Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas

Diseño Orientado a Objetos

SEMESTRE: Agosto Diciembre 2017

MTRO: Lic. Miguel Ángel Salazar Santillán

ACTIVIDAD: Tarea 6

Grupo: 006

ALUMNO: José Santiago Vázquez García

1746581

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN A 25 de noviembre de 2017

Introducción

Este documento se centra totalmente en los patrones de diseño en donde se contará con una descripción clara de lo que son y su funcionalidad.

El término patrones de diseño fue utilizado por primera vez por el arquitecto Christopher Alexander en el Libro “A Pattern Language: Towns, Buildings, Construction”, donde definió una serie de patrones arquitectónicos.

Ward Cunningham y Kent Beck trabajando en Smaltalk, diseñando interfaces de usuario, utilizaron algunas ideas de Alexander y con ello desarrollaron un pequeño lenguaje de patrones que serviría de guía a los programadores de Smaltalk.

Desde 1990 a 1994, Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides realizaron un primer catálogo de patrones de diseño. En 1994 publicaron el libro “Design Patters – Elements of Reusable Object-Oriented Software” que introduciría el termino de patrón de diseño en el desarrollo de software.

¿Qué es y para qué sirve un patrón?

Un patrón describe un problema repetitivo de nuestro entorno y describe también la solución de este, de forma que puede ser reutilizado infinidad de veces sin tener que hacer dos veces lo mismo. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí con el fin de brindar una solución ya probada y documentada a problemas de desarrollo de software que están relacionados a contextos similares.

Algunos de los objetivos de los patrones de diseño son el proporcionar catálogos de elementos reusables en el diseño de sistemas software, así como evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, además de formalizar un vocabulario común entre diseñadores, también estandariza el modo en que se realiza un diseño y facilita el aprendizaje de las nuevas generaciones de diseñadores condensando conocimientos ya existentes.

Los patrones de diseño se conforman de cuatro elementos, siendo estos **el nombre del patrón** el cual describe el problema de diseño, su solución y consecuencias en un par de palabras. **El problema** nos indica cuando se aplica el patrón, se explica el problema y su contexto. Algunas veces el problema incluye una lista de condiciones que deben cumplirse para poder aplicar el patrón. **La solución** describe los elementos que forma el diseño, sus relacione, responsabilidades y colaboraciones, como último elemento son las consecuencias, resultados de aplicar el patrón. Son muy importantes para la evaluación de diseños alternativos y para comprender los costes y beneficios de la aplicación del patrón.

Tipos de patrones

Existen una gran cantidad de patrones de diseño, sin embargo, los más conocidos o relevantes son los ***Patrones Creacionales, Estructurales y de Comportamiento.***

Los patrones creacionales facilitan la tarea de creación de nuevos objetos, con lo que el proceso de creación pueda ser desacoplado de la implementación del resto del sistema. Están basados en encapsular el conocimiento acerca de los tipos concretos que nuestro sistema utiliza, al igual que en ocultar cómo estas implementaciones concretas necesitan ser creadas y como se combinan entre sí.

Los patrones creacionales más conocidos son Abstract Factory la cual provee una interfaz para crear familias de objetos relacionados entre sí, sin especificar sus clases concretas. Singleton es otro patrón el cual asegura que una determinada clase sea instanciada una y solo una vez, proporcionando un único punto de acceso global a ella.

Los patrones Estructurales facilitan la modelización de nuestros softwares especificando la forma en la que unas clases se relacionan con otras. Algunos delos patrones estructurales más relevantes son Adapter el cual convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Bridge desvincula una abstracción de su implementación, de manera que ambas pueden variar de forma independiente.

Los patrones de Comportamiento se utilizan a la hora de definir como las clases y objetos interaccionan entre ellos. Se encuentran en la mayoría de los patrones, y se usan para gestionar algoritmos, relaciones y responsabilidades entre objetos. Algunos patrones de comportamiento son Command siendo objetos que encapsulan una acción y los parámetros que necesitan para ejecutarse. Interpreter define una representación para una gramática así como el mecanismo para evaluarla.

Ejemplo de patrón Creacional: Singleton

```
public sealed class Singleton
{
private static volatile Singleton instance; private static object syncRoot =
new Object(); private Singleton()
{
System.Windows.Forms.MessageBox.Show("Nuevo Singleton");
}
public static Singleton GetInstance
{ get {
if (instance == null)
{
lock(syncRoot)
{
if (instance == null)
instance = new Singleton();
} }
return instance;
}
}
}
```

Ejemplo de patrón Estructural: Adapter.

```
// RefinamientoAbstraccionA
public class Berlina : Vehiculo
```

```
{
// Atributo propio
private int capacidadMaletero;
// La implementacion de los vehículos se desarrolla de forma independiente public
Berlina(IMotor motor, int capacidadMaletero) : base(motor)
{
this.capacidadMaletero = capacidadMaletero;
}
// Implementación del método abstracto
public override void MostrarCaracteristicas()
{
Console.WriteLine("Vehiculo de tipo Berlina con un maletero con una capacidad de
" +
capacidadMaletero + " litros.");
}
}
```

Ejemplo de patrón de Comportamiento: Observador.

```
Public Class Articulo
Delegate Sub DelegadoCambiaPrecio(ByVal unPrecio As Object)
Public Event CambiaPrecio As DelegadoCambiaPrecio
Dim _cambiaPrecio As Object
Public WriteOnly Property Precio()
Set(ByVal value As Object)
_cambiaPrecio = value
```

```
RaiseEvent CambiaPrecio(_cambiaPrecio)

End Set

End Property

End Class
```

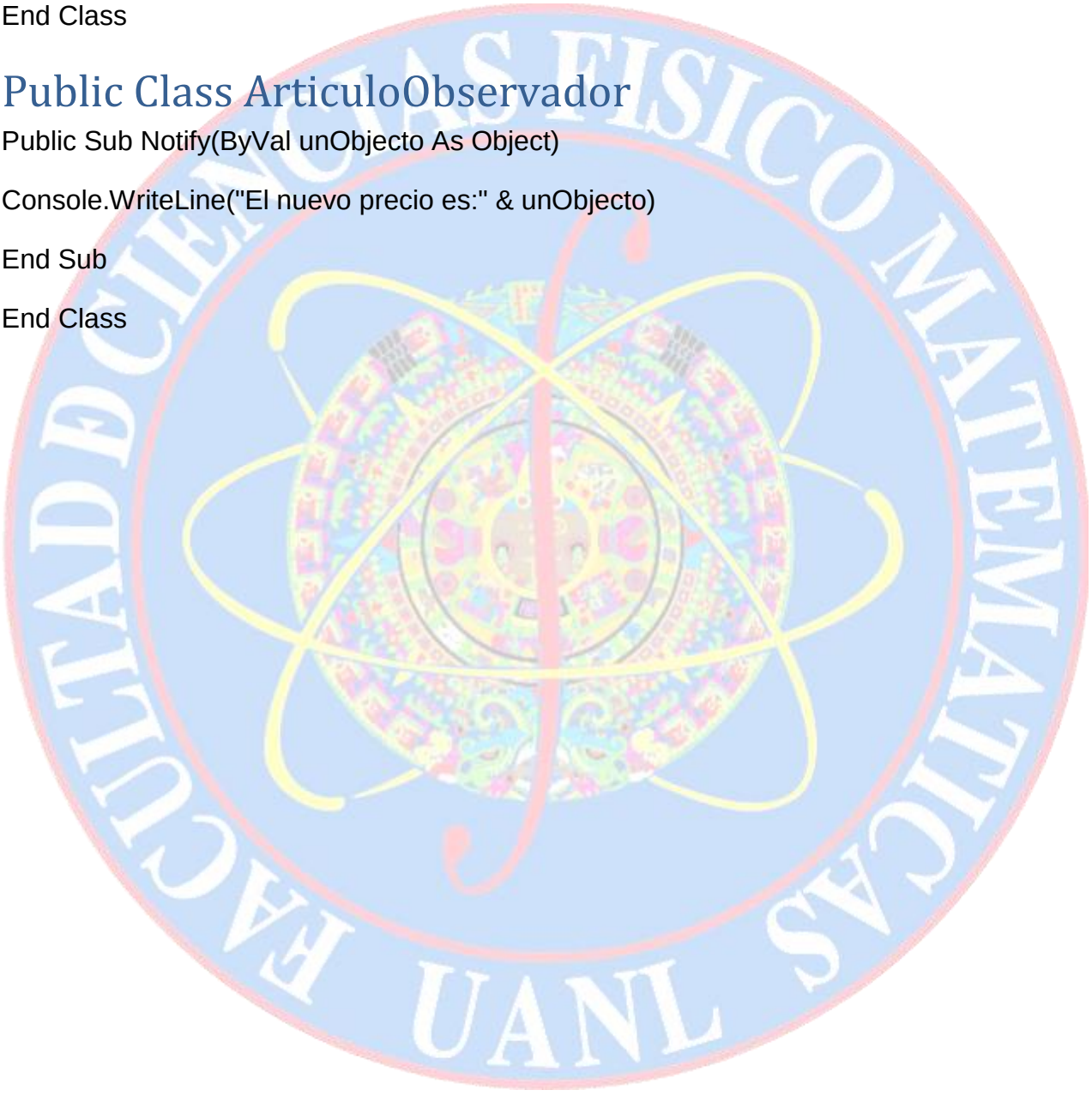
Public Class ArticuloObservador

```
Public Sub Notify(ByVal unObjecto As Object)

Console.WriteLine("El nuevo precio es:" & unObjecto)

End Sub

End Class
```



Al documentar un patrón de diseño se describe el contexto en el que se utiliza el patrón,

las fuerzas dentro del contexto en el que el patrón busca resolver, y la solución sugerida.

Además, no existe un formato único para la documentación. A continuación, se

muestran algunos formatos en la documentación de patrones:

Formato de Christopher Alexander (1979)
• Imagen mostrando un ejemplo arquitectónico del patrón.
• Párrafo introductorio del contexto del patrón (explicando la extensibilidad).
• Una línea que resuma la esencia del problema.
• El cuerpo del problema –documentación empírica, validez, manifestación, síntomas.
• La solución Representación diagramático de la solución.
• Mención de los patrones relacionados.

Formato E-LEN Project (E-LEN, 2007).
• Nombre.
• Categoría: puede ser pedagógico organizacional técnico y combinaciones.
• Abstract.
• Problema.
• Análisis.
• Soluciones conocidas.
• Preguntas de investigación: descripción de cuestiones por resolver.
• Contexto.
• Condiciones.
• Discusión/consecuencias.
• Referencias.
• Patrones relacionados.
• Autores.
• Fecha.
• Créditos

Formato de patrones pedagógicos Bergin et al (2001).
• Nombre.
• Imagen reducida o representativa.
• Contexto (audiencia) - Fuerzas (componentes del problema).

- Solución.
- Discusión/consecuencias/implementación.
- Recursos especiales.
- Patrones relacionados.
- Ejemplos/instancias.
- Contraindicaciones.
- Referencias.

Conclusión

Los patrones de diseño dan una solución a problemas repetitivos en los sistemas, de forma que se puede usar esa solución siempre que haga falta. Son guías de diseño, no reglas rigurosas. La complejidad de utilizar los patrones de diseño radica en saber cuándo utilizarlas, algo que nos dará la práctica.

Bibliografia

http://www.vbdotnetheaven.com/Code/Jun2003/2014.asp

http://blogs.vbcity.com/jspano/articles/198.aspx