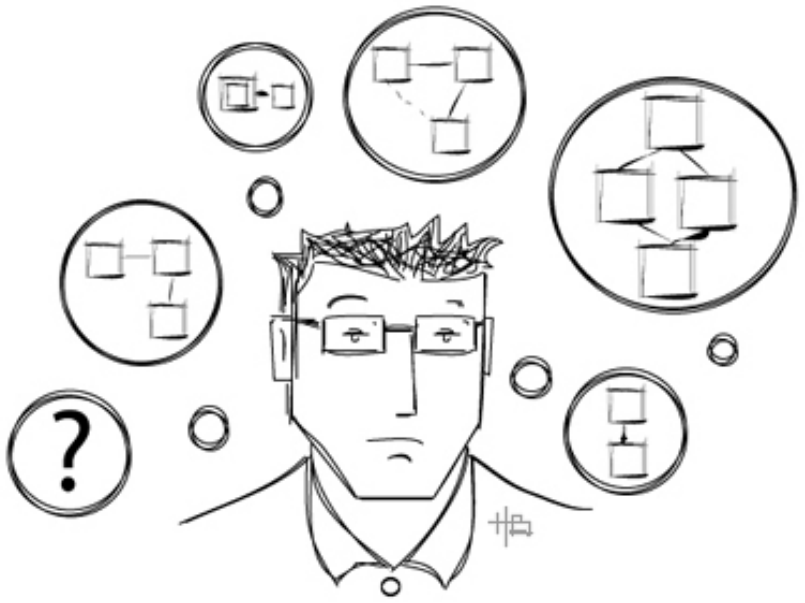
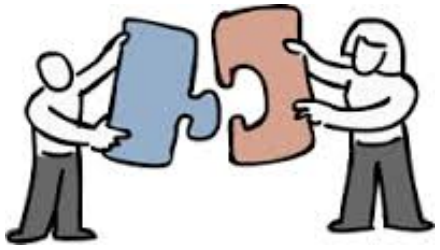
- ¿Por qué son importantes los patrones de diseño?



¿Por qué son importantes los patrones de diseño?

- ¿No alcanza con el Paradigma Orientado a Objetos?

- El Paradigma OO permite diseñar componentes reusables y extensibles



¿Por qué son importantes los patrones de diseño?

- Diseñar objetos reusables es muy difícil:
  - ¿cuál es la granularidad correcta?
  - ¿cuándo conviene usar herencia y cuándo composición?
  - ¿cuándo una jerarquía es buena?
  - ¿cuándo una relación de conocimiento es buena?
  - ¿cómo hacer un diseño específico al problema pero adaptable a los cambios?

- Según GOF los patrones son:
  - descripciones de clases y objetos y de la comunicación entre ellos, que resuelve un problema de diseño general y debe especializarse a un contexto particular

- ¿Cuáles son esas descripciones?

Mensajes abstractos

- Son importantes para establecer el protocolo de una jerarquía de clases.
- Aseguran que todo objeto instancia de una subclase puede responder a ese mensaje.
- No se especifica comportamiento, ya que a nivel de la superclase no se puede prever.

- *En el diagrama de clases se identifican por escribirse en letra itálica.*
- *¿Cómo se identifican en Smalltalk?*
- ¿Qué ocurre con el “method look-up”?

Clases abstractas

- Son clases a partir de las cuales no pueden crearse instancias.

- ¿Entonces, para qué sirven?
  - Para establecer un protocolo común
  - Para factorizar un comportamiento común
  - No necesitan estar completamente implementadas

Polimorfismo - repaso

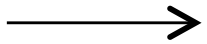
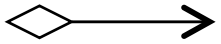
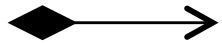
- Definición: dos o más objetos son ***polimórficos*** con respecto a un mensaje, si todos pueden entender ese mensaje, aún cuando cada uno lo haga de un modo diferente.

  - *Mismo **mensaje** puede ser enviado a diferentes objetos.*
  - *Binding dinámico*
  - *Distintos receptores reaccionan diferente (diferentes **métodos**)*

Relación de conocimiento: Colaboraciones

- Un objeto puede cumplir sus responsabilidades:
  - realizando el cómputo necesario por sí mismo
  - colaborando con otros objetos
- Un objeto colabora con otro cuando le envía mensajes para cumplir una responsabilidad
- Una colaboración puede verse como un *contrato* entre un servidor que presta un servicio y un cliente que requiere el servicio.
- Identificar servidores y clientes de un contrato permite encontrar nuevas responsabilidades o responsabilidades mal asignadas.

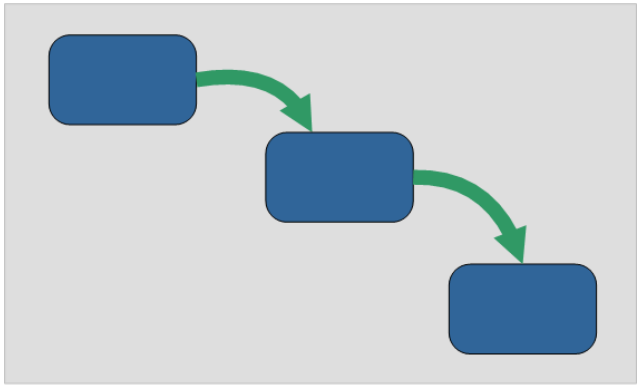
Tipos de relaciones

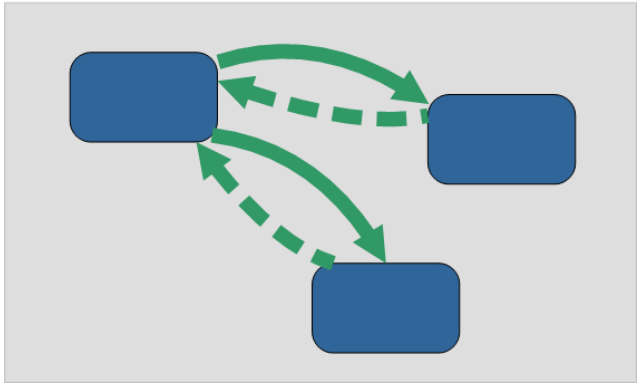
- El GOF distingue 2 tipos de relaciones:
  - conocimiento 
  - agregación o composición 
- Aunque no podamos distinguirlo en el código, la semántica de la relación es diferente y puede implicar nuevas responsabilidades
- Actualmente en UML se distingue entre agregación y composición: 
  - Por ejemplo, cuando un objeto está compuesto de otros, su borrado debería causar el borrado de las partes, y las partes sólo pueden pertenecer a un todo.

Relaciones

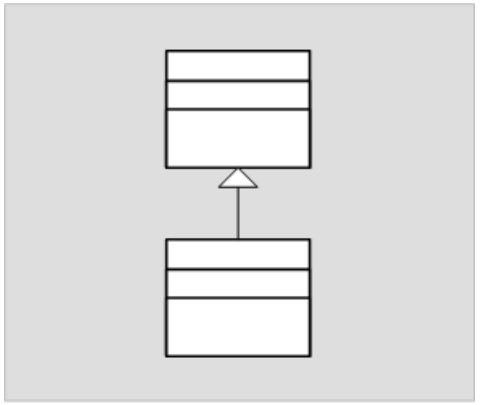
- Patrones estructurales y de comportamiento

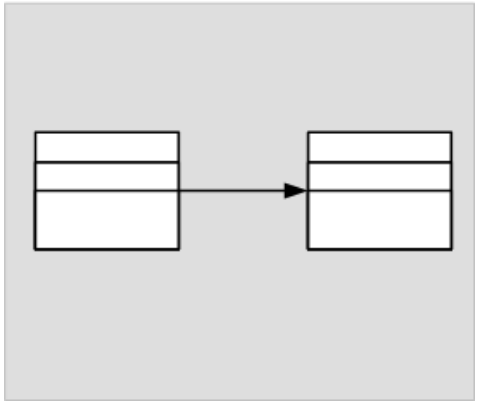
- Delegación y Colaboración



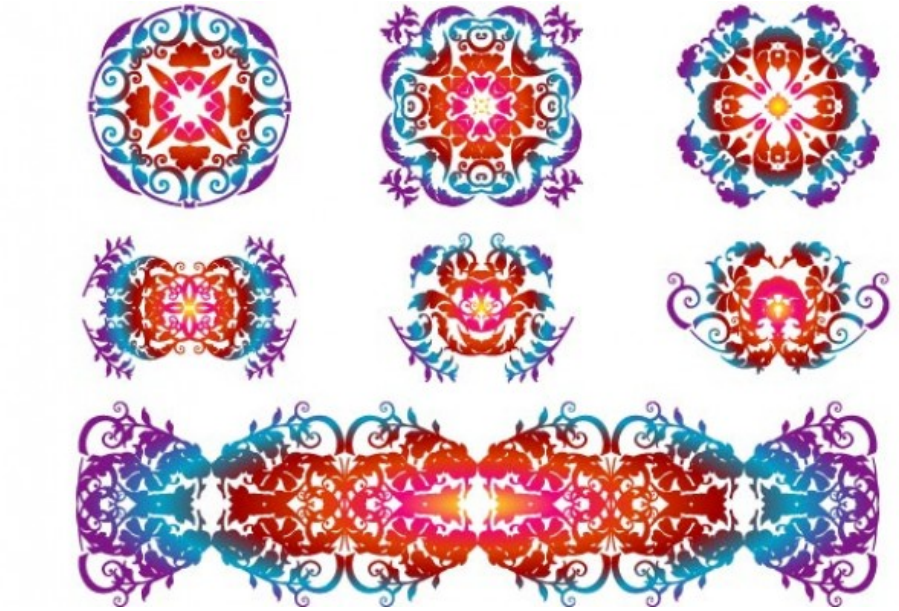


- Herencia y Composición





- PLoP
- PLoP
- PLoP
- Llega a Argentina en 2016



PLoP en Argentina!

- **PLoP: Conference on Pattern Languages of Programs**
- Desde 1994
- Lugar original: Allerton Park, Monticello, Illinois
- Luego: EuroPLoP, ChiliPLoP (Arizona), KoalaPLoP (Australia), MensorePLoP (Japon), AsianPLoP, VikingPLoP (Países Bajos) y SugarLoafPLoP
- La 1ra SugarLoafPLoP fue en Brasil en 2001.

- Este año: 11th LatinAmerican Conference on Pattern Languages of Programs (**SugarLoafPLoP 2016**) en Buenos Aires, 16-18 de noviembre:

http://hillside.net/sugarloaf/2016/

Patrones ya vistos

- Adapter (estructural)
- Composite (estructural)
- Template Method (comportamiento)
- Strategy (comportamiento)

Hoy veremos 2 nuevos patrones estructurales