Patrones Software
Tema 1-3:
Conceptos generales

Patrones de Diseño

- El objetivo de los patrones de diseño es guardar la experiencia en diseños de programas orientados a objetos (catálogos de patrones).
- Podemos encontrar patrones de clases y comunicaciones entre objetos en muchos sistemas orientados a objetos. Estos patrones solucionan problemas específicos del diseño y hacen los diseños orientados a objetos más flexibles, elegantes y reutilizables.
- Un diseñador que conoce algunos patrones puede aplicarlos inmediatamente a problemas de diseño sin tener que descubrirlos.
- Los patrones de diseño ayudan a un diseñador a conseguir un diseño correcto rápidamente.

Elementos de un patrón

- El **nombre del patrón** se utiliza para describir un problema de diseño, su solución, y consecuencias de forma resumida. Nombrar un patrón incrementa inmediatamente nuestro vocabulario de diseño.

- El **problema** describe cuando aplicar el patrón. Se explica el problema y su contexto. Podría describir problemas de diseño específicos tales como algoritmos o como objetos. Algunas veces el problema incluirá una lista de condiciones que deben cumplirse para poder aplicar el patrón.

Elementos de un patrón

- La **solución** describe los elementos que forman el diseño, sus relaciones, responsabilidades y colaboraciones.

  La solución no describe un diseño particular o implementación, proveen una descripción abstracta de un problema de diseño y una disposición general de los elementos (clases y objetos en nuestro caso) que lo soluciona.

- Las **consecuencias** son los resultados de aplicar el patrón. Estas son muy importantes para la evaluación de diseños alternativos y para comprender los costes y beneficios de la aplicación del patrón.

Elementos de un patrón

- Los patrones de diseño identifican las clases y objetos participantes, sus papeles, colaboraciones y comunicaciones y la distribución de responsabilidades.
- Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño común, que lo hace útil para la creación de diseños orientados a objetos reutilizables.
- Los patrones de diseño se pueden utilizar en cualquier lenguaje de programación orientado a objetos, adaptando los diseños generales a las características de la implementación particular.

Descripción de un patrón

- **Nombre:**
  - Un nombre descriptivo del patrón. El nombre es significativo y corto, fácil de recordar y asociar a la información que sigue. También se pueden incluir nombres alternativos, en caso de que los haya.
- **Propiedades:**
  - La clasificación del patrón. Tipo: creación, estructural, comportamiento. Nivel: Clase única, Componente (grupo de clases), Objeto.
- **Objetivo o Propósito:**
  - Breve explicación de las implicaciones del patrón.
- **Aplicabilidad:**
  - Cuando y porqué es deseable usar este patrón. En que situaciones es aplicable el patrón.

Descripción de un patrón

- **Estructura:**
  - Es el núcleo del patrón. Se describe una solución general al problema que el patrón soluciona. Esta descripción puede incluir, diagramas y texto que identifique la estructura del patrón, sus participantes y sus colaboraciones para mostrar como se soluciona el problema.
- **Consecuencias:**
  - Explica las implicaciones, buenas y malas, del uso de la solución.
- **Patrones relacionados:**
  - Otros patrones con los que está asociado o con los que tiene una relación estrecha.
- **Código de ejemplo:**
  - Ejemplo en código Java.

Cualidades de un patrón

- **Encapsulación y abstracción:**
  - Cada patrón encapsula un problema bien definido y su solución en un dominio particular y limitado.
  - Los patrones también sirven como abstracciones las cuales contienen dominios conocidos y experiencia.
- **Extensión y variabilidad:**
  - Cada patrón debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solucionar un gran problema.
  - Un patrón solución debería ser también capaz de realizar una variedad infinita de implementaciones (de forma individual, y también en conjunción con otros patrones).

Cualidades de un patrón

- **Generatividad y composición:**
  - Cada patrón, una vez aplicado, genera un contexto resultante, el cual concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo.
  - Esta subsecuencia de patrones y su composición con otros patrones podría ser aplicada progresivamente para conseguir la generación de un "todo" o solución completa.
- **Equilibrio:**
  - Cada patrón debe realizar algún tipo de balance entre sus efectos y restricciones.

Clasificación de los patrones

- **Patrones de creación:**
  - Los patrones de creación muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Éstas se suelen resolver dinámicamente decidiendo que clases instanciar o sobre que objetos un objeto delegará responsabilidades.
  - A menudo hay varios patrones de creación que se pueden aplicar en una misma situación. Otras veces se pueden combinar varios de ellos. En otros casos se debe elegir solo uno de ellos.
- **Patrones estructurales:**
  - Los patrones de esta categoría describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- **Patrones de comportamiento:**
  - Los patrones de este tipo son utilizados para organizar, manejar y combinar comportamientos. Nos permiten definir la comunicación entre los objetos de nuestro sistema y el flujo de la información entre los mismos.

Clasificación de los patrones

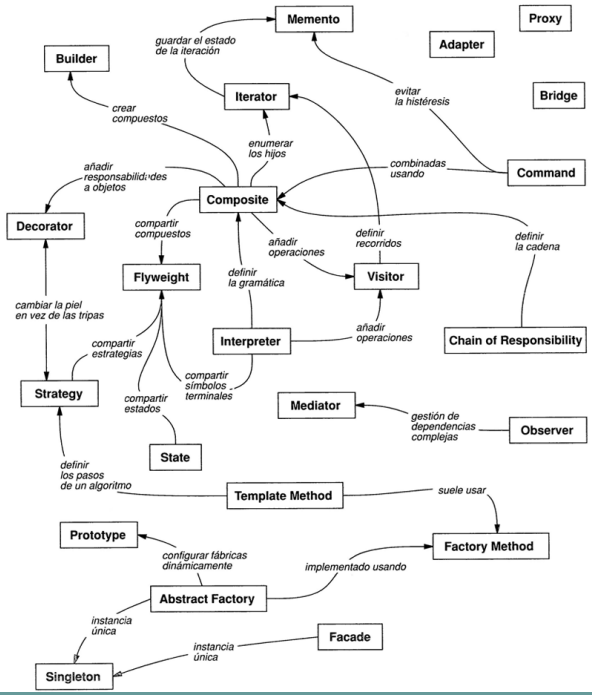
CREACIÓN	ESTRUCTURALES	COMPORTAMIENTO
Abstract Factory (O)	Adapter (C)	Chain of responsibility (O)
Builder (O)	Bridge (O)	Command (O)
Factory Method (C)	Composite (O)	Interpreter (C)
Prototype (O)	Decorator (O)	Iterator (O)
Singleton (O)	Facade (O)	Mediator (O)
	Flyweight (O)	Memento (O)
	Proxy (O)	Observer (O)
		State (O)
		Strategy (O)
		Template Method (C)
		Visitor (O)

Ámbito: (C) Clase, (O) Objeto.

Clase: Relaciones entre clases y subclases. Herencia. Estáticas

Objeto: Relaciones entre objetos. Dinámicas.

Relación entre los patrones



Patrones y Frameworks

- **Framework:** Conjunto de clases que cooperan y forman un diseño reutilizable para un tipo específico de software. Un framework ofrece una guía arquitectónica partiendo el diseño en clases abstractas y definiendo sus responsabilidades y sus colaboraciones. Un desarrollador personaliza el marco de trabajo para una aplicación particular mediante herencia y composición de instancias de las clases del framework.
- Un framework no es generalmente una aplicación completa sino que a menudo le falta la funcionalidad de una aplicación específica.
- Una **aplicación**, en cambio, puede ser construida con uno o más frameworks insertando dicha funcionalidad.
- Un framework proporciona una guía arquitectónica para dividir el diseño en clases y definir sus responsabilidades y colaboraciones. Un framework dicta la arquitectura de la aplicación.

Patrones y Frameworks

- Los frameworks pueden incluir patrones de diseño.
- Los frameworks son ejecutables mientras que los patrones de diseño representan un conocimiento o experiencia sobre software.
- Un framework se puede ver como la implementación de un sistema de patrones de diseño.
- Los frameworks son de naturaleza física, mientras que los patrones son de naturaleza lógica: los frameworks son la realización física de uno o varios patrones de soluciones software; los patrones son las instrucciones de cómo implementar estas soluciones.

Patrones y Frameworks. Diferencias

- Los patrones de diseño son más abstractos que los frameworks:
  - *Los frameworks pueden ser codificados, pero solo los ejemplos de patrones de diseño pueden ser codificados. Los frameworks pueden ser escritos bajo lenguajes de programación y no solamente estudiados para ejecutarse mientras que los patrones de diseño tienen que ser implementados cada vez que son usados.*
- Los patrones de diseño son elementos arquitectónicos más pequeños que los Frameworks:
  - *Un framework contiene varios patrones de diseño, mientras que un patrón de diseño no contiene varios frameworks.*
- Los patrones de diseño están menos especializados que los frameworks:
  - *Los frameworks siempre tienen un dominio particular de aplicación, mientras que los patrones de diseño pueden ser utilizados en cualquier tipo de aplicación.*