Asignación 6

Diseño Orientado a Objetos

Profesor Miguel Salazar

Alumna: Luisana De León Ramírez /1744174

Patrones de diseño

Introducción

Este ensayo consiste en los patrones de diseño en general en el mundo de la programación, estos describiendo sus ejemplos, creando algunos de ellos y sobre todo definiendo que son y en que consisten los patrones de diseño, que facilitan la vida de los programadores y sobre todo realizar su trabajo sin tantos inconvenientes, ya que los patrones de diseño ya están definidos y fueron creados con ese fin de hacer el trabajo menos complicado.

Patrones de diseño

El concepto de “patrón” se refiere a los hechos o las cosas recurrentes. Estos factores o elementos se repiten con previsibilidad y, por lo tanto, pueden funcionar como modelo para producir determinada cosa a partir de ellos.

En una definición muy simple *un patrón de diseño es una forma reutilizable de resolver un problema común.* Buscar siempre una nueva solución a los mismos problemas reduce la eficacia como desarrollador, porque estás perdiendo mucho tiempo en el proceso. No hay que olvidar que el desarrollo de software también es una ingeniería, y que por tanto en muchas ocasiones habrá reglas comunes para solucionar problemas comunes.

Los patrones de diseño atajan ese punto. Te permitirán solucionar la mayor parte de tus problemas de forma directa, sin tener que pensar en cómo de válidas son, o si puede haber una alternativa mejor.

¿Para qué sirven los patrones de diseño?

Te ayudan a estar seguro de la validez de tu código

Los patrones de diseño son estructuras probadas por millones de desarrolladores a lo largo de muchos años, por lo que, si eliges el patrón adecuado para modelar el problema adecuado, puedes estar seguro de que va a ser una de las soluciones más válidas (si no la que más) que puedas encontrar.

Establecen un lenguaje común

Los patrones de diseño establecen un lenguaje común entre todos los miembros de un equipo.

¿Cómo se documenta un patrón de diseño?

El uso de un patrón de diseño en un diseño concreto se documenta, fundamentalmente, señalando la relación que hay entre los elementos estructurales del patrón y los elementos estructurales del diseño. De esta forma se intenta capturar la semántica estructural de cada elemento del diseño concreto al ligarlo a un elemento del patrón de diseño pues en este se explica la semántica de cada termino estructural. Estas relaciones se establecen por medio de asignaciones de la forma elemento patrón: = elemento diseño, donde elemento puede ser un módulo, subrutina o variable de estado. La documentación está completa si el conjunto de asignaciones incluye todos los elementos estructurales del patrón. Puede incluirse un comentario informal que explique la relación entre los elementos del patrón y los del diseño concreto.

Además, es muy conveniente acompañar la documentación con una justificación en términos del análisis de cambio realizado. En otras palabras, se debe justificar el uso del patrón por la semántica estructural de un elemento del

diseño entendemos la función que este elemento cumple en la estructura del sistema, y no su semántica funcional. Por ejemplo, estructuralmente, la subrutina Ejecutar() de la

interfaz del módulo Orden (en el patrón homónimo) es el medio que tiene el cliente para ejecutar cierta porción

de código, independientemente de la función que lleve a cabo ese código (poner en negrita, justificar, consultar

un sensor, etc.).

Pattern	**Nombre que se le da al patrón en este diseño concreto**
based on	Nombre de uno de los patrones de [GHJV03]
because	Fundamentación de la elección del patrón en términos de — Los cambios que este admite y los cambios probables anticipados en el diseño concreto — Las necesidades funcionales de alguna parte del sistema — Las restricciones de diseño que se deseen imponer
where	elemento_patrón **is** elemento_diseño elemento_patrón **is** elemento_diseño elemento_patrón **is** elemento_diseño . . . . . **is** . . . . .
comments	Explicación coloquial de la relación entre los elementos del patrón y los elementos del diseño concreto; otros comentarios adicionales que ayuden a entender cómo se aplica el patrón de diseño.

3 ejemplos de patrones de diseño

Patrones Creacionales: Inicialización y configuración de objetos.

EJEMPLO: Método de Fabricación (Factory Method)

Parte del principio de que las subclases determinan la clase a implementar.

```
public class ConcreteCreator extends Creator
 {
 protected Product FactoryMethod()
    {
        return new ConcreteProduct();
    }
}
public interface Product{}
public class ConcreteProduct implements Product{}
    public class Client
    {
        public static void main(String args[])
        {
            Creator UnCreator;
            UnCreator = new ConcreteCreator();
            UnCreator.AnOperations();
        }
    }
```

Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.

EJEMPLO: Singleton

Restringe la instanciación de una clase o valor de un tipo a un solo objeto.

```
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;
        }
    }
}
```

Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

EJEMPLO: MVC (Model View Controler)

Este patrón plantea la separación del problema en tres capas: la capa model, que representa la realidad; la capa controler , que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa vista, que muestra un aspecto del modelo y es utilizada por la capa anterior para interaccionar con el usuario.

EJEMPLO Observador (Observer): Notificaciones de cambios de estado de un objeto.

```
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
```

Conclusión

Los patrones de diseño son herramientas que contribuyen y facilitan al momento de programar, hace que perdamos menos tiempo y este mismo sea más eficaz, el saber implementarlos y tener gran conocimiento de ellos hace que el trabajo de un programador sea más sencillo y así la empresa para la que el programador trabaja tenga una buena programación en sus equipos informáticos. Porque el entender estos patrones tal vez no tenga mucha ciencia, pero el saber implementarlos si lleva su tiempo y es bueno que existan.

Bibliografía

http://www.fceia.unr.edu.ar/ingsoft/patrones-doc.pdf

https://devexperto.com/patrones-de-diseno-software/

https://definicion.de/patron/

https://msdn.microsoft.com/es-es/library/bb972240.aspx

http://www.um.es/ead/red/M10/caceres.pdf