Contents

[PATRONES DE DISEÑO 2](#_Toc32243075)

[27/01/2020 2](#_Toc32243076)

[TEMARIO 2](#_Toc32243077)

[PORCENTAJE DE EVALUACIÓN 3](#_Toc32243078)

[28/01/2020 3](#_Toc32243079)

[29/01/2020 3](#_Toc32243080)

[Patrón de diseño 3](#_Toc32243081)

[Arquitectura de los patrones de diseño 3](#_Toc32243082)

[Por qué estudiar patrones de diseño? 4](#_Toc32243083)

[Qué nos resuelve un patrón de diseño? 4](#_Toc32243084)

[Características de los patrones de diseño 4](#_Toc32243085)

[Exposición 4](#_Toc32243086)

[04/02/2020 4](#_Toc32243087)

[05/02/2020 4](#_Toc32243088)

[Problemas de los patrones. 5](#_Toc32243089)

[Cómo seleccionar un patron de diseño 5](#_Toc32243090)

[Cómo usar un patrón de diseño 5](#_Toc32243091)

[Ventajas de los patrones de diseño 5](#_Toc32243092)

[Desventajas 5](#_Toc32243093)

# PATRONES DE DISEÑO

## 27/01/2020

Classroom: jsykkdh

LAB A: MARTES Y JUEVES

### TEMARIO

UNIDAD 1: INTRODUCCIÓN

1. CONCEPTO DE PATRONES DE DISEÑO
2. DESCRIPCIÓN DE PATRONES DE DISEÑO
3. CATEGORÍA DE PATRONES DE DISEÑO
4. CÓMO RESOLVER PROBLEMAS CON PATRONES DE DISEÑO
5. CÓMO SELECCIONAR UN PATRÓN DE DISEÑO
6. CÓMO UTILIZAR UN PATRÓN DE DISEÑO

UNIDAD 2: PATRONES CREACIONALES

1. FÁBRICA ABSTRACTA
2. CONSTRUCTOR
3. MÉTODO FÁBRICA
4. PROTOTIPO
5. SINGLETON
6. PISCINA DE OBJETOS

UNIDAD 3: PATRONES ESTRUCTURALES

1. ADAPTADOR
2. PUENTE
3. COMPUESTO
4. DECORADOR
5. FACHADA
6. PESO LIGERO
7. DATOS DE CLASE PRIVADA
8. PROXY

UNIDAD 4: PATRONES DE COMPORTAMIENTO

1. CADENA DE RESPONSABILIDAD
2. COMANDO
3. INTÉRPRETE
4. ITERADOR
5. MEDIADOR
6. RECORDATORIO
7. OBSERVADOR
8. ESTADO
9. ESTRATEGIA
10. MÉTODO PLANTILLA
11. OBJETO NULO
12. VISITANTE

UNIDAD 5: PATRONES DE ARQUITECTURA

1. DEL BARRO AL ESTRUTURA
   1. CAPAS
   2. TUBOS Y FILTROS
   3. PIZARRA
2. ARQUITECTURA ORIENTADA A EVENTOS
   1. TOPOLOGÍA MEDIADOR
   2. TOPOLOGÍA BROKER
3. ARQUITECTURA MICRO-KERNEL
4. ARQUITECTURA MICRO-SERVICIOS
5. ARQUITECTURA BASADA EN ESPACIOS
6. ARQUITECTURA DE SERVICIOS INTERACTIVOS
   1. INTRODUCCIÓN A ARQUITECTURAS MV
   2. MVC
   3. EVOLUCIONES DE MVC (MVP, MVVM, MVT)

### PORCENTAJE DE EVALUACIÓN

70% EXAMEN

30% PRÁCTICA

## 28/01/2020

MAPA CONCEPTUAL: CATEGORÍAS DE PATRONES DE DISEÑO

## 29/01/2020

### Patrón de diseño

Es una descripción de clases y objetos relacionados que están adaptados para resolver un problema de

diseño general en un contexto determinado.

![Descripción de un patrón de diseño.](/images/1.png "Descripción de un patrón de diseño")

Los patrones de diseño se defines con un alto nivel de abstración. Son independientes de los lenguajes de programación y de los detalles de implementación.

Los patrones de diseño promueven y facilitan la reutilización de arquitecturas y diseño de software que han demostrado su válidez en muchas aplicaciones.

### Arquitectura de los patrones de diseño

1. Definen la forma y estructura de la aplicación de software.
2. Está relacionada al propósito de la aplicación.
3. Arquitectura de modulos y sus interconexiones.

### Por qué estudiar patrones de diseño?

* Reúso de soluciones de diseño.
* Establecer terminología común.
* Dan una perspectiva de alto nivel en el análisis y diseño.

### Qué nos resuelve un patrón de diseño?

Proporciona un esquema para refinar los subsitemas o componentes de un sistema de software o las relaciones entre ellos. Describe estructuras repetitivas de comunicación que resuelven un problema de diseño en un componente particular.

### Características de los patrones de diseño

* Son soluciones concretas.
* Son soluciones técnicas.
* Se aplican en situaciones muy comunes.
* Son soluciones simples.
* Facilitan la reutilización de las clases y del propio diseño.

### Exposición

* Portada
* Introducción
* Contenido
* Desarrollo de tema
  + Definiciones
  + Ventajas y desventajas
  + Características
  + Metodología
  + Componentes
  + Aplicaciones
  + Ejemplos
  + Practicas
* Conclusiones
* Referencias

Equipo 1. Tema: Fábrica abstracta.

## 04/02/2020

## 05/02/2020

* Capturan el conocimiento de los expertos y lo hacen accessible a toda la comunidad del software.
* Los patrones favorecen la reutilización de diseño de arquitecturas a gran escala.
* Proporcionan un cuerpo de conocimiento utilizable por toda la comunidad de software
* Favorecen la transmisión de conomiento entre los profesionales y entre clientes y desarrolladores.
* Proporcionan un lenguaje común. Los patrones forman parte del vocabulario técnico del ingeniero de software.

### Problemas de los patrones.

* Los patrones no llevan de forma directa a la reutilización del código, aunque esta facilita su uso.
* La integración de los patrones en el proceso de desarrollo se hace todavía de forma manual.
* El número de patrones identificados es cada vez más grande. Las clasificaciones actuales no siempre sirven de guía para decidir cuál usar.
* El número de combinaciones patrones, estilos y atributos se dan en la práctica.
* Los patrones se validan por la experiencia y el debate, no mediante la aplicación de técnicas formales.

### Cómo seleccionar un patron de diseño

* Considerar cómo los patrones de diseño solucionan los problemas de diseño.
* Buscar las intenciones de cada patrón.
* Estudiar cómo se interelacionan los patrones.
* Evaluar el propósito de los patrones.
* Examinar la causa de un rediseño.
* Considerar qué debería hacer una variable en un diseño.

### Cómo usar un patrón de diseño

1. Leer el patrón para tener una vision general
2. Volver y estudiar la estructura los participantes y las colaboraciones.
3. Ver un ejemplo códificado del patrón.
4. Elegir nombres para los participantes del patrón que sean significativos en el context de la aplicación.
5. Definir las clases.
6. Definir los nombres específicos para las operaciones en el patrón.
7. Implementar las operaciones que realizan las responsabilidades y colaboración del patrón.

### Ventajas de los patrones de diseño

* Se definen estáticamente
* Fácil modificación de la implementación

### Desventajas

* No se cambian implementaciones en tiempo de ejecución.
* Rompe el encapsulamiento.

## 10/02/2020

### Descripción de un patrón

* Nombre: Corto y descriptivo
* Problema: Descripción del problema que refuerza la intención del patrón
* Contexto: Precondiciones necesarias para el problema y la solución que exista
* Fuerza: Son las variables principales del problema, que determinan que tan buena o mala es la solución.
* Solución: Relaciones estáticas y reglas de funcionamiento que describen cómo llegar al resultado.
* Contexto resultante: es el estado o configuración del Sistema despues de aplicar el patrón.
* Racionalidad: Es una explicación justificada de los pasos o reglas en el patrón y del patrón como un todo y acorde con sus metas, principios y filosofía.
* Relaciones: Las relaciones estáticas y dinámicas de éste patrón con otro.

### Examen 19 de feb.

## 12/02/2020

### EXPO: Builder

Patron de diseño creacional.

* DIRECTOR: Se encarga de producir el objeto.
* BUILDER:
* CONCRETEBUILDER: Proceso de constructor.
* PRODUCT: Lo que generamos, los objetos.

El código es mantenible si el numeroi de parametros para crear el objeto es mayor que cuatro.

Práctica: Crear un programa con el patrón de diseño mencionado, para la construcción de …

## 13/02/2020

## GUÍA EXAMEN

1. Definición de patrones de diseño

Es una descripción de clases y objetos relacionados que están adaptados para resolver un problema de diseño general en un contexto determinado.

1. Menciona y explica por qué estudiar patrones de diseño

* Reúso de soluciones de diseño.
* Establecer terminología común.
* Dan una perspectiva de alto nivel en el análisis y diseño.

1. Dar las características de los patrones de diseño

* Son soluciones concretas.
* Son soluciones técnicas.
* Se aplican en situaciones muy comunes.
* Son soluciones simples.
* Facilitan la reutilización de las clases y del propio diseño.

1. Menciona lo beneficios de los patrones de diseño

* Capturan el conocimiento de los expertos y lo hacen accessible a toda la comunidad del software.
* Los patrones favorecen la reutilización de diseño de arquitecturas a gran escala.
* Proporcionan un cuerpo de conocimiento utilizable por toda la comunidad de software
* Favorecen la transmisión de conomiento entre los profesionales y entre clientes y desarrolladores.
* Proporcionan un lenguaje común. Los patrones forman parte del vocabulario técnico del ingeniero de software.

1. Cómo seleccionamos un patrón de diseño

* Considerar cómo los patrones de diseño solucionan los problemas de diseño.
* Buscar las intenciones de cada patrón.
* Estudiar cómo se interrelacionan los patrones.
* Evaluar el propósito de los patrones.
* Examinar la causa de un rediseño.
* Considerar qué debería hacer una variable en un diseño.

1. Ventajas de los patrones de diseño

* Se definen estáticamente
* Fácil modificación de la implementación

1. Cómo describimos un patrón de diseño

* Nombre: Corto y descriptivo
* Problema: Descripción del problema que refuerza la intención del patrón
* Contexto: Precondiciones necesarias para el problema y la solución que exista
* Fuerza: Son las variables principales del problema, que determinan que tan buena o mala es la solución.
* Solución: Relaciones estáticas y reglas de funcionamiento que describen cómo llegar al resultado.
* Contexto resultante: es el estado o configuración del Sistema despues de aplicar el patrón.
* Racionalidad: Es una explicación justificada de los pasos o reglas en el patrón y del patrón como un todo y acorde con sus metas, principios y filosofía.
* Relaciones: Las relaciones estáticas y dinámicas de éste patrón con otro.

1. Dar la definición de solución

Relaciones estáticas y reglas de funcionamiento que describen cómo llegar al resultado.

1. Definición de contexto

Precondiciones necesarias para el problema y la solución que exista

1. Cómo se usan los patrones de diseño

* Leer el patrón para tener una vision general
* Volver y estudiar la estructura los participantes y las colaboraciones.
* Ver un ejemplo códificado del patrón.
* Elegir nombres para los participantes del patrón que sean significativos en el context de la aplicación.
* Definir las clases.
* Definir los nombres específicos para las operaciones en el patrón.
* Implementar las operaciones que realizan las responsabilidades y colaboración del patrón.

1. Dar la definición de abstract factory

Abstract Factory es un patrón de diseño creacional para el desarrollo de software. Provee una interfaz para crear familias de objetos relacionados o dependientes entre ellos sin especificar una clase en concreto. Los patrones de Abstract Factory funcionan en torno a una súper fábrica que crea otras fábricas.

1. Características de abstract factory

Aísla clases concretas. El patrón Abstract Factory ayuda a controlar las clases de objetos que crea una aplicación.

Facilita el intercambio de familias de productos. La clase de una fábrica concreta aparece solo una vez en una aplicación, es decir, donde se instancia. Esto facilita cambiar la fábrica concreta que utiliza una aplicación.

Promueve la consistencia entre los productos. Cuando los objetos de productos en una familia están diseñados para trabajar juntos, es importante que una aplicación use objetos de una sola familia a la vez.

1. Menciona y explica la estructura del patrón abstract factory

Cliente (Client): La clase que llamará a la fábrica adecuada ya que necesita crear uno de los objetos que provee la fábrica, es decir, Cliente lo que quiere es obtener una instancia de alguno de los productos.

Fábricas Abstractas (Abstract Factory): Es la definición de las interfaces de las fábricas. Debe de proveer un método para la obtención de cada objeto o producto abstracto que pueda crear.

Fábricas Concretas (Concrete Factory): Estas son las diferentes familias de productos. Provee de la instancia concreta de la que se encarga de crear.

Producto abstracto (Abstract Product): Definición de las interfaces para la familia de productos genéricos. El cliente trabajará directamente sobre esta interfaz, que será implementada por los diferentes productos concretos.

Producto concreto (Concrete Product): Implementación de los diferentes productos abstractos.

1. Escribe un ejemplo de este patrón

Un ejemplo de esto sería una clase de fábrica abstracta DocumentCreator que proporciona interfaces para crear una serie de productos (por ejemplo, createLetter() y createResume()). El sistema tendría cualquier número de versiones concretas derivadas de la claseDocumentCreator como FancyDocumentCreator o ModernDocumentCreator, cada una con una implementación diferente de createLetter() y createResume() que crearía un objeto correspondiente como FancyLetter o ModernResume. Cada uno de estos productos se deriva de una clase abstracta simple como Carta o Currículum vitae que el cliente conoce. El código del cliente obtendría una instancia apropiada de DocumentCreator y llamaría a sus métodos de fábrica. Cada uno de los objetos resultantes se crearía a partir de la misma implementación de DocumentCreator y compartiría un tema común (todos serían objetos elegantes o modernos). El cliente solo necesitaría saber cómo manejar la clase abstracta de Letter o Resume, no la versión específica que obtuvo de la fábrica de concreto.

1. Qué es el patrón builder

El patrón de diseño Builder separa la creación de un objeto complejo de su representación de modo que el mismo proceso de construcción pueda crear representaciones diferentes.

1. Menciona y explica la estructura del patrón builder

Director: Se encarga de construir un objeto utilizando el Constructor (Builder).

Builder: Interfaz abstracta que permite la creación de objetos.

Concrete Builder: Implementación concreta del Builder definida para cada uno de los tipos. Permite crear el objeto concreto recopilando y creando cada una de las partes que lo compone.

Product: Objeto que se ha construido tras el proceso definido por el patrón.

1. Menciona aplicaciones del patrón builder

Cuando alguna de las características puede ser separada de las partes que componen el objeto.

Cuando el proceso de construcción debe permitir distintas representaciones para el objeto que se construye.

Cuando el objeto a construir es complejo y sus distintas configuraciones son limitadas.

## 18/02/02

### Método de fábrica

Trata la creación de objetos sin especificar explícitamente la clase real que tendrá la instancia.

IProduct: Producto abstract

Concrete Product: Producto concreto

Concrete Factory: Fábrica concreta

Abstract Factory: Fábrica abstracta

### Patrón prototype

CLONAR OBJETOS

IPrototype

ConcretePrototype extends IPrototype

PrototypeFactory clone IPrototype

Client use Prototype and IPrototype

Clonación simple y clonación complete

## 24/02/2020

### Piscina de objetos

Cliente: Quien usa lo objetos.

ReusablePool: Guarda los objetos

ObjectPool: Lista de objetos disponibles y los que ya se han utilizado.

## 27/02/2020

### Adaptador

Target

Adaptee

Client

Adapter

### Puente

Bridge

Conceptos clave

Abstracción

Desacoplar

Componentes

Cliente, abstracción e implementación

Abstracción, abstración refinada

Implementador, implementador concreto

## 04/02/2020

### Guía examen

Examen día martes 10 de marzo

1. Definición de método fábrica
2. Menciona las ventajas y desventajas de factory method
3. Definición de patrón prototype
4. ¿Cuándo debemos aplicar el patrón prototype?
5. Menciona y explica los componentes del patrón singleton
6. Da un ejemplo en código del patrón singleton
7. Da la definición del patrón object pool
8. ¿Cuándo se debe usar el patrón object pool?
9. Menciona y explica la estructura del patrón object pool
10. Menciona las ventajas y desventajas del patrón adapter
11. Menciona algunas aplicaciones del patrón adapter
12. Definición del patrón bridge
13. Menciona los participantes en el patrón bridge
14. Da un ejemplo del patrón bridge
15. Definición patrón composite
16. Menciona y explica los elementos del patrón composite
17. ¿Cuál es la metodología para aplicar el patrón composite?
18. Definición del patrón decorator
19. Ejemplo patrón decorator
20. Menciona la estructura del patrón decorator

### Patrón Facade

Componentes

Fachada. Conoce los responsables de cada petición

Subclases. Contienen los procesos para cada petición. Las subclases no conocen que existe una fachada.

### Patrón FlyWeight

Client

Flyweight Factory

Flyweght

## 11/02/2020

### Datos de clase privada

Encapsular los atributos de una clase en una clase privada única.

Busca reducir la exposición de los atributos de una clase.