UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

Diseño Orientado a Objetos

Patrones de diseño

Nombre: Luis Guillermo Hernández Araujo

Matricula: 1682364

Fecha: 27/Octubre/2017

¿Qué son los patrones de diseño?

Los parones de diseño son una técnica para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacciono o interfaces.

Un patrón de diseño resulta ser una solución a un problema de diseño.

Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Para qué sirven los patrones de diseño

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistema software
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente
- Formalizar un vocabulario común entre diseñadores
- Estandarizar el modo en que se realiza el diseño
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensado conocimiento ya existente

Para que NO sirven los patrones de diseño

Los patrones de diseño NO pretenden:

- Imponer ciertas alternativas de diseño frente a otras
- Eliminar la creatividad inherente al proceso de diseño

Como se documenta un patrón de diseño

- **Nombre e intención del patrón**
  - Referencia al patrón
  - Incrementa el vocabulario de diseño
- **Problema y contexto**
  - Cuando aplicar el patrón
- **Solución**
  - Estructura: elementos que conforman el diseño, sus relaciones, responsabilidades y colaboraciones
    - Es una descripción abstracta de como una disposición de elementos (clase y objetos) solucionan el problema
  - Se ilustra con un ejemplo de código
- **Consecuencias (positivas y negativas)**
  - Necesidades (tiempo, memoria), aspectos de implementación y lenguaje de programación, flexibilidad, extensibilidad, portabilidad
- **Patrones relacionados**

Categorías de patrones de diseño

- **Patrones de creación**
  - Tratan de la inicialización y configuración de clases y objetos
- **Patrones estructurales**
  - Tratan de desacoplar interfaz e implementación de clases y objetos
    - ¿Cómo se componen clases y objetos?
- **Patrones de comportamiento**
  - Tratan de las interacciones dinámicas entre sociedades de clases y objetos
    - ¿Cómo interaccionan y se distribuyen responsabilidades los objetos?

Ejemplos de patrones creacionales

Abstract Factory

El problema a solucionar por este patrón es el de crear diferentes familias de objetos, como por ejemplo la creación de interfaces graficas de distintos tipos (ventana, menú, botón, etc.)

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();
            }
      }
```

Prototype

Se basa en la clonación de ejemplares copiándolos de un prototipo

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;
                  }
            }
       }
```

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

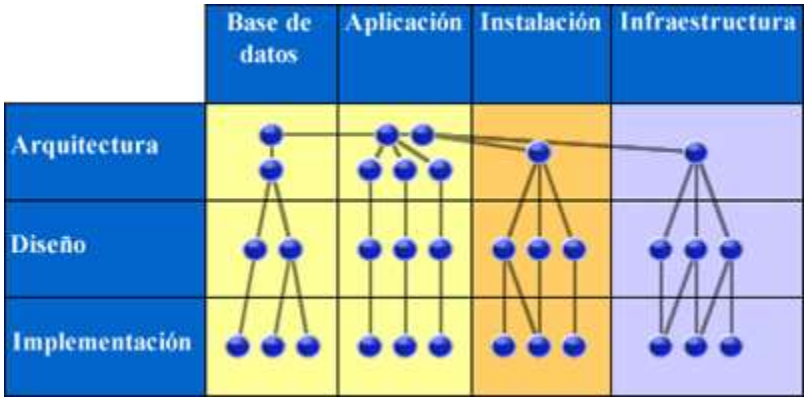
Ejemplos de patrones Estructurales

- Adaptador (Adapter): Convierte una interfaz en otra.
- Puente (Bridge): Desacopla una abstracción de su implementación permitiendo modificarlas independientemente.
- Objeto Compuesto (Composite): Utilizado para construir objetos complejos a partir de otros más simples, utilizando para ello la composición recursiva y una estructura de árbol.
- Envoltorio (Decorator): Permite añadir dinámicamente funcionalidad a una clase existente, evitando heredar sucesivas clases para incorporar la nueva funcionalidad.
- Fachada (Facade): Permite simplificar la interfaz para un subsistema.
- Peso Ligero (Flyweight): Elimina la redundancia o la reduce cuando tenemos gran cantidad de objetos con información idéntica.
- Apoderado (Proxy): Un objeto se aproxima a otro.

Ejemplos de patrones de comportamiento

- Cadena de responsabilidad (Chain of responsibility): La base es permitir que más de un objeto tenga la posibilidad de atender una petición.
- Orden (Command): Encapsula una petición como un objeto dando la posibilidad de "deshacer" la petición.
- Intérprete (Interpreter): Intérprete de lenguaje para una gramática simple y sencilla.
- Iterador (Iterator): Define una interfaz que declara los métodos necesarios para acceder secuencialmente a una colección de objetos sin exponer su estructura interna.
- Mediador (Mediator): Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.
- Recuerdo (Memento): Almacena el estado de un objeto y lo restaura posteriormente.
- Observador (Observer): Notificaciones de cambios de estado de un objeto.
- Estado (Server): Se utiliza cuando el comportamiento de un objeto cambia dependiendo del estado del mismo.
- Estrategia (Strategy): Utilizado para manejar la selección de un algoritmo.
- Método plantilla (Template Method): Algoritmo con varios pasos suministrados por una clase derivada.
- Visitante (Visitor): Operaciones aplicadas a elementos de una estructura de objetos heterogénea.

Conclusión



En el diagrama de la figura, Microsoft sostiene que las comunas de la tabla son enfoques o puntos de vista de la solución, mientras que las filas representan los niveles de abstracción.

Bibliografía

https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o

https://www.fdi.ucm.es/profesor/jpavon/poo/2.14PDOO.pdf

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