Universidad Autónoma de Nuevo León

Que son los Patrones de Diseño

Autor:

Brandon Uriel Quiñones Baldazo

Maestro:

Miguel Salazar

¿Que son los patrones de diseño?

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.”
En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). Grande fue mi sorpresa al averiguar que existen varios patrones de diseño popularmente conocidos, los cuales se clasifican como se muestra a continuación:

- Patrones Creacionales: Inicialización y configuración de objetos.
- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

Tipos de Patrones de diseño

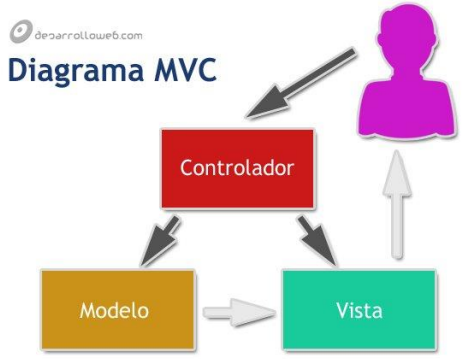
Método de Fabricación (*Factory Method*)
Parte del principio de que las subclases determinan la clase a implementar.

```
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();
            }
      }
```

Prototipado (*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;
                  }
            }
       }
```

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.



Conclusión

El patrón de diseño es algo que se presenta a menudo y que se puede solucionar con un mismo programa ya que va a ser lo mismo, no hay necesidad de crear más objetos si se va a hacer la misma autentificación.