

TEMA 5-A02: Diagramas de clases.

Índice

1.A02. Diagramas de clases.....	2
1.1 Introducción.....	2
1.2 Actividade.....	2
Introdución ós diagramas de clases.....	2
Os diagramas estruturais de UML.....	2
Propósito e función dos diagramas de clases.....	3
Elementos dos diagramas de clases.....	3
Clase, atributo, operación.....	3
Recomendacións para a elaboración dos diagramas.....	4
Asociación.....	6
Xeneralización.....	8
Clases abstractas.....	9
Dependencia.....	10
Interface.....	10
Composición débil ou Agregación.....	11
Composición.....	12
Diagramas de obxectos.....	13
1.3 Tarefas.....	13
1.3.1 Tarefa 1. Interpretación dun diagrama de clases.....	13
1.3.2 Tarefa 2. Elaboración dun diagrama de clases.....	14
1.3.3 Tarefa 3. Elaboración dun diagrama de clases.....	14
1.3.4 Tarefa 4. Elaboración dun diagrama de clases.....	15
1.3.5 Tarefa 5. Elaboración dun diagrama de clases.....	15
1.3.6 Tarefa 6. Elaboración dun diagrama de clases.....	15

1. A02. Diagramas de clases.

1.1 Introducción

Na actividade que nos ocupa aprenderanse os seguintes conceptos e manexo de destrezas:

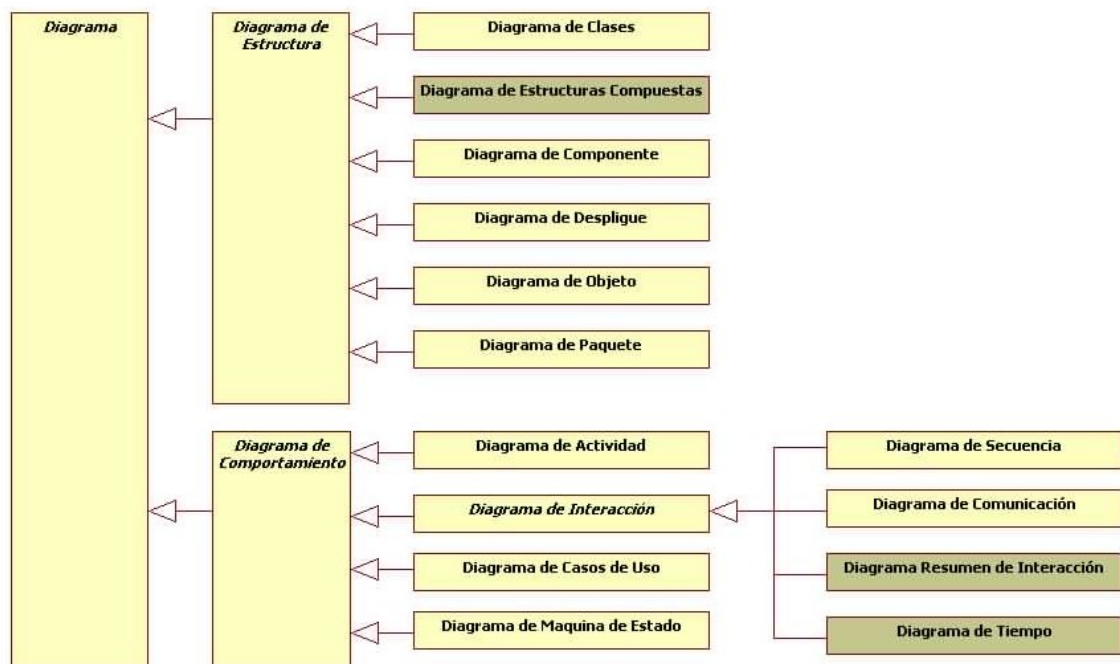
- Comprender o propósito e función dos diagramas de clases.
- Manexar diagramas de clases, interpretando correctamente diagramas xa feitos e desenvolvendo diagramas novos a partir de escenarios propostos.

1.2 Actividade

Introdución ós diagramas de clases

Os diagramas estruturais de UML

Os diagramas de clases son sen lugar a dúbidas os diagramas máis utilizados no modelado de sistemas orientados a obxectos e pertencen o grupo dos chamados *diagramas estruturais* de UML. O conxunto de diagramas de UML subdivídese habitualmente, tal e como pode verse na seguinte figura¹, en dous grandes subconxuntos: *diagramas estruturais* (ou *de estrutura*) e *diagramas de comportamento*.



Os diagramas estruturais *presentan elementos estáticos do modelo*, tales como clases, paquetes ou compoñentes; en tanto que os diagramas de comportamento *mostran a conduta en tempo de execución do sistema*, tanto visto como un todo como das instancias ou obxectos que o integran.

Os diagramas sinalados nunha cor distinta foron incorporados na versión 2 de UML polo que resultan ser dos menos coñecidos e empregados.

Propósito e función dos diagramas de clases

Os diagramas de clases permítennos representar unha vista dos compoñentes estáticos dun sistema, xa sexan estas clases ou módulos, indicando as relacións entre estas e os atributos (datos) das clases, así como os seus métodos (código).

Os diagramas de clases poden ser utilizados para presentar a vista estática do *modelo de dominio* – modelo da fase de análise que serve para a comprensión do entorno ó cal o sistema ten que servir ou emular -, do *modelo de deseño* - encárgase de refinar a arquitectura definida na fase de análise adaptándoa ao ambiente de implementación-, ou ben, do *detalle da implementación* dun sistema nunha linguaxe de programación orientada a obxecto, como Eiffel, Java ou C++. Para todos estes usos, o que se desexa é expresar as unidades en que o código se organiza -as clases- así como algunhas características destas, como son as súas relacións, atributos e métodos.

Aínda que a especificación de UML fala de diagramas diferentes para os paquetes e as clases, é válido combinalos dando lugar a un diagrama que mostra simultaneamente clases e paquetes. Isto axuda a documentar elementos que estando en distintos paquetes gardan algunha relación entre eles.

É importante mencionar tamén que os autores de UML consideran os diagramas de clases como un superconxunto dos coñecidos diagramas de entidade/relación empregados para deseñar bases de datos relacionais.

Elementos dos diagramas de clases

Clase, atributo, operación

Unha *clase* é a descrición dos atributos e operacións que comparten un conxunto de obxectos. En UML represéntanse graficamente cun rectángulo de tres filas.

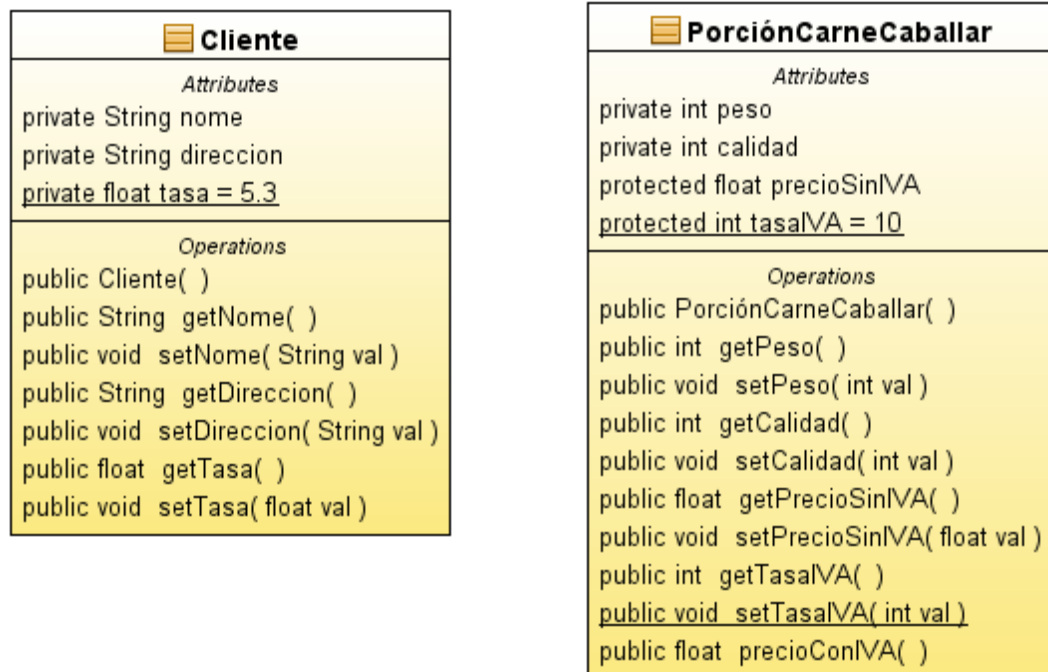
O *nome* da clase escríbese normalmente coa primeira letra de cada palabra en maiúsculas e está formado por un substantivo singular seguido dun ou varios adxectivos que o cualifican sen espazos no medio. Graficamente colócase na primeira fila do rectángulo. Se se quere facer referencia ao paquete ao que pertence a clase, farase nomeando a clase como nomepaquete::nomeclase.

O *atributo* é unha propiedade da clase. Unha clase pode ter ou non atributos. O atributo ten un nome corto que normalmente empeza por minúsculas e ten o resto das primeiras letras de cada palabra en maiúsculas sen espazos no medio. Por cada atributo pódese indicar o modificador de visibilidade (+=public, #=protected, -=private), tipo de dato (int, string, ...), nome e valor por defecto. Graficamente colócanse na segunda fila do rectángulo. Os atributos ou métodos de clase (*static*) aparecerán subliñados.

Unha *operación* ou *método* é a implementación dun servizo que pode ser requirido por calquera obxecto da clase para que se execute. Unha clase normalmente ten operacións. O nome do método normalmente é un verbo ou unha expresión verbal que empeza por minúsculas e ten o resto das primeiras letras de cada palabra en maiúsculas sen espazos no medio. O método pode ter ou non parámetros e pode ter ou non valor de retorno. Graficamente colócase a firma completa (modificador da visibilidade, tipo do retorno, nome do método, e para cada parámetro: tipo, nome e valor por defecto) na terceira fila do rectángulo.

Opcionalmente pode engadirse unha fila máis ao rectángulo para poñer a responsabilidade da clase ou texto libre e curto no que se expresan as obrigacións da clase.

Un diagrama de clases normalmente está formado por máis dunha clase relacionadas; neste caso pode ser máis claro que non aparezan tan detallados os atributos e os métodos.



Recomendacións para a elaboración dos diagramas

Nas conversacións cos clientes débese prestar atención aos substantivos que utilizan para describir o seu negocio porque normalmente deles saíran as clases; dos substantivos relacionados coas clases saíran os atributos; dos verbos saíran os métodos.

Por exemplo supoñamos a seguinte conversación entre un adestrador de baloncesto e un analista que necesita saber como funciona o xogo:

- *Analista*: "Adestrador, de que trata o xogo? "
- *Adestrador*: "Consiste en botar o balón a través dun aro, coñecido como cesto, e facer unha maior puntuación que o opoñente. Cada equipo consta de cinco xogadores: dous defensas, dous dianteiros e un central. Cada equipo leva o balón ao cesto do equipo opoñente co obxectivo de facer que o balón sexa encestrado. "
- *Analista*: "Como se fai para levar o balón ao outro cesto? "
- *Adestrador*: "Mediante pases e dribles. Pero o equipo terá que encestar antes de que remate o lapso para tirar. "
- *Analista*: "O lapso para tirar? "
- *Adestrador*: "Son 24 segundos no baloncesto profesional, 30 nun xogo internacional, e 35 no colexial para tirar o balón logo do cal un equipo toma posesión del, "
- *Analista*: "Como se puntúa? "
- *Adestrador*: "Cada canastra vale dous puntos, a menos que o tiro fose feito detrás da liña dos tres puntos. En tal caso, serán tres puntos. Un tiro libre contará como un punto. A propósito, un tiro libre é a penalización que paga un equipo por cometer unha infracción. Se un xogador realiza unha infracción sobre un opoñente, se detén o xogo e o opoñente pode realizar diversos tiros ao cesto dende a liña de tiro libre"
- *Analista*: "Fáleme máis acerca do que fai cada xogador. "
- *Adestrador*: "Os que xogan de defensa son, en xeral, os que realizan a maior parte de dribles e pases. Polo xeral teñen menor estatura que os dianteiros, e estes, á súa vez, son menos altos que o central (que tamén se coñece como 'poste'). Suponse que

todos os xogadores poden burlar, pasar, tirar e rebotar. Os dianteiros realizan a maioría dos rebotes e os disparos de mediano alcance, mentres que o central se mantén preto do cesto e dispara dende un alcance curto. "


- *Analista*: " Cales son as dimensións da cancha? E xa que estamos niso canto dura o xogo? "
- *Adestrador*: "Nun xogo internacional, a cancha mide 28 metros de lonxitude e 15 de ancho; o cesto encóntrase a 3.05 m do piso. Nun xogo profesional, o xogo dura 48 minutos, divididos en catro cuartos de 12 minutos cada un. Nun xogo internacional, a duración é de 40 minutos, divididos en dúas metades de 20 minutos. Un cronómetro do xogo leva o control do tempo restante. ".


Substantivos que apareceron: balón, cesto, equipo, xogadores, defensas, dianteiros, centro, tiro, lapso para tirar, liña de tres puntos, tiro libre, infracción, liña de tiro libre, cancha, cronómetro do xogo.


Verbos que apareceron: tirar, avanzar, driblar (o burlar), pasar, facer unha infracción, rebotar.


Información adicional respecto a algúns substantivos: estaturas relativas dos xogadores de cada posición, dimensións da cancha, cantidade total de tempo nun lapso de tiro e duración dun xogo.


Unhas clases posibles poderían ser:


 Jugador
<i>Attributes</i>
private String nombre private short estatura private short peso
<i>Operations</i>
public Jugador() public String getNombre() public void setNombre(String val) public short getEstatura() public void setEstatura(short val) public short getPeso() public void setPeso(short val) public void driblarBalon() public void pasarBalon() public void tirarBalon() public void rebotar() public void infraccionarOponente()

 Defensa
<i>Attributes</i>
<i>Operations</i>
public Defensa()

 Equipo
<i>Attributes</i>
<i>Operations</i>
public Equipo()


 Delantero
<i>Attributes</i>
<i>Operations</i>
public Delantero()

 Centro
<i>Attributes</i>
<i>Operations</i>
public Centro()

 Tiro
<i>Attributes</i>
<i>Operations</i>
public Tiro()

 TiroLibre
<i>Attributes</i>
<i>Operations</i>
public TiroLibre()


 LineaTiroLibre
<i>Attributes</i>
<i>Operations</i>
public LineaTiroLibre()


 Linea3Puntos
<i>Attributes</i>
<i>Operations</i>
public Linea3Puntos()


 LapsoDeTiro
<i>Attributes</i>
<i>Operations</i>
public LapsoDeTiro()


 Balon
<i>Attributes</i>
private short diametro private short volumen
<i>Operations</i>
public Balon() public short getDiametro() public void setDiametro(short val) public short getVolumen() public void setVolumen(short val) public void driblar() public void tirar() public void pasar() public void avanzar()

 Cesto
<i>Attributes</i>
<i>Operations</i>
public Cesto()

 Cancha
<i>Attributes</i>
<i>Operations</i>
public Cancha()

 CronometroDeJuego
<i>Attributes</i>
<i>Operations</i>
public CronometroDeJuego()

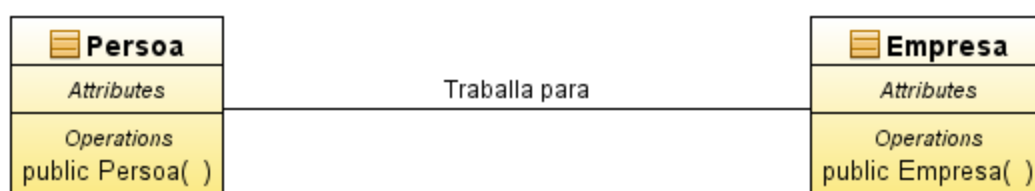
 Duracion
<i>Attributes</i>
<i>Operations</i>
public Duracion()

 Infraccion
<i>Attributes</i>
<i>Operations</i>
public Infraccion()

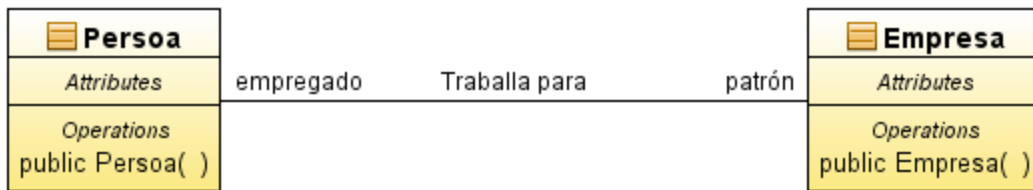
Asociación

Unha *asociación* é unha relación entre clases que especifica que os obxectos dunha clase están conectados cos doutra. Gráficamente represéntase como una liña que une as clases e soe ter nome que se coloca riba da liña que une as dúas clases.

As asociación son normalmente binarias (entre dúas clases). Cando se establece unha asociación, pódese navegar desde un obxecto dunha clase ata un obxecto da outra e vice-versa. Si a liña remata cunha punta de frecha nun dos extremos dise que a asociación ten *navegabilidade*, isto quere dicir que e posible ir dun obxecto a outro no sentido que indica a frecha pero non ó contrario.



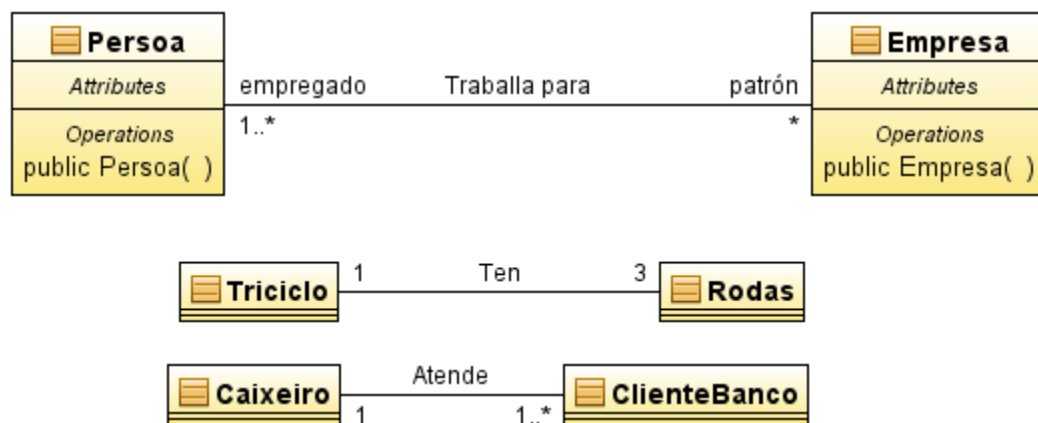
Un *rol* é a cara que unha clase presenta á outra clase da asociación. Pode nomearse explicitamente riba da liña da asociación e a carón da clase correspondente.



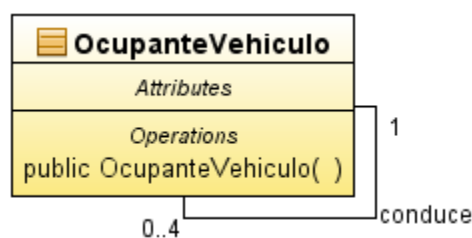
A *multiplicidade* ou *cardinalidade* da asociación é o número de obxectos dunha clase que poden relacionarse cun obxecto da outra clase e colócase na liña de asociación e a carón da clase correspondente. A multiplicidade pode ser: 1, cero ou un: 0..1, cero ou máis: = 0..* (tamén se pode poñer só *), un ou máis: 1..*, un número exacto, ou expresións máis complexas como: 0..1,3..4,6..* (calquera número agás o 2 e o 5).

Cando se pon o símbolo da multiplicidade nun extremo da asociación indica que para cada obxecto da clase do extremo oposto pode haber os obxectos desa clase que se indican na multiplicidade.

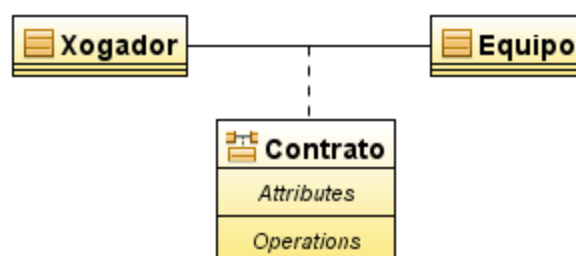
Exemplos:



A asociación reflexiva dáse cando os extremos dunha asociación están conectados á mesma clase.



Unha asociación pode ser tamén unha clase e por tanto conter atributos e operacións. Neste caso utilízase unha liña descontinua para enlazar a clase de asociación coa propia asociación.



Tamén existe a posibilidade de incluír *restricións*, é dicir, que a asociación teña que seguir algunha regra. Esta restrición pode expresarse mediante una expresión entre

chaves, unha decisión {Or} ou mediante unha linguaxe denominada OCL (Linguaxe de restrición de obxectos). Ver máis sobre OCL en <https://www.omg.org/spec/OCL/2.0/About-OCL/>

A asociación pode especializarse e entón transformarse en dependencia, xeneralización, composición forte (composición) ou composición débil (agregación).

Xeneralización

A *xeneralización* é unha especialización da asociación na que intervén a herdanza.

Terminoloxía relacionada coa herdanza entre clases:

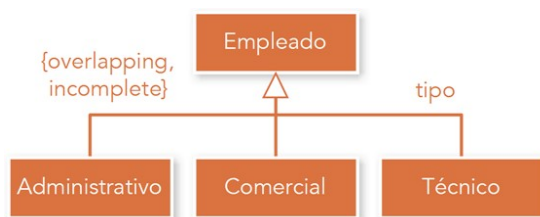
- Unha clase sen pai e con clases filla denomínase clase raíz ou clase base.
- Unha clase sen fillas chámase clase folla.
- Unha clase cunha única clase pai dise que ten herdanza simple.
- Unha clase con máis dun pai dise que ten herdanza múltiple.

Unha *xeneralización* é unha relación entre unha clase xeneral chamada súper clase ou pai e unha clase máis específica chamada subclase ou filla. A clase filla herda os atributos e operacións da clase pai, pode engadir atributos e operacións aos que herda e pode redefinir operacións herdadas (polimorfismo).

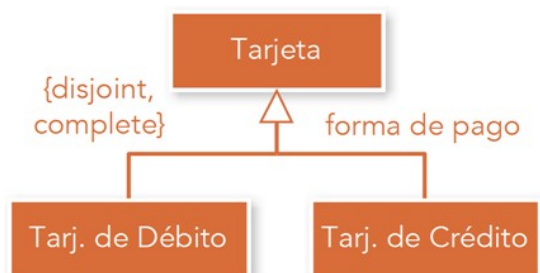
Unha xeneralización represéntase en UML como unha liña continua coa punta da frecha sen recheo apuntando ao pai.

Na xeneralización pode especificarse máis a relación entre a súper clase e as subclases indicando mediante estereotipos estándar:

- Cobertura: Indica se é obrigatorio que tódalas instancias dunha clase pai se clasifiquen nalgunha das clases fillas:
 - ✓ Complete: Cada instancia da superclase é obrigatoriamente tamén instancia dalgunha ou varias subclases.
 - ✓ Incomplete: Pode haber instancias da superclase que non sexan instancias de ningunha subclase.
- Solapamento: Se unha instancia pode pertencer á vez a diferentes subclases:
 - ✓ Disjoint: Unha instancia da clase pai só pode ser dunha única clase filla
 - ✓ Overlapping. Permítese que unha instancia pertenza a varias subclases.

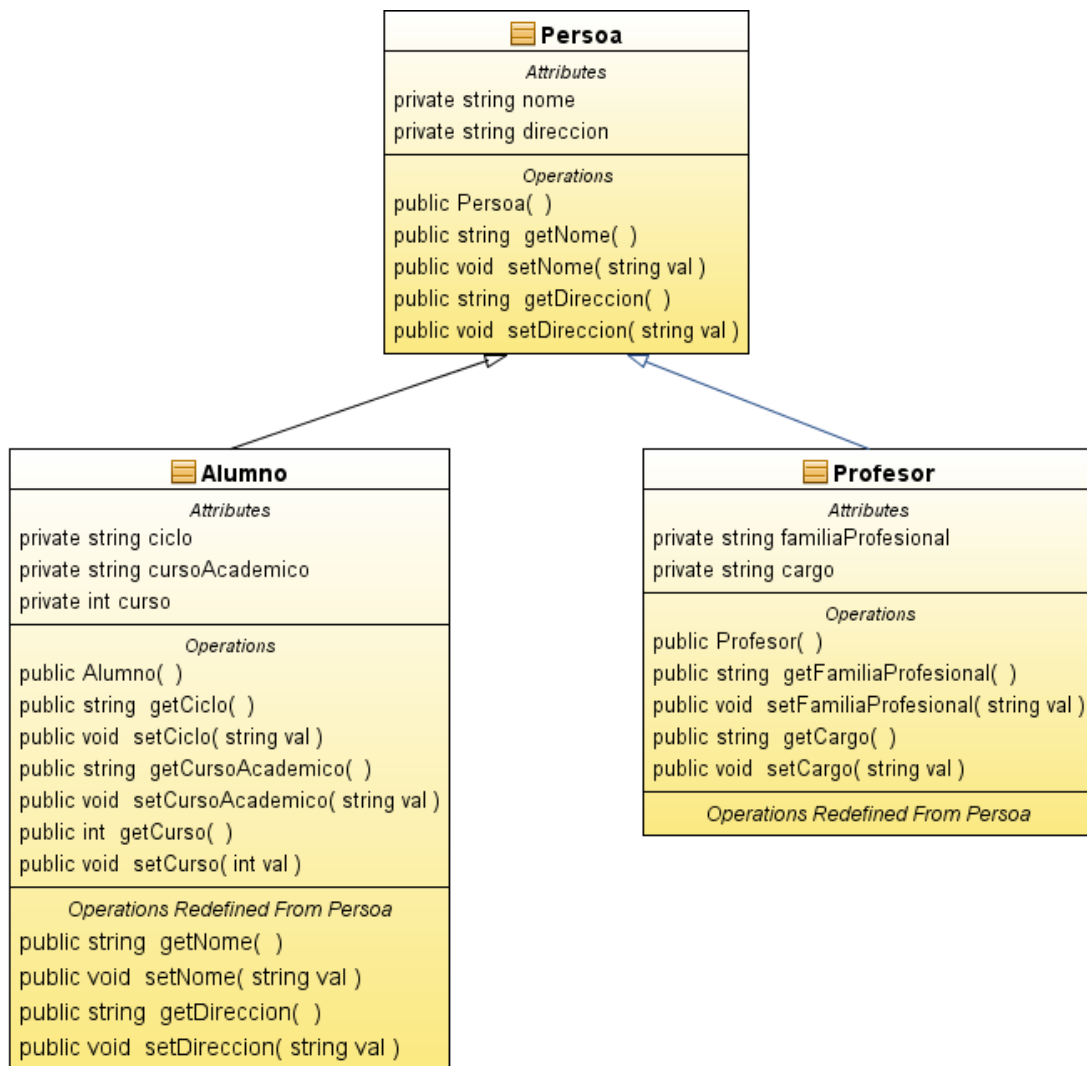


Neste caso un empregado podería ser Técnico e Administrativo á vez. Tamén podería non ser de ningunha categoría e pertencer directamente á clase Empleado



Neste outro caso as tarxetas só poden ser de dous tipos: crédito ou débito. Unha tarxeta de débito non pode ser nunca de crédito e viceversa

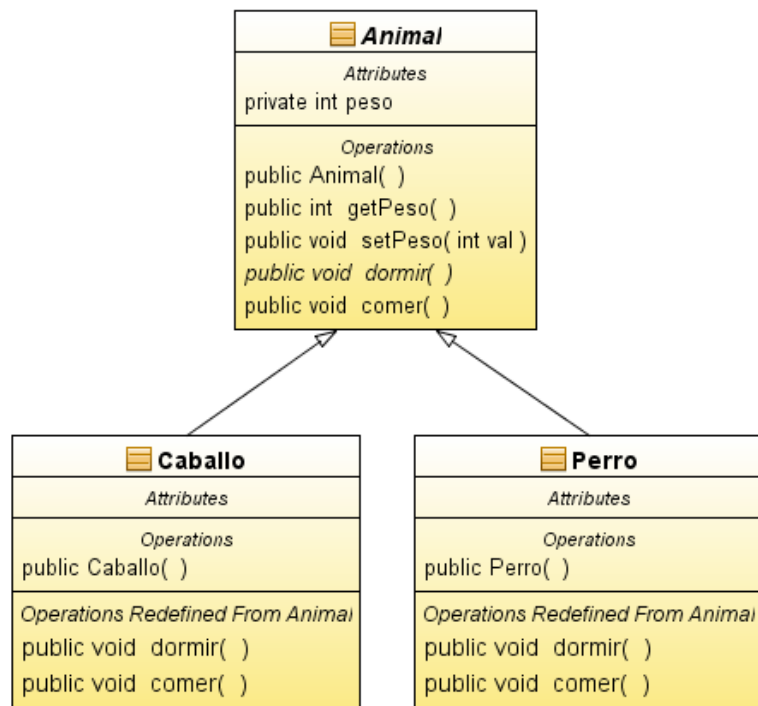
Se algunha subclase redefine métodos da clase pai, pode indicarse na subclase cales son os métodos que se redefinen.



Clases abstractas

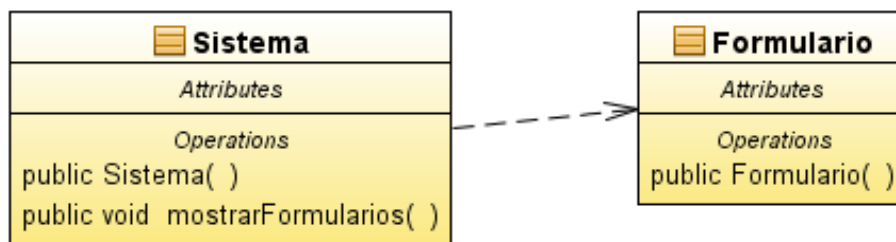
Unha clase *abstracta* é aquela que non ten implementados tódolos métodos e por tanto non pode ser instanciada. Pode ter métodos abstractos (só a firma e non o contido) ou non. A súa finalidade é posuír subclases concretas que poden ser instanciadas e nas que se definirán os métodos abstractos da súper clase. Utilízanse cando nunha xerarquía de clases algún comportamento está presente en todas elas pero se materializa de forma distinta para cada unha.

As clases e métodos abstractos represéntanse co nome en cursiva e está dentro dunha clase pai dunha xeneralización.



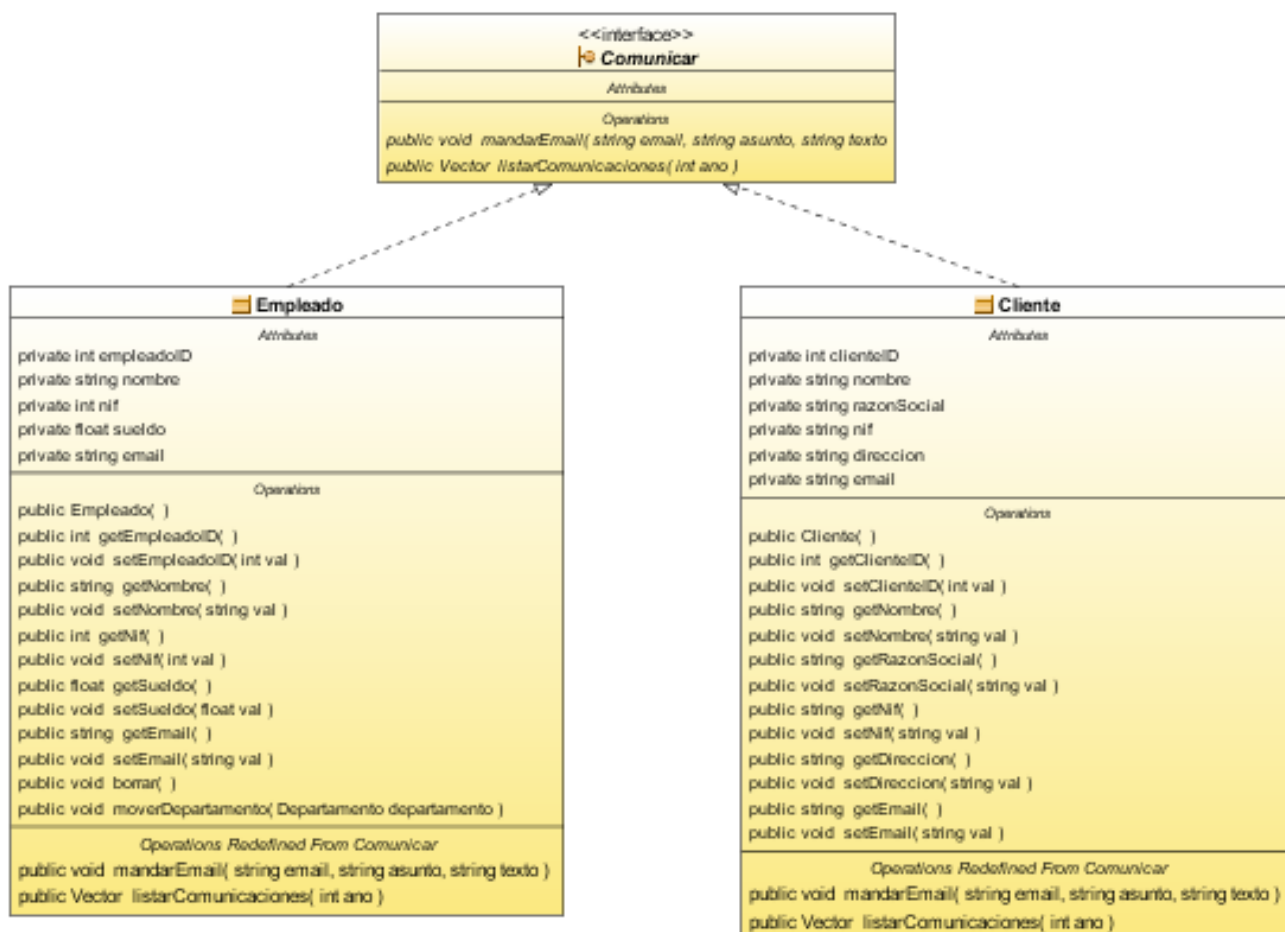
Dependencia

Unha *dependencia* é unha asociación na que se indica que unha clase necesita doutra para o seu cometido. Representase cunha frecha descontinua que vai dende a clase dependente á clase utilizada. As dependencias indican que un cambio na clase utilizada pode afectar ao funcionamento da clase dependente pero non ao contrario.



Interface

Unha **interface** pode ser definida como unha clase abstracta pura, é dicir, declara a forma dunha clase e por tanto só define métodos abstractos e atributos constantes que logo serán implementados en clases. A interface represéntase cunha icona específica e o estereotipo `<<interface>>` riba do nome e as clases que implementan esa interface serán dependentes da interface.



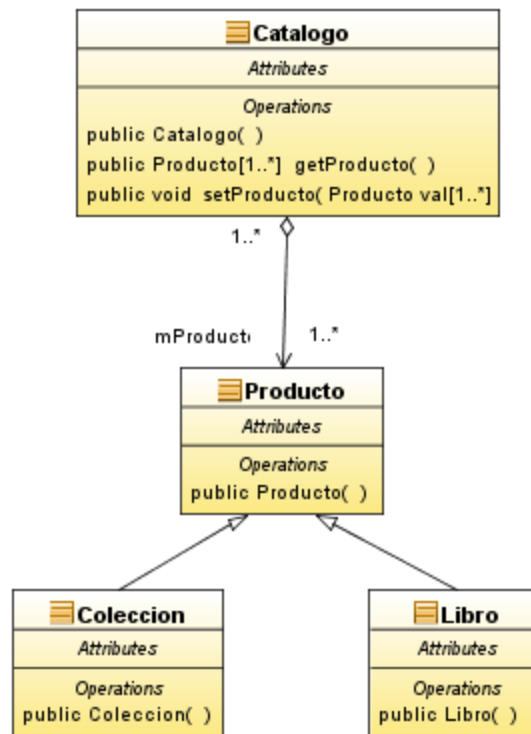
Composición débil ou Agregación

É unha asociación entre clases que indica que unha clase A está composta pola clase B de tal forma que os compoñentes poden ser compartidos por outros elementos compostos. Como consecuencia, a supresión do obxecto da clase A non implica a supresión do obxecto agregado da clase B.

Represéntase graficamente cunha liña continua cun rombo sen recheo na clase A.

A agregación pode levar navegabilidade para indicar que a propiedade agregada se inclúe automaticamente na clase A; en caso contrario, habería que definir explicitamente esa propiedade.

Exemplo: unha librería na que os produtos se ofrecen agrupados en catálogos. Cada catálogo está formado por un ou máis produtos. Os produtos poden ser libros ou coleccións. Agrégase a clase Produto á clase Catalogo e se a agregación é con navegabilidade agréganse os métodos get e set na clase Catalogo.



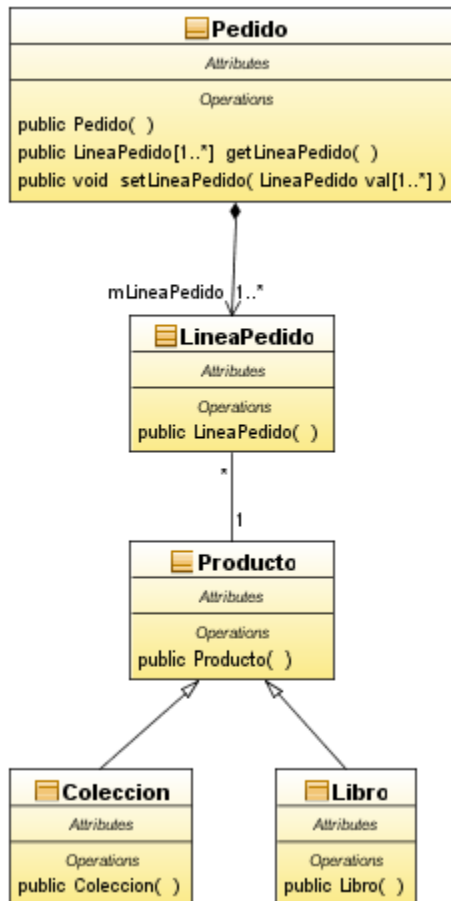
Composición

É unha asociación entre clases que especifica que unha clase A está composta pola clase B de tal forma que os obxectos de B teñan a mesma vida que os obxectos de A e non poden ser compartidos por outros obxectos compostos. Como consecuencia, a supresión do obxecto da clase A implica a supresión do obxecto da clase B.

Represéntase graficamente cunha liña continua cun rombo con recheo na clase A.

A composición pode levar navegabilidade para indicar que a propiedade compoñente se inclúe automaticamente na clase A; en caso contrario, habería que definir explicitamente esa propiedade.

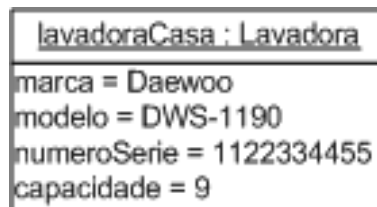
Exemplo: pedidos compostos cada un por unha ou máis liñas de pedido (unha para cada produto).



Diagramas de obxectos

Permite representar unha instancia dunha clase. O obxecto está identificado polo nome da clase seguido de dous puntos e o nome do obxecto e todo iso subliñado. O nome do obxecto iníciase con letra minúscula. Os atributos aparecen normalmente con valores.

Por exemplo o diagrama do obxecto `lavadoraCasa` da clase `Lavadora`:



Os obxectos de clases relacionadas mediante unha asociación tamén están asociadas e neste caso o nome da asociación tamén aparece subliñado.

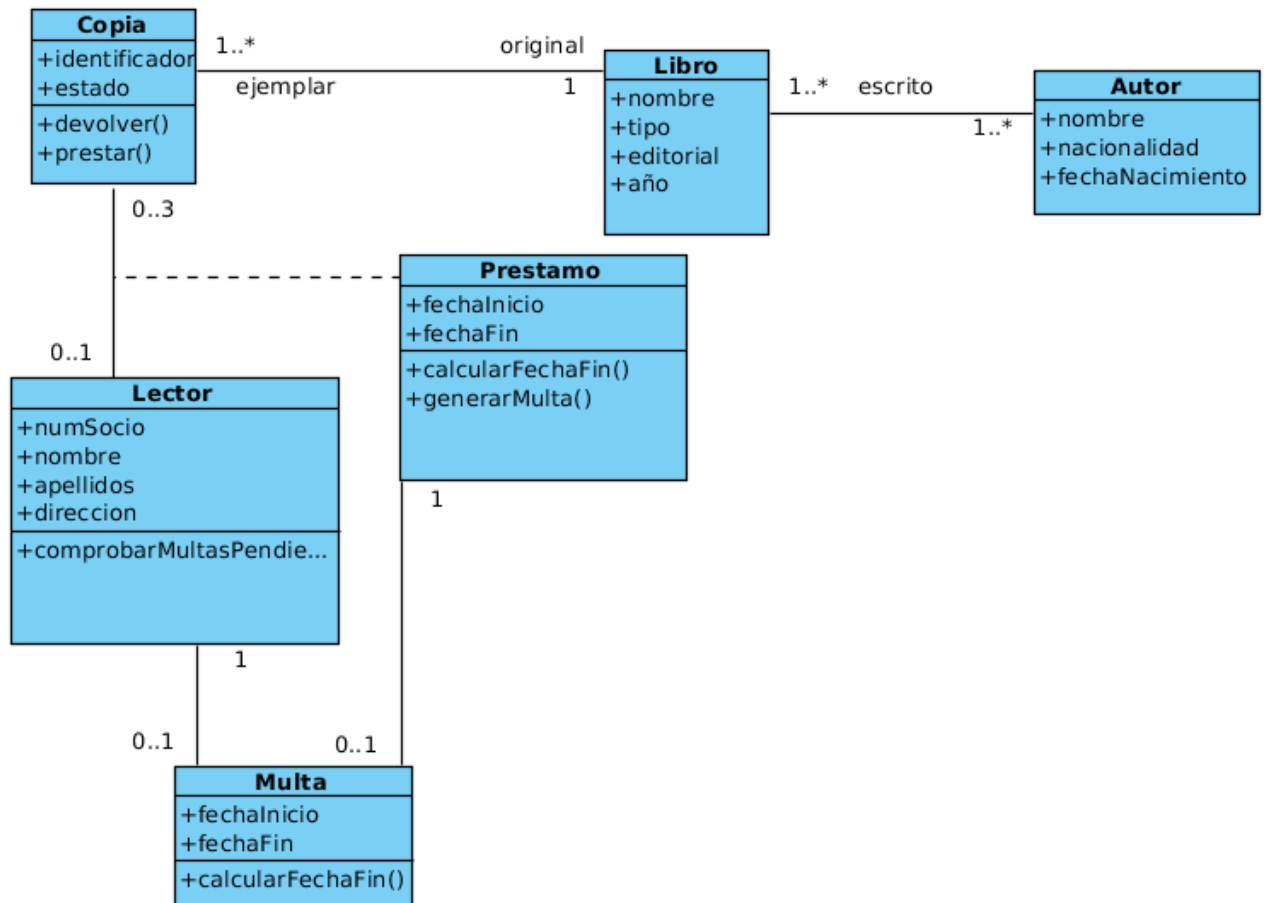
1.3 Tarefas

As tarefas propostas son as seguintes.

- **Tarefa 1.** *Interpretación dun diagrama de clases.* Nesta tarefa imos interpretar a información mostrada nun diagrama de clases dado.
- **Tarefa 2, 3, 4, 5 e 6.** *Elaboración dun diagrama de clases.* Nestas tarefas imos a elaborar diagramas de clases para distintos escenarios dados.

1.3.1 Tarefa 1. Interpretación dun diagrama de clases

Dado o seguinte diagrama de clases, redacta un posible escenario que se adapte á información recollida no diagrama.



1.3.2 Tarefa 2. Elaboración dun diagrama de clases

Representa mediante diagramas de clase os vínculos de herdanza entre os periféricos nos dous seguintes casos:

- Diagrama periférico1: Os periféricos poden ser extraíbles e non extraíbles.
- Diagrama periférico2: Os disquetes, memorias USB e os discos duros son periféricos.

1.3.3 Tarefa 3. Elaboración dun diagrama de clases

Representa mediante diagramas de clase a seguinte estrutura:

- Unha superclase ObxectoGráfico que ten como atributos protexidos: grosorTrazo, ordenada, abscisa, color e como operacións públicas: mover(), visualizar() e xirar().
- Tres subclases Punto, Círculo (ten atributo privado diámetro) e Cadrado (ten atributo privado lado).Cada unha destas clases ten unha maneira diferente de moverse, visualizarse e xirar.
- Contesta as seguintes preguntas:
 - ¿É accesible diámetro dende Cadrado?.
 - ¿Un obxecto Círculo posúe un atributo color?.
 - ¿Pode aplicarse o método mover a un obxecto Punto?.
 - ¿Que interese pode ter que a clase ObxectoGráfico sexa abstracta?. Modifica o diagrama de clases para que a clase ObxectoGráfico sexa abstracta.

Estende o modelo anterior para que un novo obxecto gráfico chamado GráficoComposto estea composto de varios obxectos gráficos, de tal forma que a supresión do obxecto GráficoComposto implique a supresión dos ObxectoGráfico que o compoñen.

1.3.4 Tarefa 4. Elaboración dun diagrama de clases

Representa a través dun diagrama de clases o seguinte escenario:

- Necesitamos desenvolver unha aplicación para xestionar a información de diferentes empresas, dos seus clientes e dos seus empregados.
- Tanto de clientes coma de empregados almacenaremos o seu nome e a súa idade. Da empresa, o seu nome.
- Dos empregados almacenaremos o seu salario bruto e a aplicación terá que calcular o salario neto. Os empregados que son directivos teñen unha categoría así como un conxunto de empregados subordinados.
- Dos clientes ademais necesítase coñecer o seu teléfono de contacto
- A aplicación debe de poder mostrar os datos de empregados, clientes e empresas así como o listado de empregados subordinados dun directivo.
- Representa tamén no diagrama os métodos para implementar as *Propiedades* de acceso ós atributos dos obxectos (métodos `get` e `set`)

1.3.5 Tarefa 5. Elaboración dun diagrama de clases

Representa a través dun diagrama de clases o seguinte escenario, especificando os tipos de datos que consideres apropiados para os atributos, incluíndo os métodos `set` e `get` para estes e os construtores das clases:

- Deséxase almacenar a información de distintos tipos de obras culturais. As devanditas obras poden ser libros, discos ou películas.
- De toda obra nos interesa almacenar o seu título e ano de edición.
- Dos libros interésanos almacenar o seu editorial e o seu número de páxinas.
- Dos discos interésanos almacenar a súa discográfica e o número de cancións.
- Das películas interésanos almacenar a súa produtora.
- Ademais interésanos almacenar os distintos artistas (autores) que crearon a obra. Así mesmo, no caso das películas interésannos os artistas (actores) que a interpretan.
- Dos artistas interésanos o seu nome e ano de nacemento.
- A aplicación debe de ser capaz de mostrar por pantalla a información de obras e artistas.

1.3.6 Tarefa 6. Elaboración dun diagrama de clases

Representa mediante un diagrama de clase a xestión dunha conferencia científica coas consideracións seguintes:

- A conferencia pode ter varias sesións.
- Unha sesión posúe fecha e hora de inicio, pertence só a unha conferencia e non ten razón de ser sen unha conferencia.
- Os participantes nunha sesión poden ser oradores ou público. Todos eles teñen que inscribirse na conferencia. Pode cancelarse ou confirmarse unha inscrición.
- Un ou máis artigos científicos preséntanse nunha sesión. Cada artigo pode ser curto ou longo e trata dun tema determinado.
- Un autor pode ter un ou varios artigos presentados na conferencia.
- Coloca os atributos que che parezan convenientes.