivo general de esta práctica es que el alumno adquiera familiaridad con el patrón *Iterator*, para lo que se aplicará dicho patrón 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 *Iterator*.
(2) Ser capaz de construir iteradores para recorrer estructuras de clases creadas de acuerdo con el patrón *Composite*.
(3) Comprender el funcionamiento general del patrón *State*.
(4) Ser capaz de aplicar el patrón *State* a la implementación de iteradores que necesiten recorrer estructuras compuestas como las generadas por el patrón *Composite*.

Para alcanzar estos objetivos, el alumno deberá realizar las actividades que se describen a continuación.

[1] Famosa frase extraída del clásico del cine “El Mago de Oz” (Victor Fleming, 1939).

3. Actividades.

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

1. Analizar las distintas fases o estados en los que se puede encontrar un iterador que necesite recorrer un elemento compuesto conforme al patrón *Composite*. Tomar como base el ejemplo de iterador proporcionado a través de la plataforma *Moodle*.
2. Crear un iterador para recorrer un sistema de archivos *Sparrow*, utilizando el patrón *State* para mejorar la mantenabilidad y escalabilidad de dicho implementador.
3. Crear, en una clase denominada *SparrowHelper*, una función estática que acepte un sistema de archivos *Sparrow* y una cadena de caracteres, y devuelva la colección de elementos de dicho sistema de archivos cuyo nombre contiene la cadena de caracteres pasada como parámetro.
4. Crear, en la clase *SparrowHelper*, otra función estática que acepte un sistema de archivos *Sparrow* y una cadena de caracteres y devuelva un elemento del sistema (el primero que encuentre) cuyo nombre contiene la cadena de caracteres pasado como parámetro. Para la creación de esta función queda terminantemente prohibido romper el bucle mediante una sentencia *return* o *break*.

4. Pasos para Aplicar el Patrón *Iterator*

Para instanciar el patrón *Iterator* se recomiendan seguir los siguientes pasos

1. Identificar la clase o conjunto de clases que se deben recorrer.
2. Hacer que dicha clase o conjunto de clases hereden de la interfaz *Iterable* que corresponda a cada lenguaje de programación[2]. Si el lenguaje de programación que se estuviese utilizando no contuviese ninguna interfaz de este tipo, simplemente hacer que la clase, o conjunto de clases, que se desea recorrer contenga un método *GetIterator()* que devuelva un iterador para el conjunto de objetos a recorrer.
3. Por cada tipo concreto de objeto a recorrer, crear una clase iterador para iterar sobre los elementos de dicho tipo concreto. Cada iterador concreto deberá implementar la interfaz *Iterator* que corresponda a cada lenguaje de programación[3]. Si el lenguaje de programación que se estuviese utilizando no contuviese ninguna interfaz de este tipo, crear una interfaz *Iterator* que tenga los métodos *first(), next(), currentItem()* e *isDone()*. A continuación, heredar de dicha interfaz e implementarla.

5. Pasos para Aplicar el Patrón *State*

Para instanciar el patrón *State* se recomiendan seguir los siguientes pasos

1. Identificar los diferentes estados para los cuales la implementación de los métodos de la clase varía.

[2] En C# dicha interfaz es IEnumerable<T>.
[3] En C# dicha interfaz es IEnumerator<T>

2. Identificar las transiciones entre esos estados y las condiciones bajo las cuales el objeto cambia de estado.
3. Crear una clase abstracta que contenga exactamente todos los métodos de la clase cuya implementación varía en función del estado en el cual se encuentre un objeto de dicha clase. Añadir el sufijo *State* al nombre de dicha clase.
4. Añadir una referencia *state* desde la clase con comportamiento variable a la clase abstracta creada en el punto anterior.
5. Crear, por cada estado en el cual se puede encontrar el objeto, una clase concreta que herede de la clase abstracta con sufijo *State*.
6. Detectar en cada clase concreta cuándo es necesario realizar el cambio de estado. Realizar dicho cambio de estado creando una instancia de la clase concreta correspondiente al nuevo estado y asignando esa instancia a la referencia *state* de la clase con comportamiento variable.
7. Implementar en cada clase concreta cada método de la clase abstracta teniendo en cuenta cómo debe reaccionar el objeto en el estado representado por la clase concreta donde se encuentra el método.

Pablo Sánchez Barreiro