ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DTT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	Miércoles, 31/07/2024
Hora de inicio:	15:40

Análisis y Diseño de Sistemas 2 [B]

Luis Angel Barrera Velásquez

AGENDA

1	Avisos
2	Grupos
3	Teoría Introductoria de Patrones de diseño
4	Repaso UML

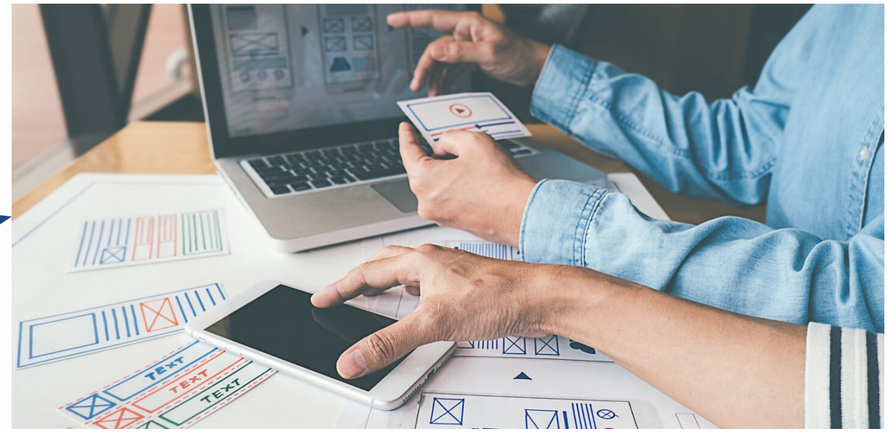
Elementos de Diseño

¿Que es un elemento de diseño en software?

Los elementos de diseño en software son componentes fundamentales y conceptos que se utilizan para definir la arquitectura, estructura y comportamiento de un sistema de software. Estos elementos ayudan a los desarrolladores a crear software que es eficiente, mantenible, escalable y fácil de entender.

Patrones de diseño de software

Los patrones de diseño son soluciones reutilizables a problemas comunes que enfrentan los diseñadores y desarrolladores de software. Estos patrones proporcionan una estructura para elaborar diseños, lo que permite la creación de soluciones eficientes y mantenibles.



Importancia y aplicabilidad de los patrones de diseño

1

Estandarización:
Proporcionan un vocabulario común para los desarrolladores, lo que facilita la comunicación y la comprensión de las soluciones.

2

Reutilización:
Permiten reutilizar soluciones probadas, reduciendo el tiempo y esfuerzo necesarios para resolver problemas recurrentes.

3

Mantenimiento:
Los patrones bien aplicados mejoran la claridad y la organización del código, lo que facilita su mantenimiento.

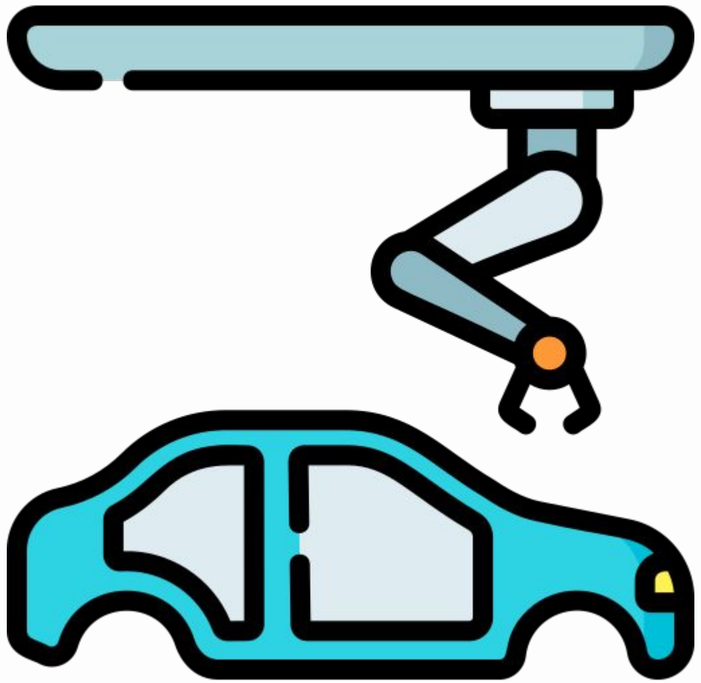
4

Flexibilidad:
Ayudan a crear sistemas más adaptables y flexibles que pueden evolucionar con cambios en los requisitos o el entorno.

Tipos de Patrones de diseño

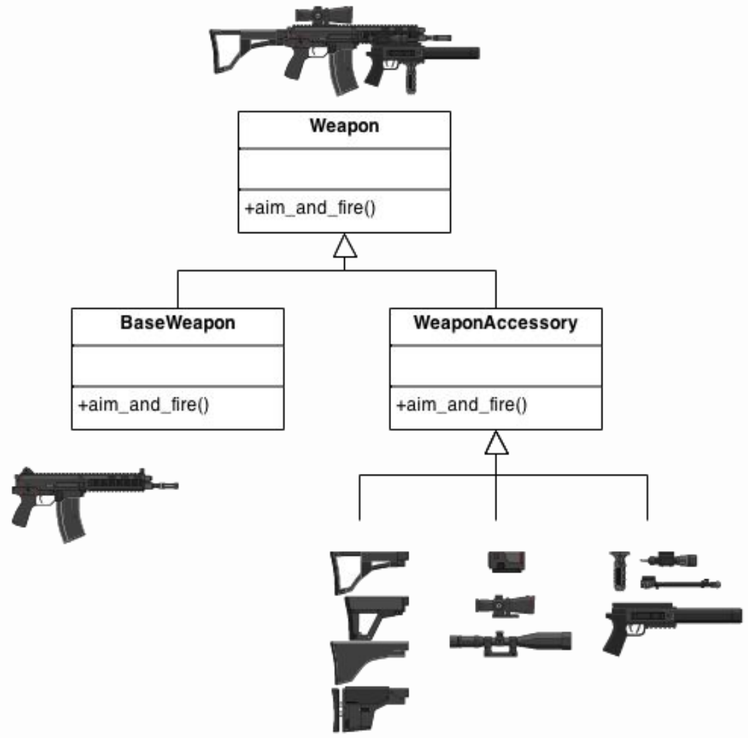
Patrones Creacionales

Son patrones de diseño relacionados con la creación o construcción de objetos. Estos patrones intentan controlar la forma en que los objetos son creados implementando mecanismos que eviten la creación directa de objetos.



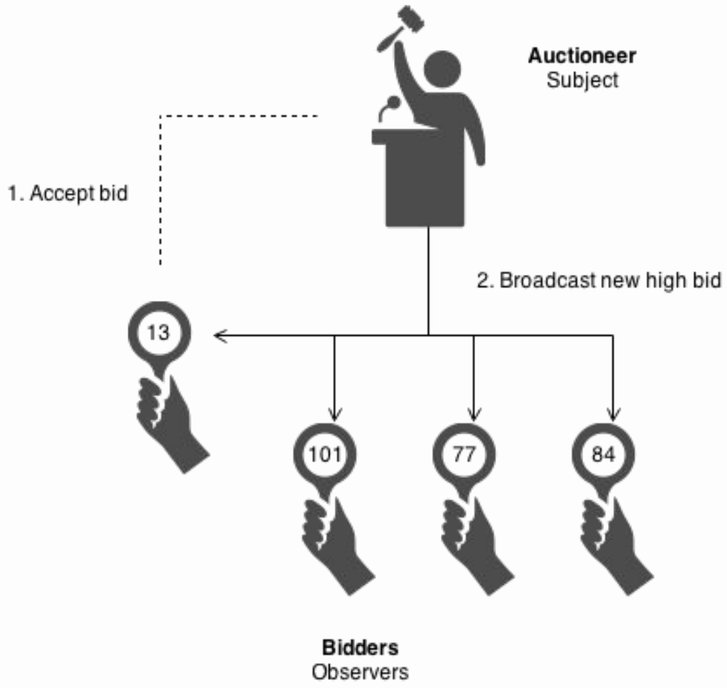
Patrones Estructurales

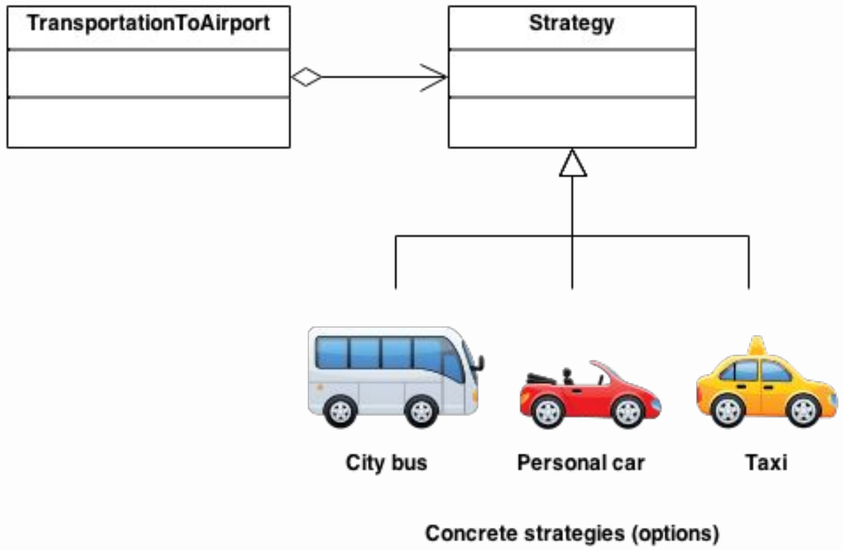
Son patrones que tiene que ver con la forma en que las clases se relacionan con otras clases. Estos patrones ayudan a dar un mayor orden a nuestras clases ayudando a crear componentes más flexibles y extensibles.



Patrones de Comportamiento

Son patrones que están relacionados con procedimientos y con la asignación de responsabilidad a los objetos. Los patrones de comportamiento engloban también patrones de comunicación entre ellos.





Principios de Diseño

Don't Repeat Yourself:
Write the same code only once.
Do not copy-paste.

You Ain't Gonna Need It:
Don't write code you don't need right now.

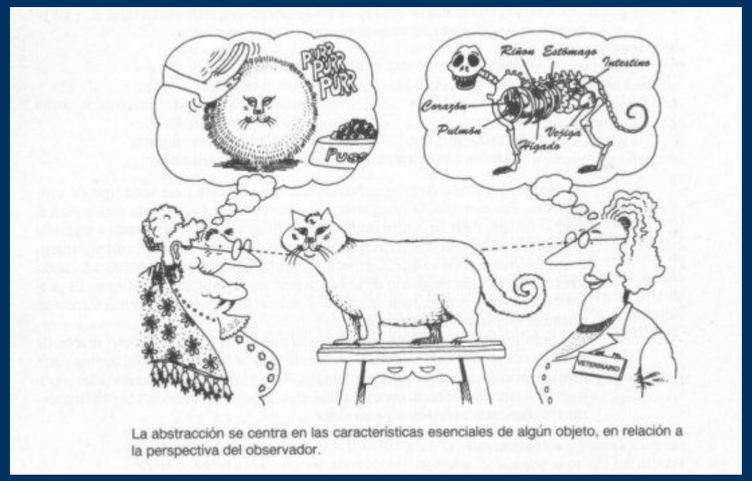
Keep It Simple Stupid:
Create the simplest solution you can think of. Refactor.

Simple Is Not Easy:
It's harder to create a simple solution than a complex one. Simplicity requires work. Still, do it.

Desafíos actuales en la implementación de patrones de diseño

1

Excesiva Abstracción



La abstracción se centra en las características esenciales de algún objeto, en relación a la perspectiva del observador.

2

Resistencia al cambio

La Resistencia al Cambio, ¿Es un Riesgo de TI?

3

Cultura y capacitación

Repaso

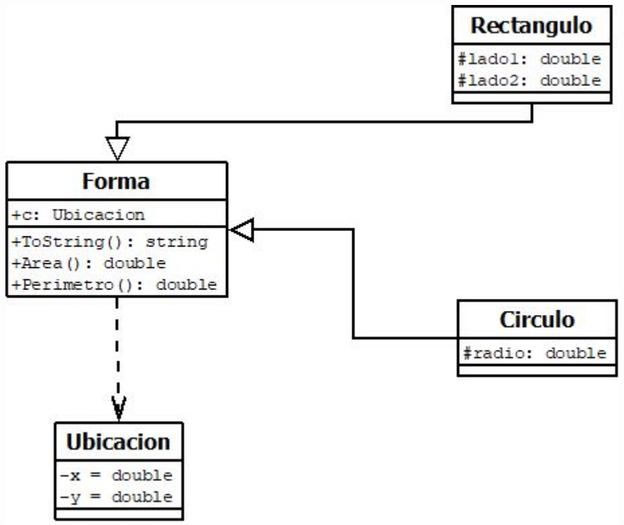
UML

UML

UML (Lenguaje Unificado de Modelado) es una herramienta estándar en la industria del software que nos permite visualizar, especificar, construir y documentar sistemas. Es como un lenguaje visual que nos ayuda a entender y comunicar de manera más efectiva la estructura y el comportamiento de un sistema, ya sea un software, un proceso o un negocio.

Diagrama de clases

El diagrama de clases es uno de los diagramas más utilizados en UML. Representa la estructura estática de un sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre ellas.

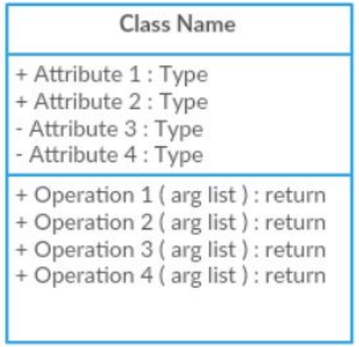


Clase

Una clase en UML representa una plantilla o blueprint para crear objetos. Es decir, define las características (atributos) y comportamientos (métodos) que compartirán todos los objetos de ese tipo.

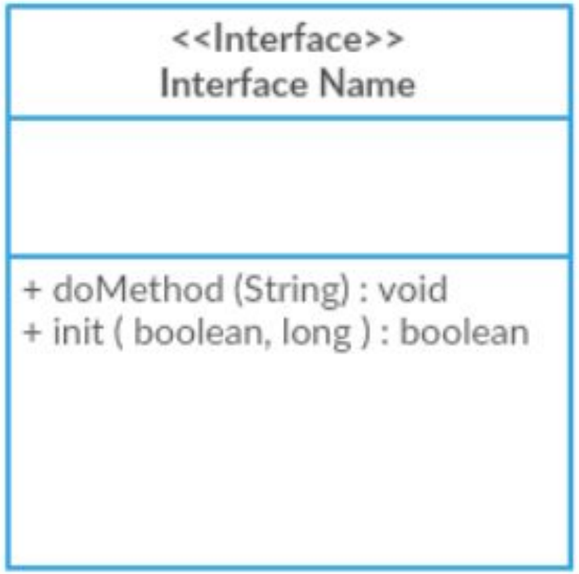
Componentes de una clase:

- **Nombre:** Identifica de forma única a la clase.
- **Atributos:** Son las propiedades o características de los objetos de esa clase. Se representan como variables.
- **Métodos:** Son las acciones o operaciones que pueden realizar los objetos de esa clase. Se representan como funciones.



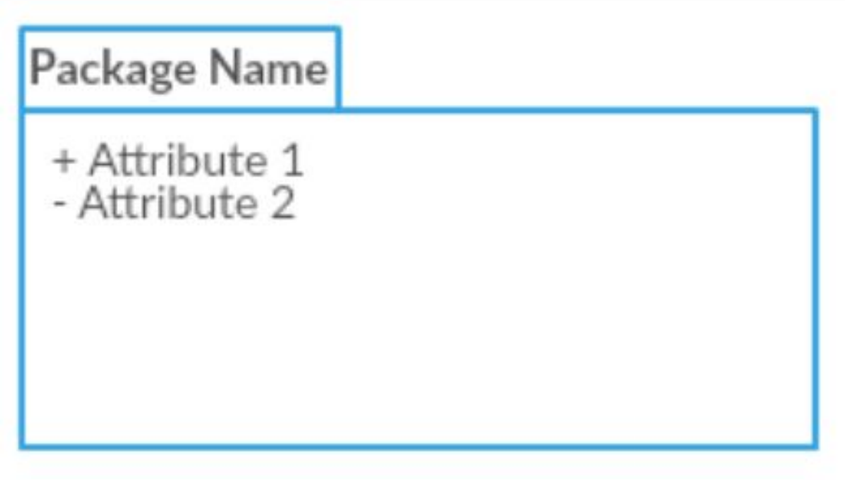
Interfaz

Una interfaz en UML define un contrato de comportamiento. Es decir, especifica un conjunto de métodos que una clase debe implementar, sin proporcionar una implementación concreta. Las interfaces promueven la programación orientada a interfaces y la reutilización de código.



Paquete

Un paquete en UML es un mecanismo para organizar elementos del modelo, como clases, interfaces, diagramas, etc. Sirve para agrupar elementos relacionados y mejorar la legibilidad del modelo.



Relación de diagrama de clases

Class Diagram Relationship Type	Notation
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	

- **Asociación:** Es una relación entre dos clases en la que una clase conoce o utiliza los servicios de otra clase.
- **Dependencia:** Es una relación entre dos clases donde una clase, conocida como la clase cliente, utiliza los servicios de otra clase, conocida como la clase proveedora, pero no hay una relación de propiedad o vida entre ellas.
- **Herencia:** Es una relación entre dos clases donde una clase (subclase o clase derivada) hereda atributos y métodos de otra clase (superclase o clase base).
- **Agregación:** Es una relación entre dos clases donde una clase es parte de otra clase. La agregación es una relación "todo-parte", donde las partes pueden existir independientemente del todo.
- **Composición:** Es similar a la agregación, pero en este caso, las partes no pueden existir sin el todo. Es una relación "todo-parte" más fuerte.

Multiplicidad

Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)

Diagrama de secuencia

Objetos: Representan las instancias de las clases que interactúan entre sí en el escenario específico que se está modelando. Los objetos se muestran como cajas rectangulares con el nombre de la instancia en la parte superior y el tipo de clase entre paréntesis.

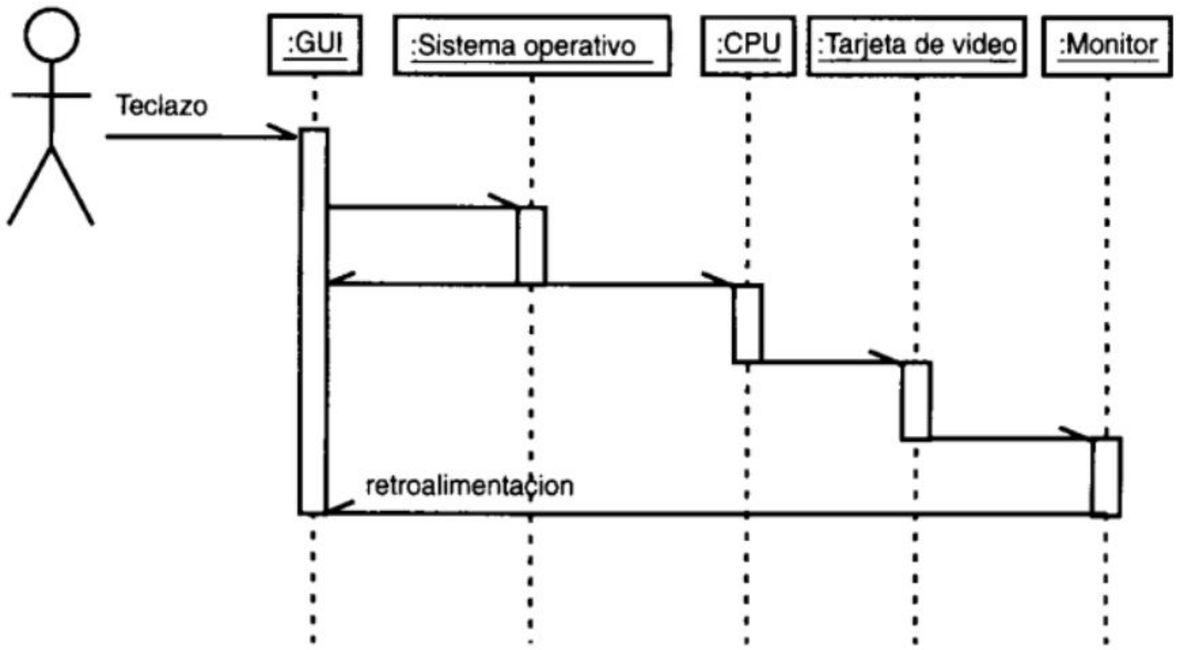
Líneas de vida (Lifelines): Representan la existencia temporal de un objeto durante la ejecución del escenario. Se dibujan como líneas verticales con un nombre de objeto al final.

Mensajes: Representan las interacciones entre objetos en forma de llamadas de método o envío de mensajes. Se muestran como flechas direccionales desde el remitente al destinatario, con el nombre del método o mensaje y, opcionalmente, los argumentos.

Activaciones: Representan el tiempo que un objeto está realizando una operación. Se dibujan como una caja rectangular en la línea de vida del objeto.

Fragmentos (Fragments): Representan estructuras de control condicional (como bucles o condicionales) o paralelismo en el flujo de interacción. Se pueden usar fragmentos como bucles (loop), alternativas (alt), opciones (opt), entre otros.

Diagrama de secuencia



La secuencia

Suponga que el usuario de una GUI presiona una tecla alfanumérica; si asumimos que utiliza una aplicación como un procesador de textos, el carácter correspondiente deberá aparecer de inmediato en la pantalla. ¿Qué ocurre tras bambalinas para que esto suceda?

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de vídeo.
5. La tarjeta de vídeo envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

Diagrama de secuencia