Conceptos básicos de OO

¿Qué es un patrón de diseño?

*"Cada patrón **describe un problema** que se repite una y otra vez en nuestro ambiente y **después describe el núcleo de la solución** a este problema, de forma tal que pueda utilizarse 1 millón de veces, sin tener que hacerlo igual 2 veces"*

Christopher Alexander

¿Qué es un patrón de diseño?

“Los patrones de diseño serán descripciones de objetos y clases comunicándose de forma particular para resolver un problema de diseño general en un contexto particular”

Elementos esenciales de un patrón

Nombre del patrón: un identificador (*handler*) que se utiliza para describir el **problema** de diseño, su **solución** y **consecuencias**; en 1 o 2 palabras.

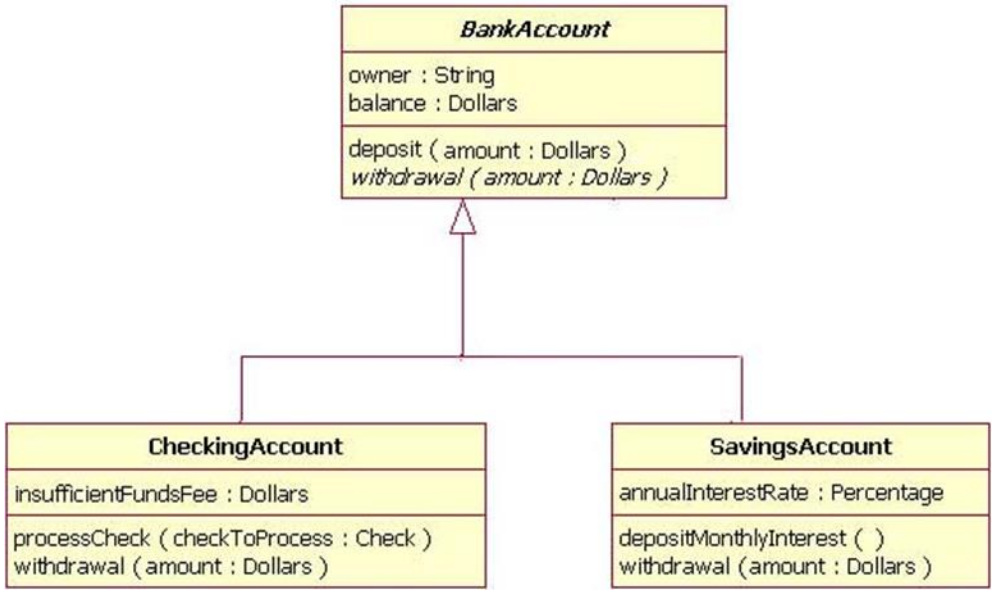
Problema: describe **cuando aplicar** el patrón. Explica el **problema** y su **contexto**.

Solución: describe **los elementos que conforman el diseño, sus relaciones, responsabilidades y colaboraciones**.

Consecuencias: los **resultados** y **compensaciones** de aplicar el patrón.

Interfaces, Tipos, Super/SubTipos

- Considere las siguientes clases:



- Determine:
  - La interface de la clase ***BankAccount*** y de la clase **CheckingAccount**.
  - Los tipos de la clases ***BankAccount*** y **SavingsAccount**
  - La relación de inclusión entre las interfaces de las clases ***BankAccount*** y **ChekingAccount**.
  - La relación de tipos (Supertipo y subtipo) entre las clases ***BankAccount*** y **CheckingAccount** en base a la inclusión de sus interfaces.

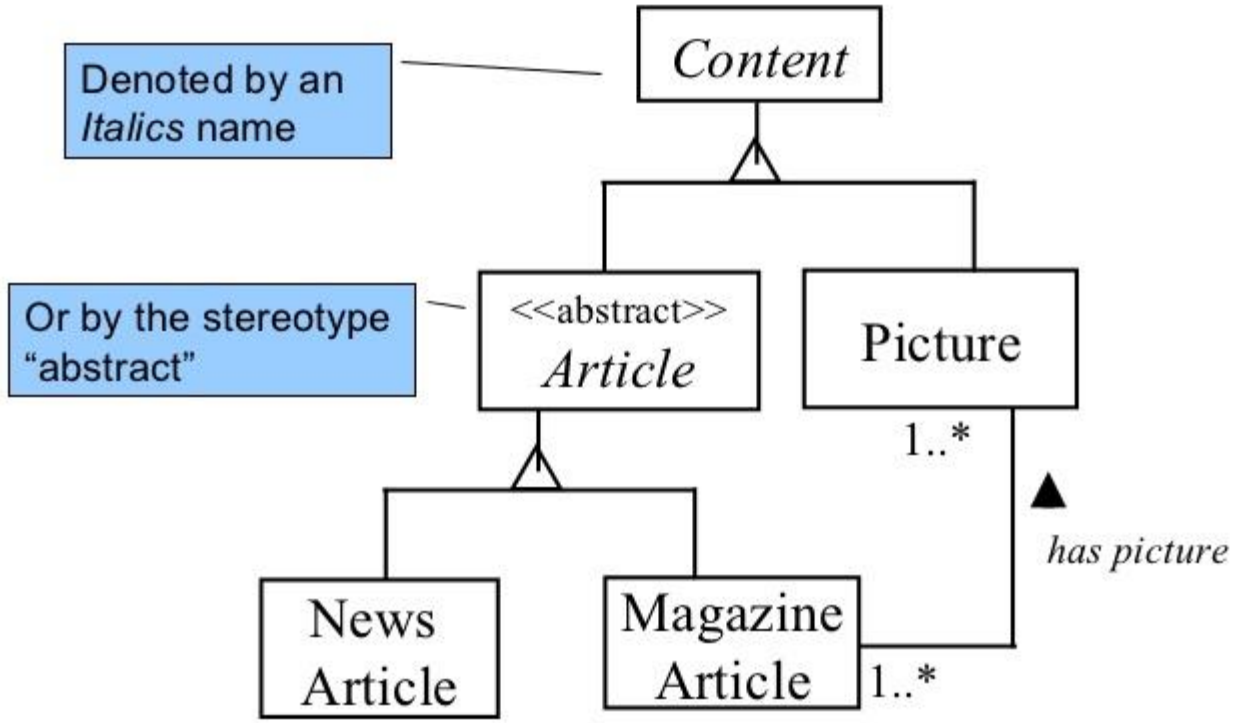
Para reflexión: responda en el chat

“¿Cuál es el propósito de las clases abstractas?”

Clases abstractas

Abstract Class

- A class that has no direct instances



En términos de la ***intención*** estudiada de las clases abstractas:

¿Cuál es la razón por la cual el diseñador de este sistema OO definió la relación entre la clase ***Article*** *y* ***News Article*** *y* ***Magazine Article****?*

¿Y entre ***Magazine Article*** *y* ***Picture****?*

Herencia de Clases/Herencia de Tipos

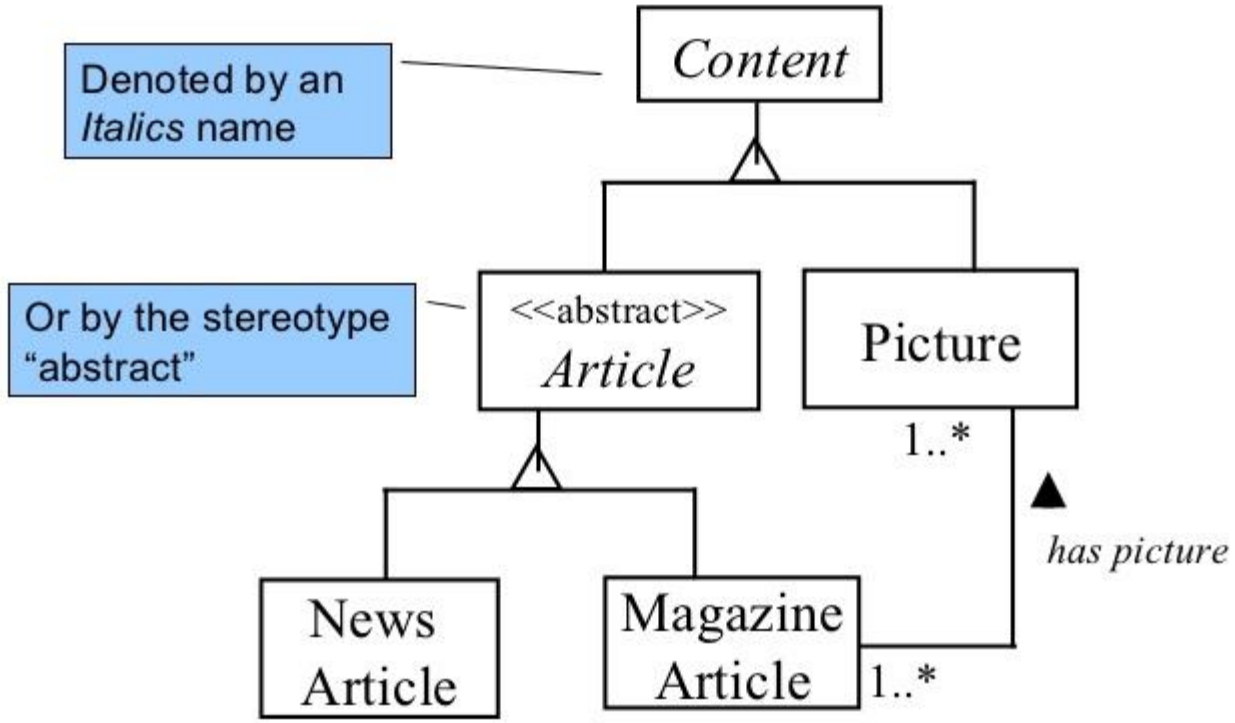
La **Herencia de Clase** define la **implementación** de un objeto en términos de la implementación de otro objeto. Es un mecanismo para compartición de código y representación.

La **Herencia de Tipos** describe cuando un objeto se puede utilizar en lugar de otro.

Herencia de Clases/Herencia de Tipos

Abstract Class

- A class that has no direct instances



¿Qué significaría la Herencia de Clases y la Herencia de Tipos entre las clases ***Article*** y **News Article** y **Magazine Article**?

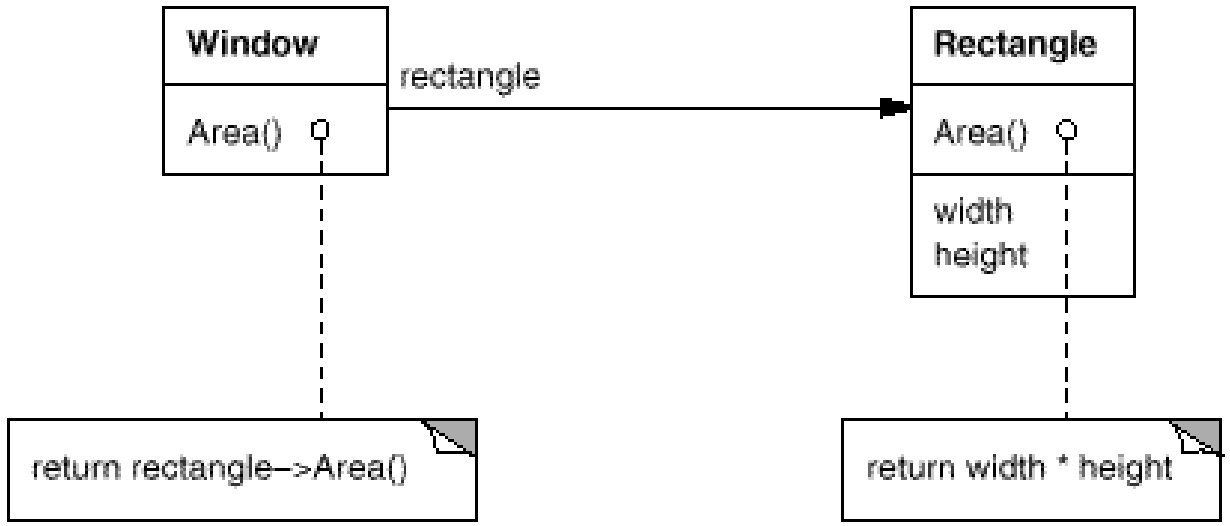
¿Y entre ***Content*** y **Magazine Article**?

Delegación

*En la **delegación**, **dos objetos están involucrados en el manejo de una petición***

*Un **objeto receptor** delega operaciones a su **delegado**.*

Ejemplo de Delegación



En lugar de que **Window** se subclasifique de **Rectangle**, **Window** reutiliza la funcionalidad de **Rectangle** por ***delegación***.

En lugar de la relación **Window IS-A Rectangle** se prefiere **Window HAS-A Rectangle**.

Delegación

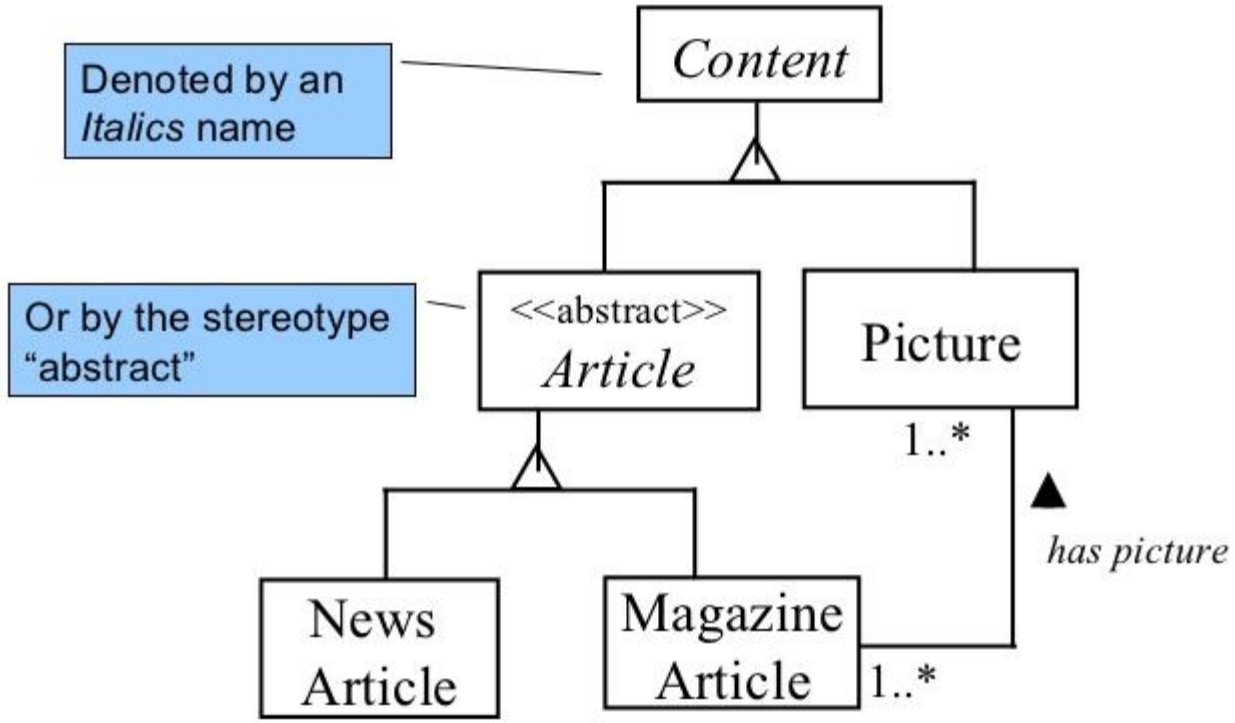
*La **principal ventaja** de la **delegación** es que permite **componer comportamientos en tiempo de ejecución**.*

***Window** podría ser **circular** en tiempo de ejecución simplemente reemplazando una instancia **Rectangle** por una instancia **Circle**, asumiendo ambas del mismo tipo.*

Delegación

Abstract Class

- A class that has no direct instances



¿Cómo podrías cambiar la representación con herencia de ***Article*** y **News Article** y **Magazine Article** por un diseño con delegación?

Represéntalo con un diagrama de clases UML

Problemas típicos que resuelven los patrones de diseño

- *Crear un objeto especificando explícitamente una clase.*
- *Dependencia en operaciones específicas.*
- *Dependencia en plataformas HW y SW específicas.*
- *Dependencia en representaciones o implementaciones objetuales.*
- *Dependencias algorítmicas.*
- *Alto acoplamiento.*
- *Extensión de funcionalidad mediante subclasificación.*
- *Inhabilidad de alterar convenientemente una clase.*