Introducción a UML

Programación I

Grado en Ingeniería Informática
JCRdP y JDGD

Introducción

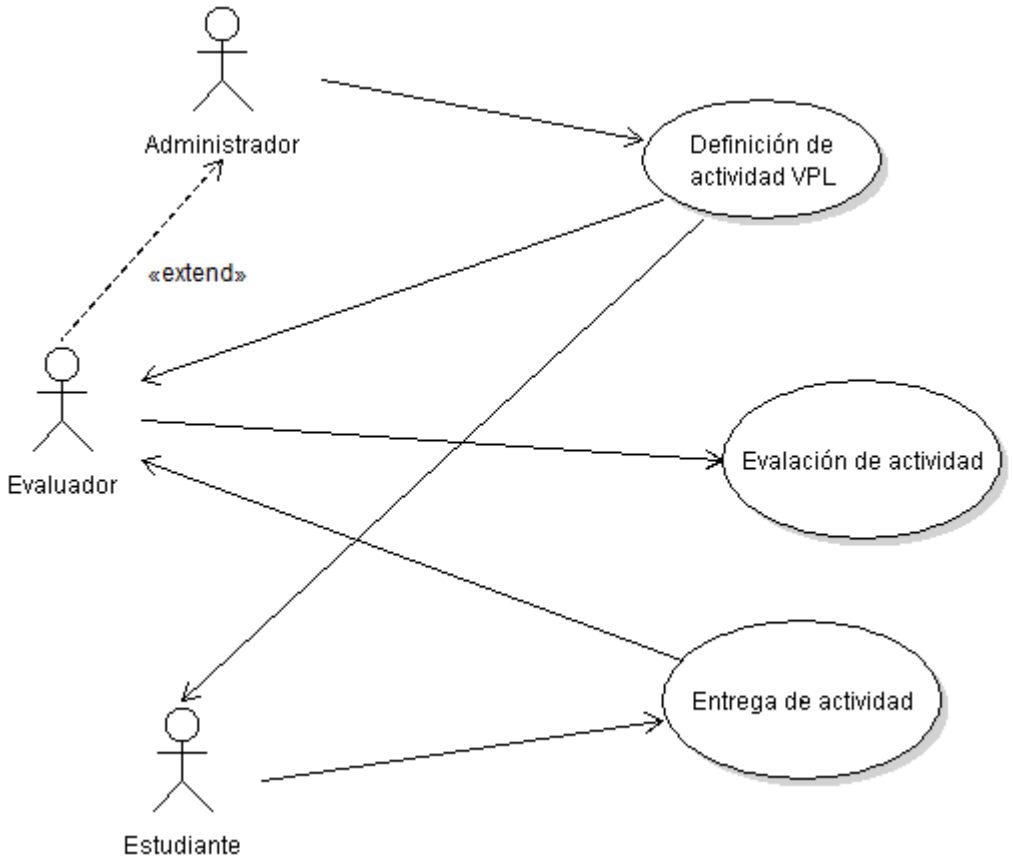
- Es un lenguaje gráfico de modelado de software
- Nos permite crear un **plano** del sistema
- Es aceptado por ISO como el lenguaje estándar de modelado de sistemas software
- No es un método de desarrollo de software pero sí compatible con muchos
- El modelo completo requiere documentación escrita

Introducción

- Define varias tipologías de **diagramas** que representan distintos aspectos del sistema
- Dos vistas diferentes del sistema:
  - **Vista estática** o estructural: Diagrama de clases, diagrama de componentes, etc.
  - **Vista dinámica** o de comportamiento: diagrama de comunicación, diagrama de estado, etc.

Diagrama de casos de uso

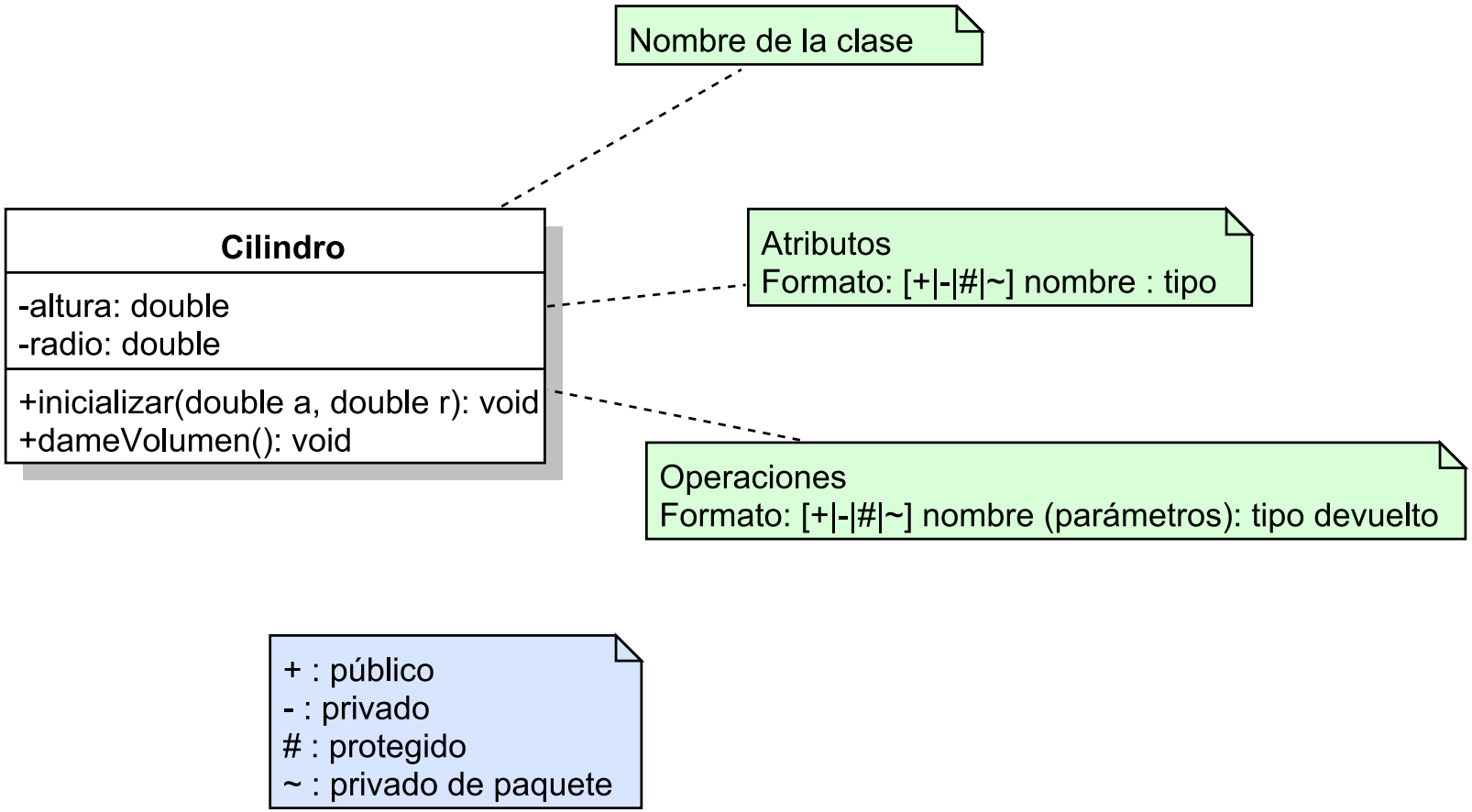
- Captura parte de los requerimientos del sistema
- Muestra las interacciones de los distintos tipos de usuario con el sistema



Diagramas de clases

- Representa las clases y sus relaciones
- Resultado del análisis y diseño de la vista estática de la aplicación
- Describe las responsabilidades del sistema
- Base de los componentes del diagrama de despliegue
- Permite generación automática de código e ingeniería inversa

Diagramas de clases
Dibujo de clases



Diagramas de clases
Relaciones entre clases

- En una aplicación los distintos objetos no actúan aislados, colaboran e intercambian información manteniendo distintos tipos de relaciones:
  - Generalización (herencia)
  - Composición
  - Agregación
  - Asociación
  - Dependencia

- Las relaciones están nombradas de la más fuerte a la más débil.

Diagramas de clases
Representación/Multiplicidad relaciones

- Se representan con líneas que unen las clases relacionadas
- Las líneas pueden ser continuas o discontinuas y terminar en diferentes símbolos
- En algunas relaciones se puede establecer el número de objetos que intervienen
- El número o rango de objetos se establece al lado de la clase correspondiente
- Formato:
  - Número fijo: 1, 2, 3, ...
  - Rango: 1..3, 0..5
  - Rango sin límite: 0..*, 3..*

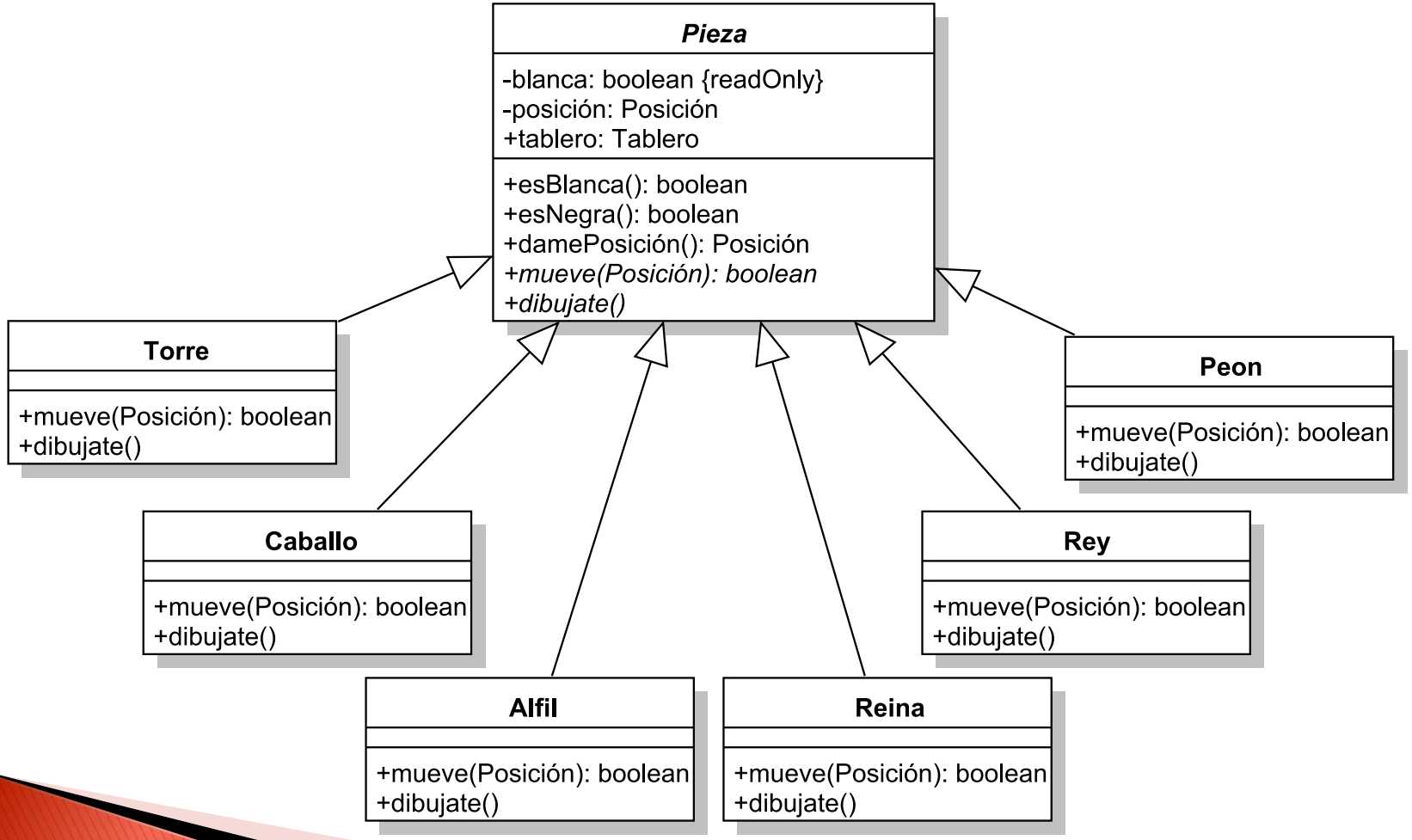
Diagramas de clases
Generalización (herencia)

- La generalización (herencia inversa) es la relación "es un"
- El término tiene que ver con la fase de diseño en que se toman los elementos comunes a distintas clases para generalizarlas en una superior
- Gráficamente se indica con una flecha con línea continua y punta en triángulo sin relleno que va de la clase derivada a la base
- No existe multiplicidad

Diagramas de clases
Ejemplo: Generalización (herencia)

- Se desea representar los elementos de un juego de ajedrez
- Se tienen los diferentes tipos de piezas de ajedrez que son: las torres, caballos, alfiles, reinas, reyes y peones
- Las piezas se encuentra, opcionalmente, en una posición en el tablero, tienen un color blanco o negro, se pueden dibujar y mover

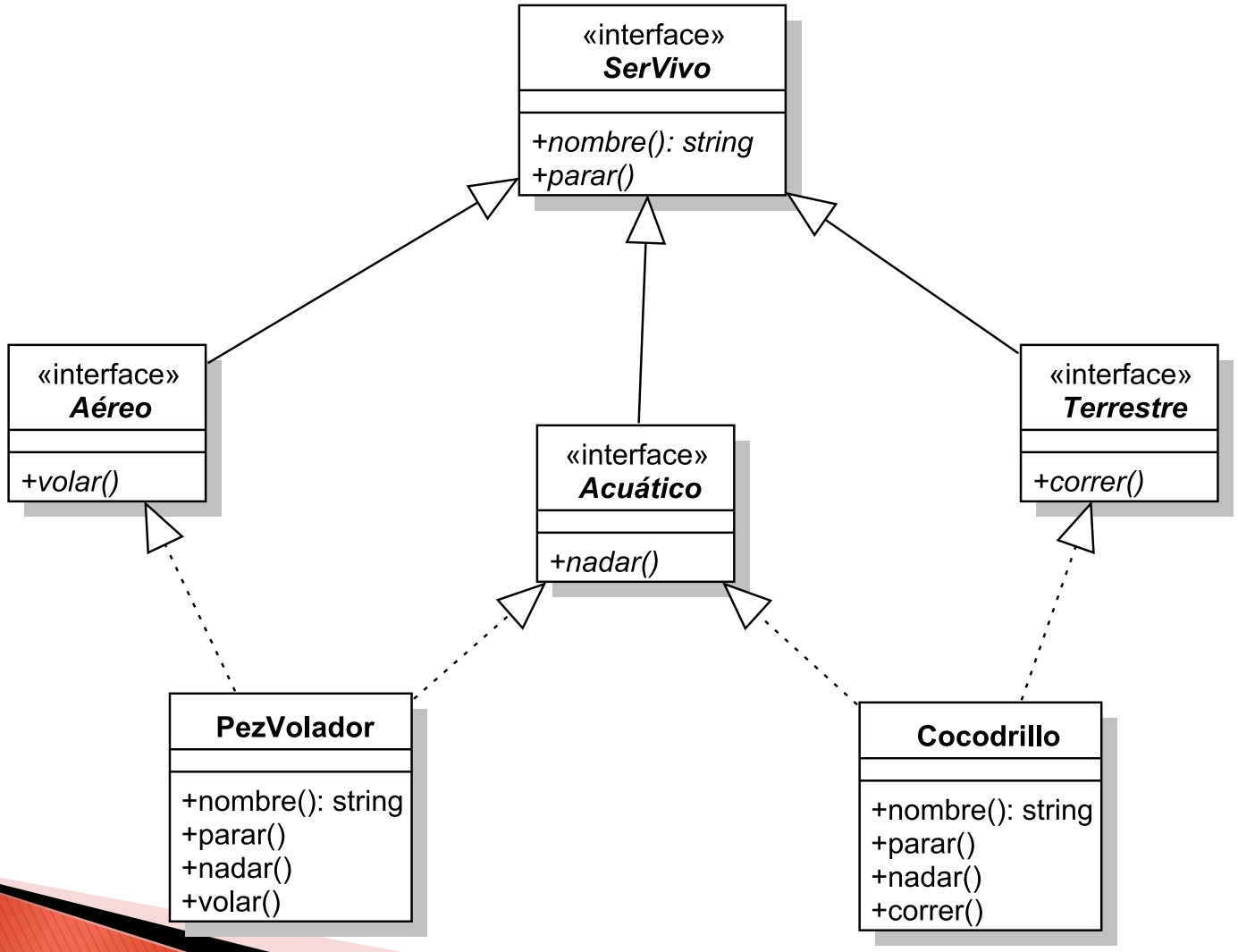
Diagramas de clases
Generalización (herencia)



Diagramas de clases
Ejemplo: Generalización (interface)

- Se desea representar seres vivos que se desplazan y se le puede pedir que paren su movimiento. Los seres vivos pueden ser acuáticos, que pueden nadar, aéreos, que pueden volar y, terrestres, que pueden correr
- Los tipos concretos de animales que se quiere representar son los cocodrilos y los peces voladores

Diagramas de clases
Implementación de interfaz



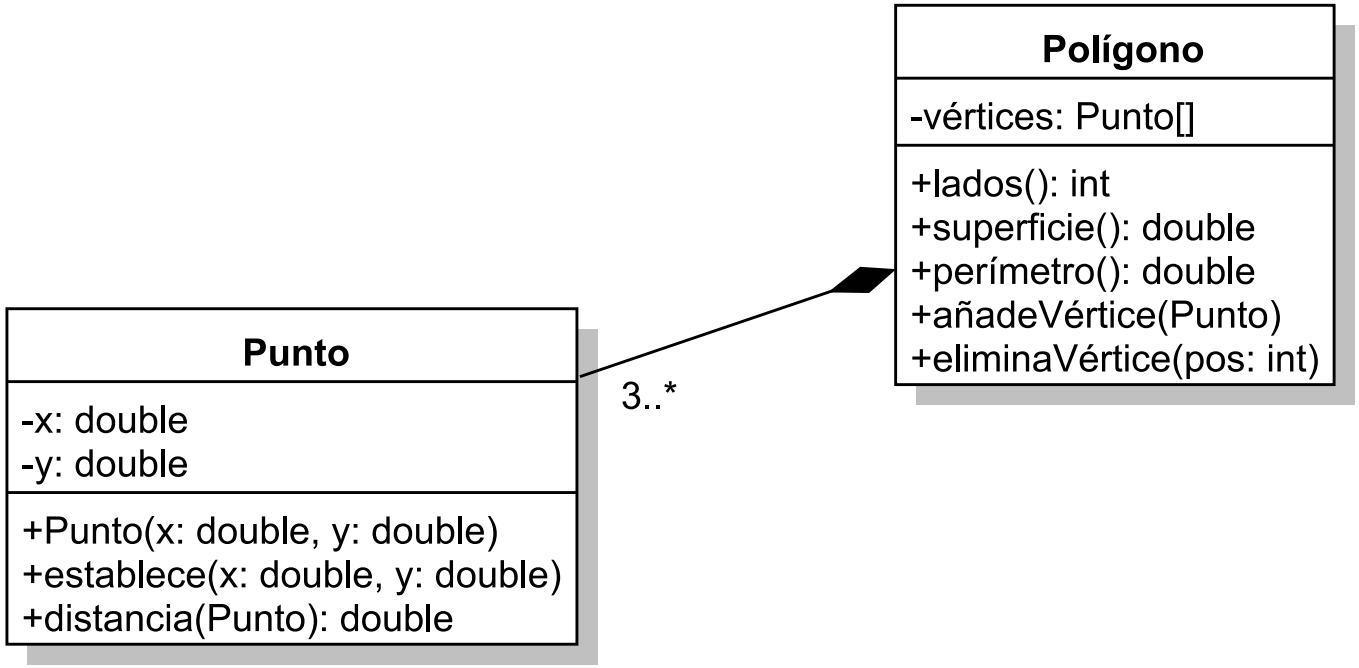
Diagramas de clases
Composición

- La composición es un tipo fuerte de relación en el que un objeto "tiene" otro(s) objeto(s) que lo forman
- Un objeto tiene otros que lo componen
- Aunque los objetos que "componen" tienen entidad independiente suficiente su existencia está ligada al objeto compuesto
- Cuando el objeto compuesto desaparece, desaparecen los que lo componen

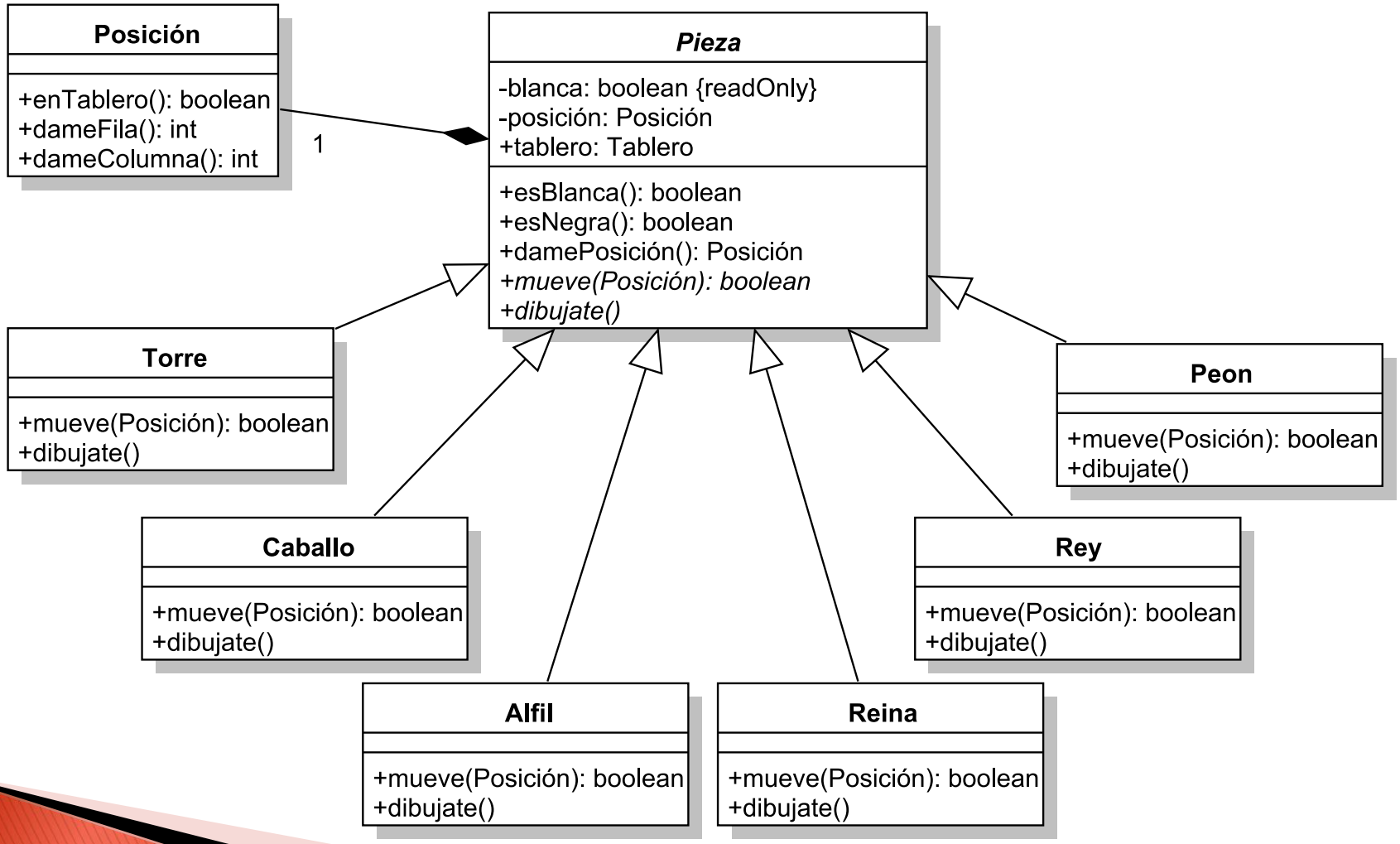
Diagramas de clases
Ejemplo: Composición

- Se desea representar un polígono cerrado. El polígono se sitúa en un plano y sus vértices se representan por Puntos en un plano
- Los puntos se representan en coordenadas cartesianas, tienen un constructor que los inicializa, se puede modificar su valor y calcular la distancia a otro punto
- El polígono dispone de operaciones como: número de lados, superficie, perímetro, añade punto, elimina punto

Diagramas de clases
Ejemplo: Relación de Composición



Diagramas de clases
Ejemplo: Relación de Composición

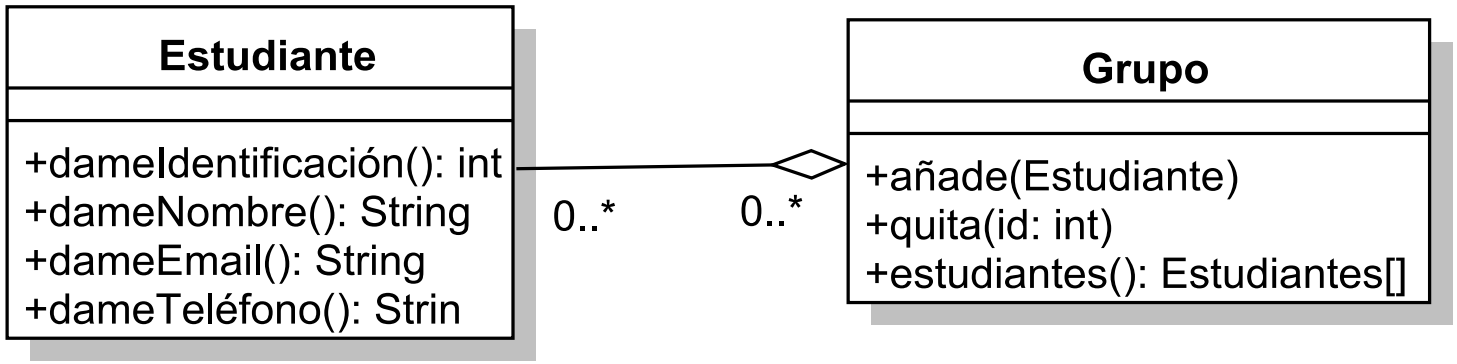


Diagramas de clases
Agregación

- La agregación es similar a la composición, "tiene" otro(s) objeto(s) que lo forman
- Aunque los objetos que lo forman tienen entidad y vida independiente al objeto que los contiene
- Cuando el objeto compuesto desaparece, los objetos agregados siguen existiendo

Diagramas de clases
Ejemplo: Agregación

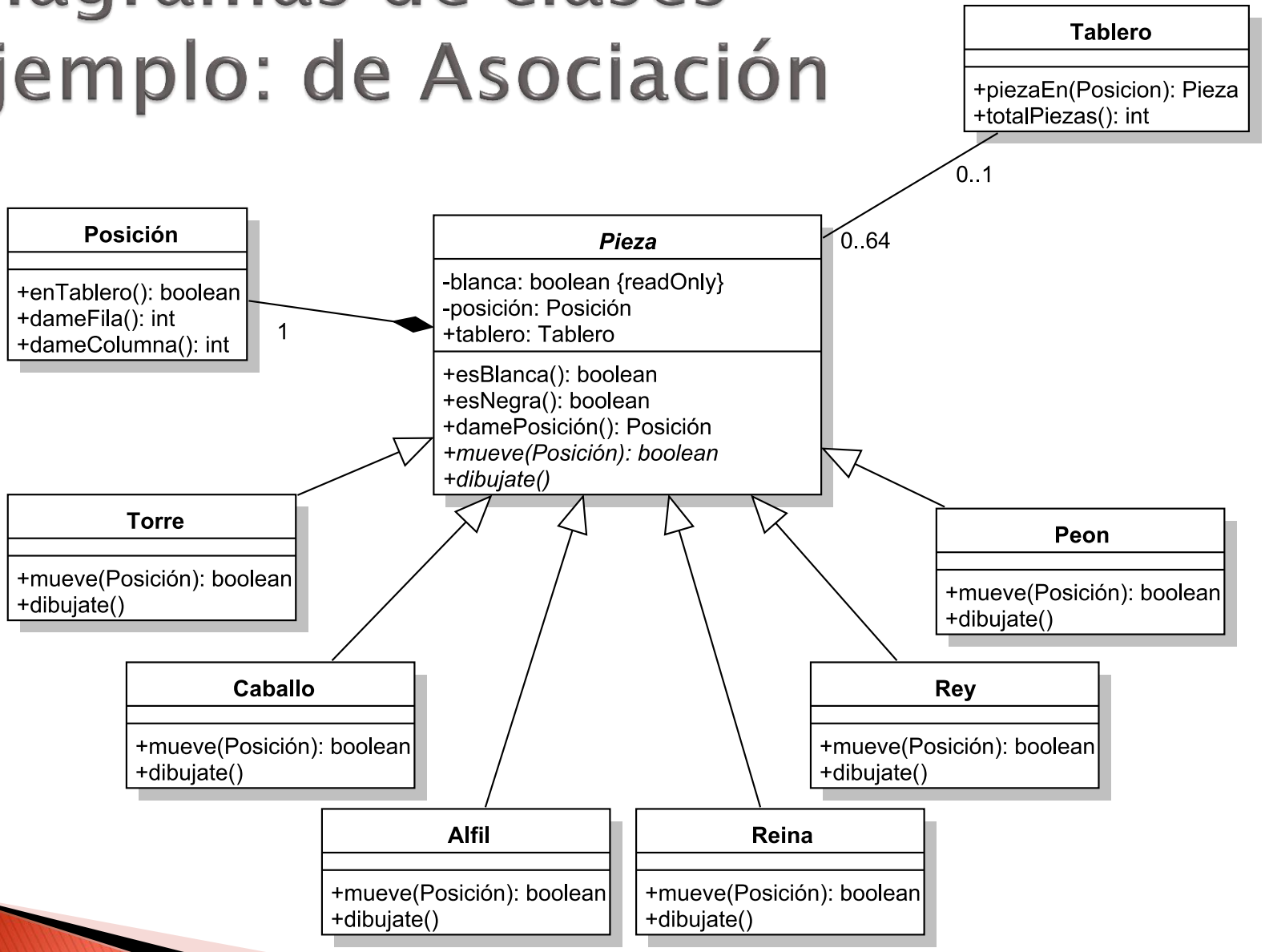
- Se tiene una clase que representa estudiantes y otra que representa un grupo de estudiantes
- Los estudiantes pueden pertenecer a varios grupos



Diagramas de clases
Relación de Asociación

- Se usa cuando se tiene algún tipo de asociación entre los objetos de dos clases pero sin llegar a ser tan fuerte como una agregación
- Es la relación más común
- Se mantiene durante algún tiempo de vida de los objetos implicados
- Por ejemplo: se puede tener una factura que este asociada a varias entregas de productos. La importante relación entre estas dos clases no encaja en otras relaciones

Diagramas de clases
Ejemplo: de Asociación



Diagramas de clases
Dependencia

- La dependencia es la relación más débil entre clases
- No es aplicable cuando se tiene otras relaciones más fuertes
- Se emplea para reflejar una clase hace uso de otra, normalmente como parámetro o como variable local de un método
- No se usa siempre, se emplea para destacar dependencias importantes

Control de calidad

- Cohesión: medida de clase bien definida y cumplimiento de responsabilidades mediante una interfaz apropiada
- Acoplamiento: medida de interconexión entre clases o subsistemas.
- Correctitud: Cada acción tiene el efecto esperado

- Se pueden estudiar mediante preguntas.
- A continuación se muestra ejemplo de preguntas sobre la clase Cilindro

Comprobaciones de cohesión

- ¿Describe claramente la clase el nombre 'Cilindro'?
- ¿Es 'Cilindro' un nombre o un sintagma nominal?
- ¿Podría malinterpretarse el significado de 'Cilindro'?
- ¿Debería Cilindro estar en su propia clase o ser un atributo de otra?
- ¿Hace Cilindro exactamente una sola cosa y la hace bien?
- ¿Podría Cilindro ser dividido en dos o más clases?

Comprobación de correctitud

- ¿Comienzan todos los atributos de Cilindro con valores significativos?
- ¿Podría escribir un invariante para esta clase?
- ¿Establecen la invariante de la clase todos los constructores?
- ¿Mantienen la invariante de la clase todas las operaciones?

Comprobación de relaciones

- ¿Planea crear subclases de Cilindro?
- ¿Podría eliminar del modelo a Cilindro?
- ¿Hay alguna otra clase en el modelo que deba ser revisada o eliminada porque sirve al mismo propósito que Cilindro?
- ¿Por qué motivos será actualizada una instancia de Cilindro?
- ¿Existe algún otro objeto que deba ser actualizado cuando sea actualizado Cilindro?

Bibliografía

- Unified Modeling Language http://www.uml.org/