Aspectos Básicos de Diseño

Universidad de Chile

Departamento de Ciencias de la Computación

Prof.: Nancy Hitschfeld Kahler

Contenido

- Diseño de una buena clase
- Relaciones entre clases
- Notación UML
- Uso correcto de herencia
- Patrones de diseño
- Evaluación de diseños: Métricas

¿Cómo diseñar una buena clase? [Mey97]

- Clase conocida por su interfaz (métodos)
- Principio de lista de compras
- Descripción simple y coherente (no más de una página)
- Nombre apropiados
- Uso de contratos: precondiciones, post condiciones e invariantes (depuración y documentación) [Mey97]
- Mecanismo de excepción para situaciones anormales

Pasos a seguir: (enfoque minimal)

- Encontrar objetos
- Diseñar su métodos públicos (interfaz)
- Definir su implementación
- Proceso iterativo e incremental

¿Qué puede ser un objeto?

[Daniel Halbert y Patrick O'Brien: “Using Types and ...”]

- Cosas reales o abstractas
  - artículos, cuentas y personas
  - pila, cola y grafo
- Procesos
  - ordenar, iterador, formateador

Cómo diseñar la interfaz?

- Donde poner una funcionalidad?
  - Ejemplo: inscribir un estudiante en un curso
    - En la clase Curso?
    - En la clase Estudiante?
    - Crear una nueva clase Inscripción?
  - No hay una regla clara: depende de los requerimientos de la aplicación
- Examinar:
  - Reusabilidad
  - Complejidad
  - Aplicabilidad
  - Conocimiento de la implementación

Reusabilidad

- Es útil este comportamiento en más de un contexto?
  - Crear clase si el comportamiento puede ser reusado en varios contextos u otras clases
- Ejemplo:
  - Recorrer una lista : el mismo recorrido puede ser aplicado a varias listas. (Patrón de diseño **iterator**)
  - Concepto de rectángulo: reusado en ventana y figuras gráficas

Aplicabilidad

- Qué tan relevante es un comportamiento?
- Desea la mayor parte de los clientes usar este comportamiento?
- Ejemplos:
  - Convertir a mayúsculas como método en clase String (es solo relevante aquí)
  - Parsing de una línea en clase aparte (no es método clásico asociado a String). En Java?

Complejidad

- Qué tan difícil es implementar este comportamiento?
  - Crear una clase si el comportamiento es complejo
- Ejemplo:
  - Calcular el área es simple; incluir en Rectángulo
  - Parsing de una línea es complejo; modelar este porceso como una clase
  - Intersección de figuras: va en mas de una clase el mismo método y es complejo; Modelar como una clase

Conocimiento de la implementación

- En qué grado su implementación depende del tipo al que es asignado

Muy dependiente => incluirlo en clase asociada

- Que decidir en caso que dos criterios den resultados distintos?
  - Usar criterio de mayor peso (depende de los requerimientos de la aplicación)

Relaciones entre clases

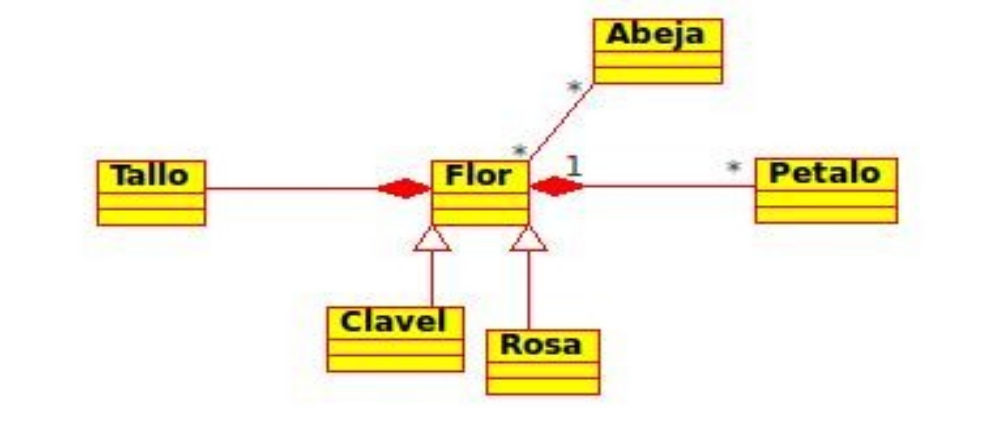
- Especialización/generalización
  - Herencia
  - IS-A?
- Parte/Todo
  - IS-PART-OF, HAS-A?
  - Agregación y composición
- Asociación
  - Colaboración
  - Independencia
- De uso
  - Relación más débil

Relaciones entre clases (2)

Ejemplo:

- Una rosa es una flor
- Existen rosas de diferentes colores
- Una flor se compone de pétalos, aroma, etc.
- Las abejas usan el néctar de las flores para alimentarse
- Las abejas ayudan a polinizar las flores
- El clavel es otra flor

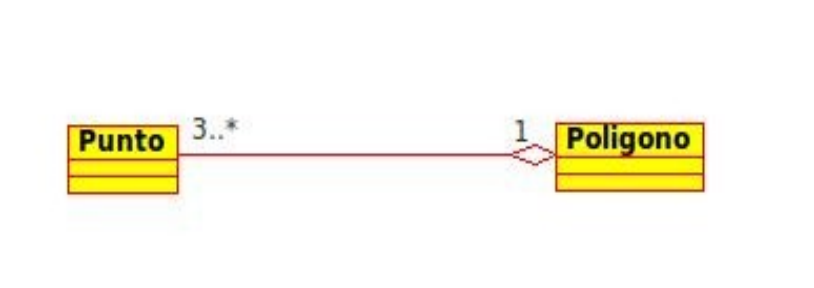
Relaciones entre clases (3)



Relaciones entre clases (4)

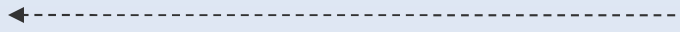
Diferencia entre composición y agregación

- Composición: tiempo de vida de la variable de instancia está ligada a la del objeto que la contiene
- Agregación: tiempo de vida de la variable de instancia es independiente del objeto que la contiene.

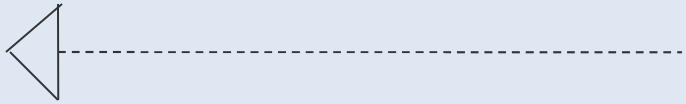


Relaciones entre clases (5)

- Relación de uso:



- Implementación de una interfaz



- Asociación dirigida

¿Cuándo usar herencia? [Alb87]

- Subtipos (forma natural: subconjuntos)

¿Por qué?

- El objeto heredado válido en el contexto de clase padre
- El objeto heredado redefine métodos y/o agrega nuevos

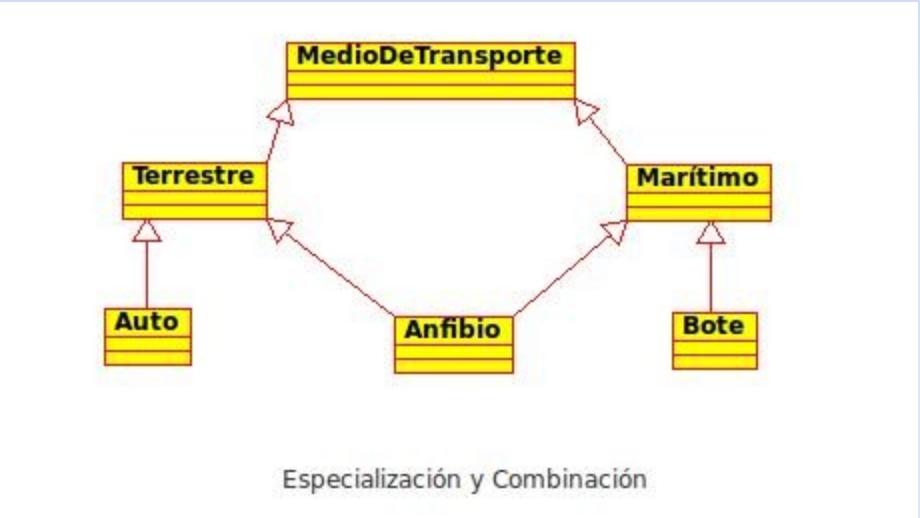
Problemas en caso contrario:

- Malos diseños
- Falsas expectativas

¿Cuándo usar herencia?(2)

- Especialización
  - subclases subconjuntos de clase padre
- Implementación
  - misma interfaz, distinta implementación
- Combinación
  - subclase subconjunto de dos o más clases

¿Cuándo usar herencia?(3)



Especialización y Combinación

¿Cuándo NO usar herencia?

- Generalización: subclase generalización de clase padre
- Varianza: subclase hermano de clase padre

Errores comunes:

- Heredar de partes
- Heredar privadamente (C++). Alternativa: agregación o composición