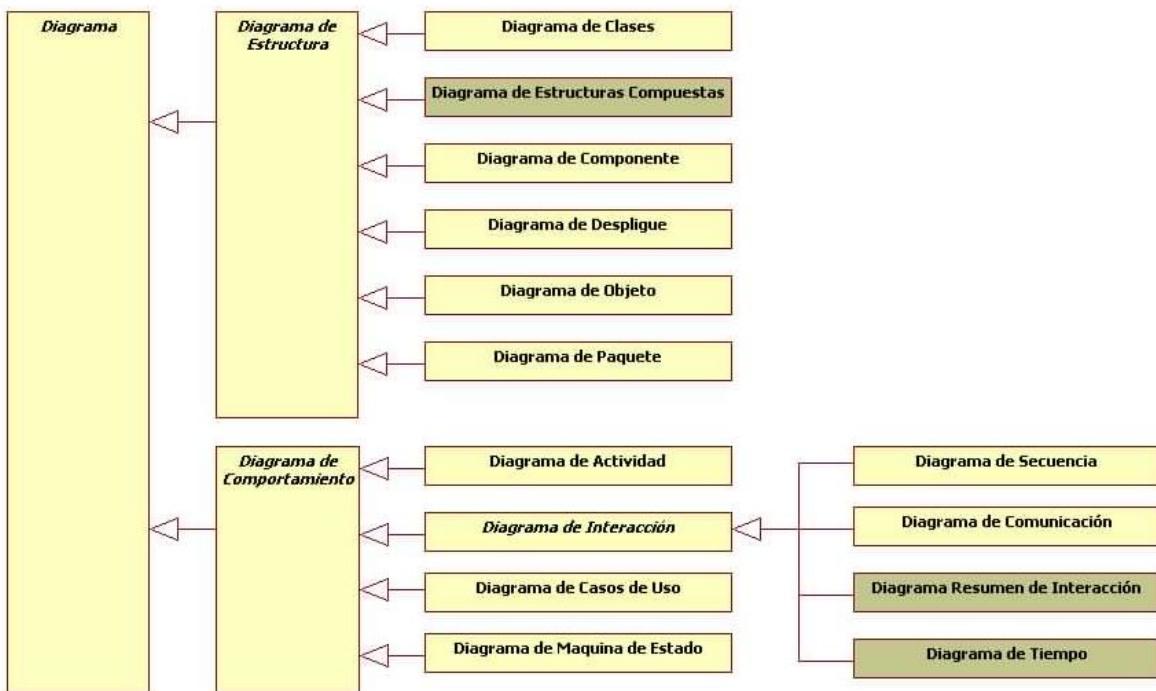


1. Diagramas de clases

1.1 Introdución aos diagramas de clases

Os diagramas estruturais de UML

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



Os **diagramas estruturais** presentan os 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 estes 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 compresión do contorno ao que 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 obxectos, como Eiffel, Java ou C++. Para todos estes usos, o que se desea é expresar as unidades en que o código se organiza -as clases- así como algunas 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 valido 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 descripción 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::nomoclase.

O **atributo** é unha propiedade da clase. Unha clase pode ter ou non atributos. O atributo ten un nome curto que normalmente comeza 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.

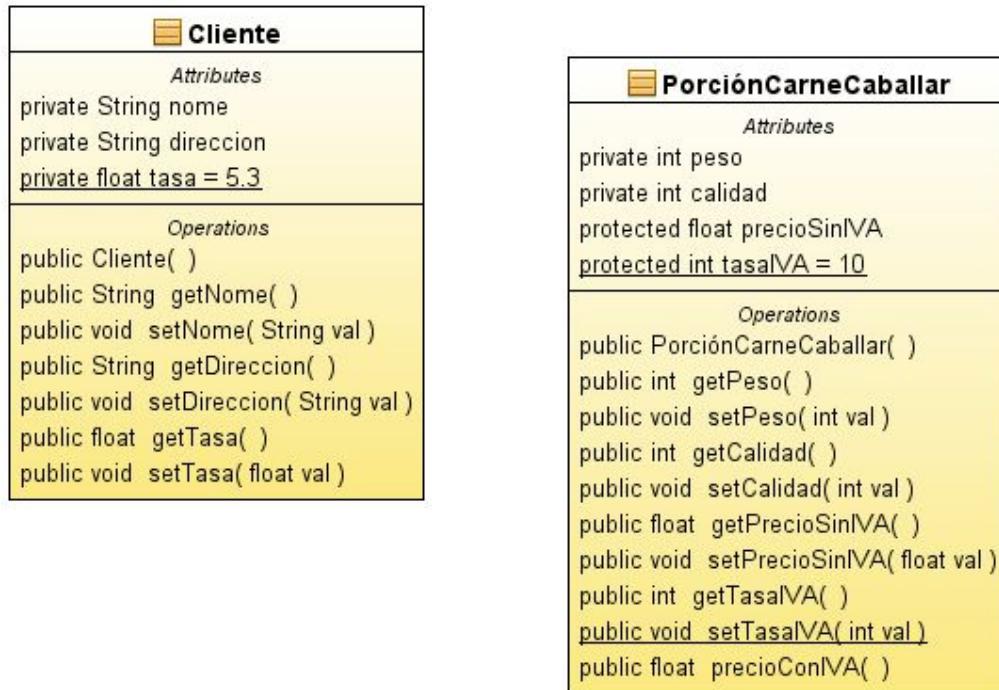
Os **atributos** ou **métodos de clase** (static) aparecerán subliñados.

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 sinatura 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 obligacións da clase.

Un diagrama de clases normalmente está formado por máis dunha clase relacionada; neste caso pode ser más 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 sairán as clases; dos substantivos relacionados coas clases sairán os atributos; dos verbos sairán 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 oponente. Cada equipo consta de cinco xogadores: dous defensas, dous dianteiros e un central. Cada equipo leva o balón ao cesto do equipo oponente co obxectivo de facer que o balón sexa encestado."
- *Analista*: "Como se fai para levar o balón ao outro cesto?"

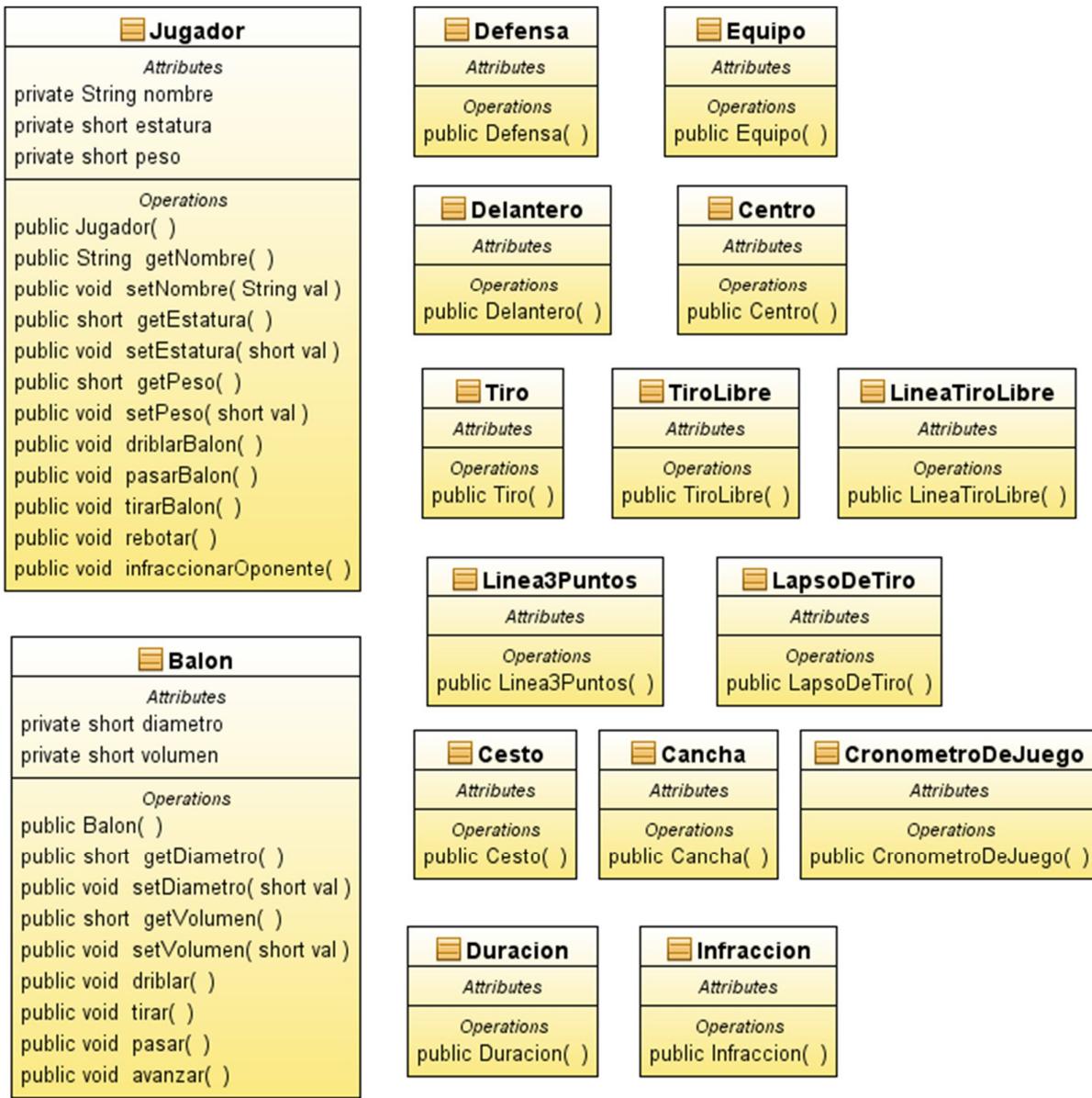
- *Adestrador:* "Mediante pases e dribles. Pero o equipo terá que encestar antes de que remate o lapso para tirar."
- *Analista:* "O lapso para tirar?"
- *Adestrador:* "Son 24 segundos no baloncesto profesional, 30 nun xogo internacional, e 35 no colexial para tirar o balón logo do cal un equipo toma 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 oponente, detense o xogo e o oponente 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'). Supонse 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 dimensíons 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 (ou burlar), pasar, facer unha infracción, rebotar.

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

Unhas clases posibles poderían ser:



Asociación

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

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



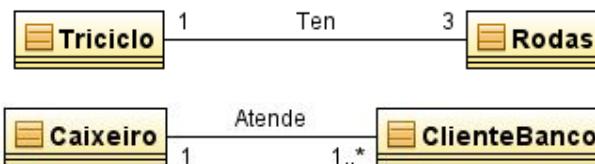
Un **rol** é a cara que unha clase presenta á outra clase da asociación. Pode nomearse explicitamenteriba da liña da asociación e a carón da clase correspondente.



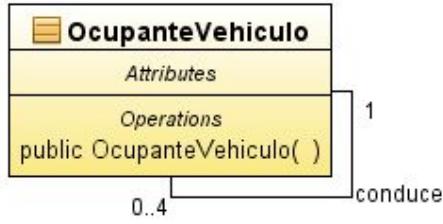
A **multiplicidade** ou **cardinalidade** da asociación é o número de obxectos dunha clase que poden relacionarse cun obxecto da outra clase e colócase na liña de asociación e a carón da clase correspondente. A multiplicidade pode ser: 1, cero ou un: 0..1, cero ou máis: = 0..* (tamén se pode poñer só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

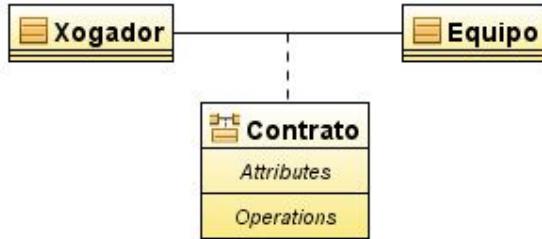
Exemplos:



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 **restriccións**, 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).

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:

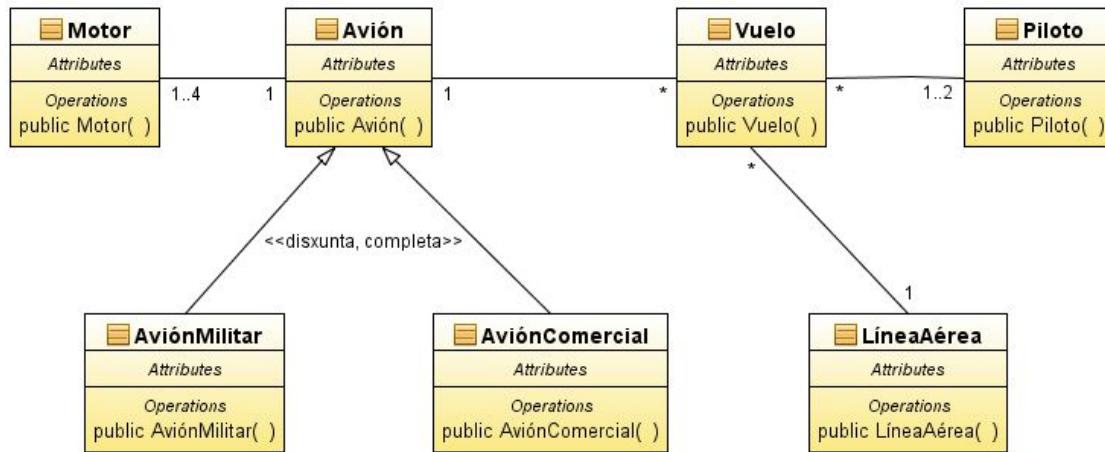
- Unha clase sen pai e con clases filla denominase **clase raíz** ou clase base.
- Unha clase sen fillas chámase **clase folla**.
- Unha clase cunha única clase pai dise que ten **herdanza simple**.
- Unha clase con máis dun pai dise que ten **herdanza múltiple**.

Unha **xeneralización** é unha relación entre unha clase xeneral chamada superclase ou pai e unha clase más 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).

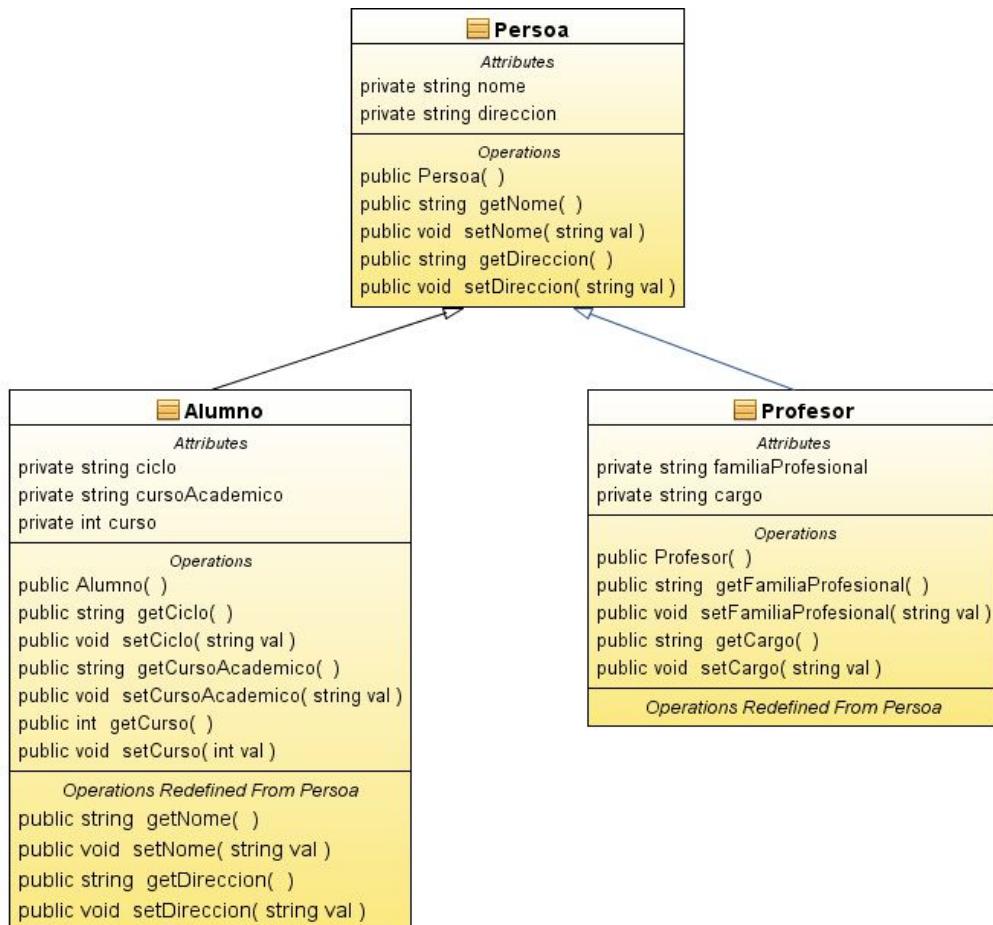
Unha xeneralización represéntase en UML como unha liña continua coa punta da frecha sen recheo apuntando ao pai.

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

- Que o grupo de subclases complete (complete) ou non (incomplete) a superclase.
- Que as subclases non teñan nada en común (disjoint) ou teñan elementos comúns (overlapping).



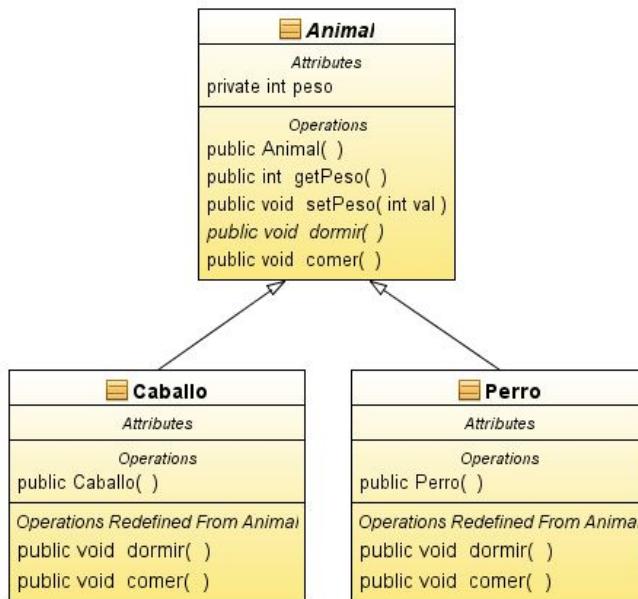
Se algunha subclase redefine métodos da clase pai, pode indicarse na subclase cales son os métodos que se redefinen.



Clases abstractas

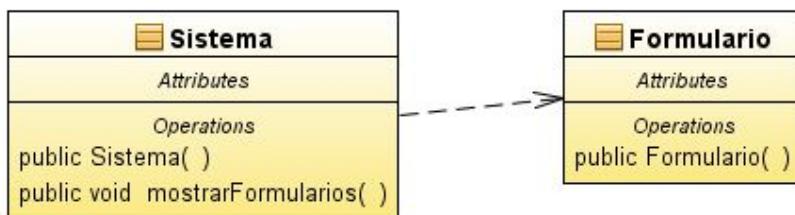
Unha clase **abstracta** é aquela que non ten implementados todos os métodos e, por tanto, non pode ser instanciada. Pode ter métodos abstractos (só a firma e non o contido) ou non. A súa finalidade é posuír subclases concretas que poden ser instanciadas e nas que se definirán os métodos abstractos da superclase. 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 están 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. Represéntase cunha frecha descontinua que vai dende a clase dependente á clase utilizada. As dependencias indican que un cambio na clase utilizada pode afectar ao funcionamiento 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>> enriba do nome e as clases que implementan esa interface serán dependentes da interface.



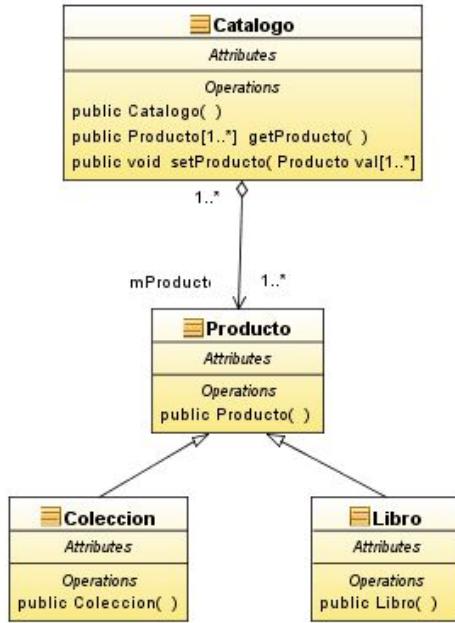
Composición débil ou Agregación

É unha asociación entre clases que indica que unha clase A está composta pola clase B de tal forma que os componentes 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.

Representase 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 **Producto** á clase **Catalogo** e se a agregación é con navegabilidade agregaranse os métodos `get` e `set` na clase **Catalogo**.

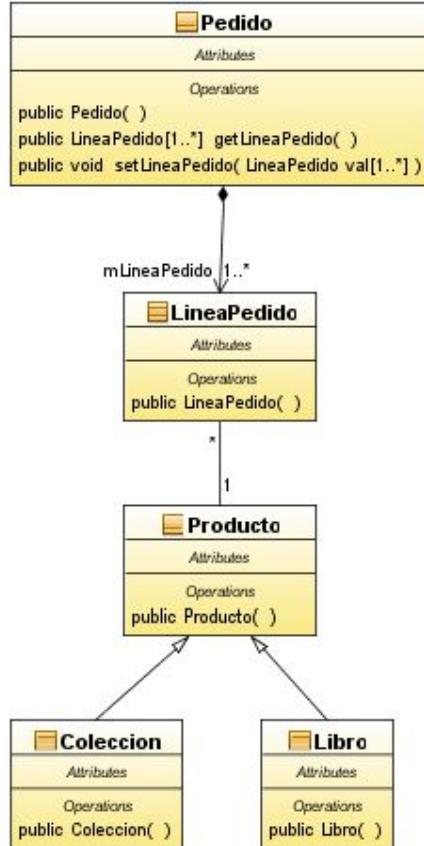


Composición

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.

Représéntase 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.

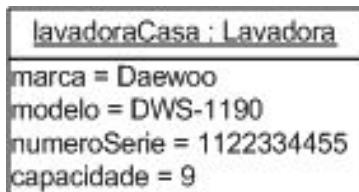
Exemplo: pedidos compostos cada un por unha ou máis liñas de pedido (unha para cada produto).



Diagramas de obxectos

Permiten 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.

2. Ferramentas para o traballo con diagramas de clases

2.1 Introdución ás ferramentas software UML

Aínda que o traballo con diagramas UML é perfectamente posible sen o emprego de ferramentas software, o máis habitual é emplegar aplicacións que nos faciliten a representación dos diagramas e que incorporen a maiores unha serie de funcionalidades para o traballo con eles como poden ser a **xeración automática de código** a partir dos propios diagramas ou incluso a posibilidade de facer “**enxeñaría inversa**” a partir do noso código, é dicir, xerar diagramas UML a partir de código fonte do noso proxecto.

Criterios para a selección dunha ferramenta UML

Aspectos a ter en conta para elixir unha ferramenta que permita elaborar e traballar con modelos UML:

- Licenza e prezo.
- Plataforma sobre a que traballa.
- Tipo de diagramas que permite.
- Linguaxes de programación que permita asociar ao modelo e, por tanto, que poida xerar código a partir de diagramas.
- Posibilidade de xerar automaticamente documentación.
- Posibilidade de enxeñaría inversa, é dicir, xerar diagramas a partir de código.
- Facilidade de navegación no modelo.
- Posibilidades de exportación do modelo.
- Interface de comunicación co usuario.

Existen centos de ferramentas UML independentes con licenza propietaria ou libre e moitos contornos de desenvolvemento que poden utilizar ferramentas UML.

Comparativas

Relación de sitios web con comparativas entre ferramentas UML:

http://en.wikipedia.org/wiki/Comparison_of_Unified_Modeling_Language_tools

<https://socialcompare.com/en/comparison/uml-tools>

Algunhas ferramentas Open Source



Dia (<http://dia-installer.de/index.html.en>)

Umbrello (<http://uml.sourceforge.net/>)



Modelio (<http://www.modelio.org/>)

easyUML é un plugin para Netbeans que permite crear diagramas UML de forma visual.



Aínda que foi desenvolvido para a versión 8 de NetBeans, segue funcionando nas versión



actuais do contorno de desenvolvemento.



UMLet é unha ferramenta multiplataforma que inclúe extensión para Eclipse e VS Code.

Dispón de versión web **UMLetino** (<https://www.umletino.com/umletino.html>) que non precisa instalación.

Ferramentas gratuítas



JDeveloper é un contorno de desenvolvemento integrado desenvolvido por Oracle Corporation para as linguaxes Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML e outros. É un software propietario pero gratuito dende 2005.

Algunhas ferramentas propietarias



Visual Paradigm (<http://www.visual-paradigm.com/>) é una potentísima ferramenta que nos permite traballar con proxectos UML. Permite a integración con NetBeans. Dispón dunha edición gratuita (Community Edition) e una versión de proba de 30 días. Existe, tamén, un editor de diagramas online (<https://online.visual-paradigm.com/es/diagrams/>).

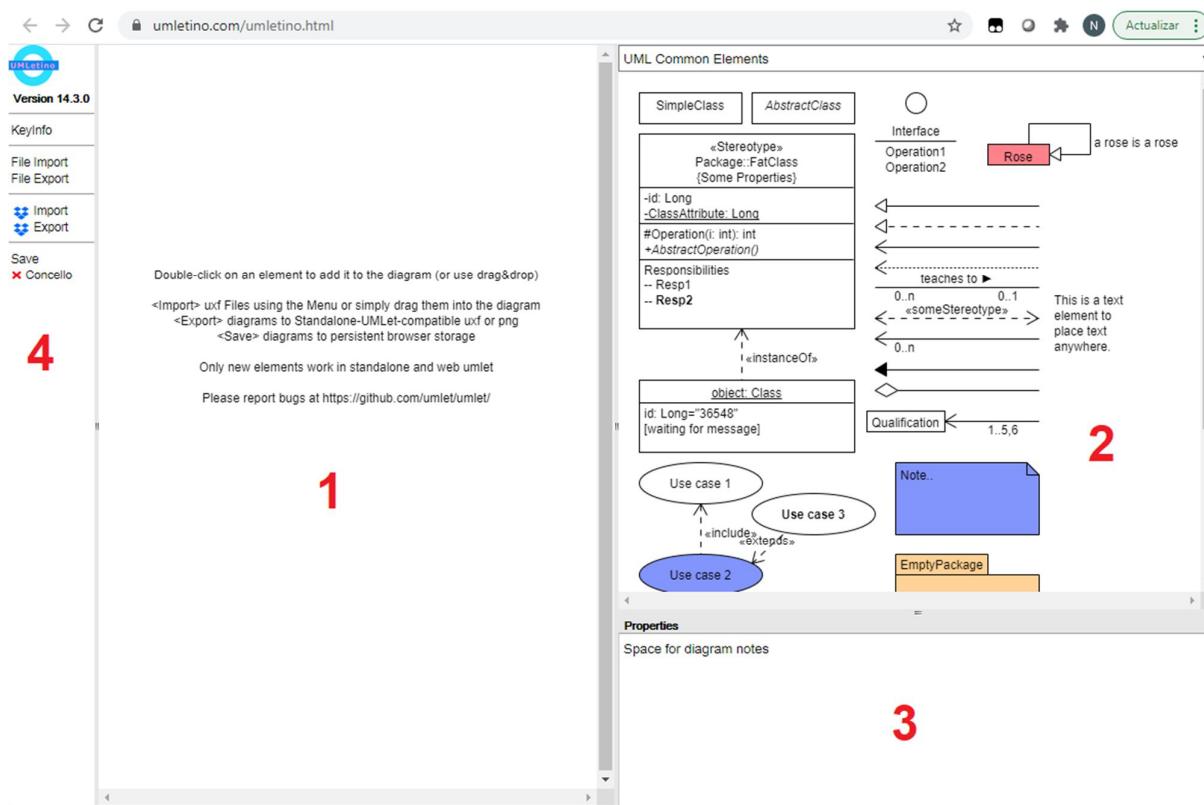
UModel (<http://www.altova.com/es/umodel.html>) defínese como unha ferramenta UML para o modelado de software e desenvolvemento de aplicacións. Pertence á compañía Altova que desenvolve aplicacións para traballo con XML, bases de datos e UML. Dispón dunha versión de proba de 30 días.

UMLetino

Podemos utilizar esta ferramenta web na URL <https://www.umletino.com/umletino.html>.

UMLetino é un editor de diagramas moi simple.

A interface de UMLetino amósase na seguinte imaxe:



1. **Rexión do editor.** Nesta rexión é onde estará o diagrama activo. Os compoñentes desta rexión poden agrandarse, rotarse e moverse manipulándoo co rato.
2. **Rexión de componentes.** Nesta rexión atopamos todos os componentes do diagrama.
3. **Rexión de edición de texto/axuda.** Ao pinchar nun elemento da rexión do editor, esta área amosa a información do elemento seleccionado e pode editarse (por exemplo, cambiar o nome da clase).
4. **Menú.** Os menús desta rexión serven principalmente para gardar, importar e exportar diagramas.

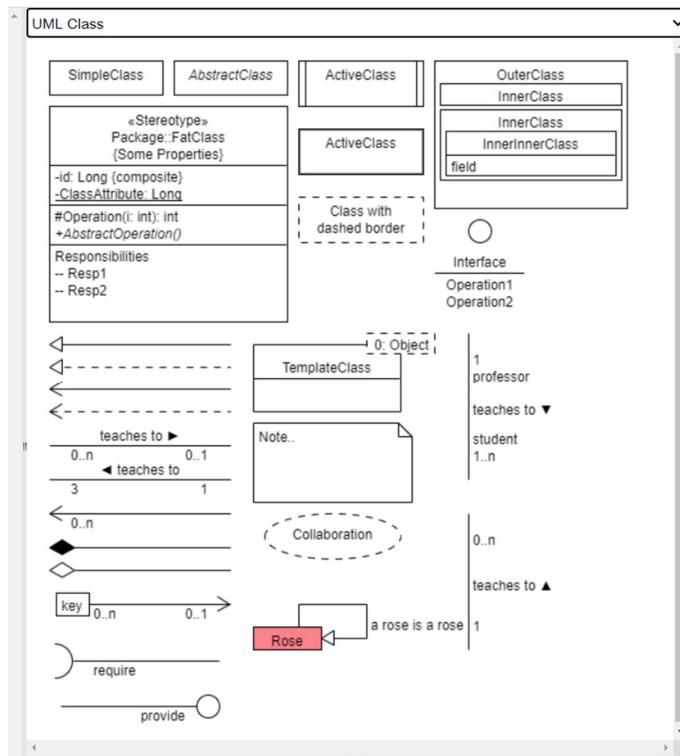
As opcións do menú que aparece na zona esquerda son:

- **Keyinfo.** Amosa os atallos de teclado.
- **File Import.** Importa un proxecto UML no formato propio de UMLet (.uxf).
- **File Export.** Exporta un proxecto UML de tipo UMLet (.uxf) ou como imaxe (.png).
- **Import.** Importa un proxecto UML no formato propio de UMLet (.uxf) desde Dropbox.
- **Export.** Exporta un proxecto UML de tipo UMLet (.uxf) a Dropbox.
- **Save.** Salva o proxecto no sistema de almacenamento persistente do propio navegador.

Os **atallos de teclado** que aparecen ao seleccionar **Keyinfo** son os que se visualizan na seguinte imaxe:

KEYBOARD SHORTCUTS	
DIAGRAM	
DELETE	delete the currently selected elements
BACKSPACE	
Ctrl+A	select all elements
Ctrl+Shift+A	deselect all elements
Ctrl+D	
Ctrl+C	copy selected elements
Ctrl+X	cut selected elements
Ctrl+V	paste cut or copied elements
Ctrl+S	save current diagram in browser storage
SHIFT	hold to disable sticking of elements
Cursor ↑	moves selected element(s) up
Cursor ↓	moves selected element(s) down
Cursor ←	moves selected element(s) left
Cursor →	moves selected element(s) right
BROWSER (only if browser supports them)	
F11	switch to fullscreen
Ctrl+PLUS	zoom diagram in
Ctrl-MINUS	zoom diagram out
Ctrl+0	reset diagram zoom
PROPERTIES PANEL	
Ctrl+SPACE	Show all autocompletion suggestions

Na rexión de compoñentes poden verse distintos símbolos propios de UML.



Para utilizar calquera elemento faremos dobre clic sobre o elemento en cuestión e arrastrarémolo á zona central.

Na parte superior da rexión de compoñentes aparece un menú despregable para seleccionar distintos tipos de diagrama tal e como se visualiza na seguinte imaxe:



Na rexión do editor amósase o diagrama que se está deseñando.



Se pulsamos en calquera dos elementos do diagrama, o elemento no que pulsamos cambiará de cor e aparecerá a súa información na zona inferior dereita.



As propiedades da clase Habitante da imaxe anterior serían:

Properties

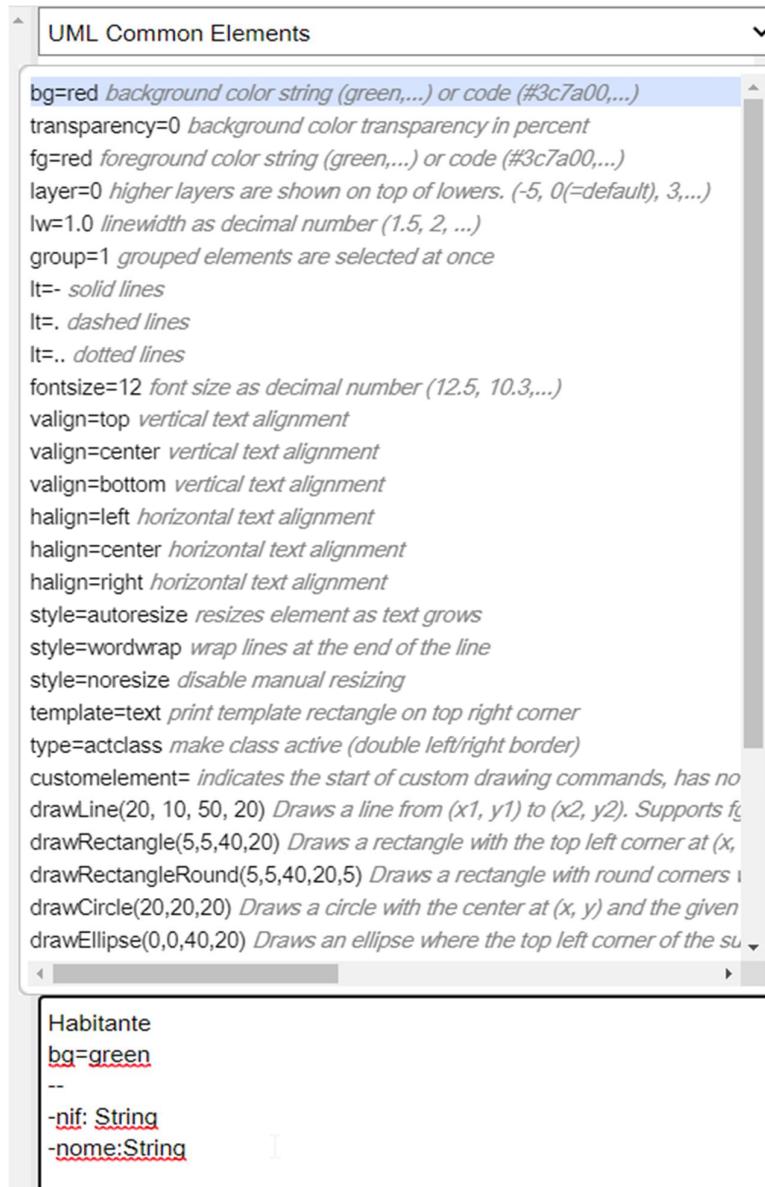
```

Habitante
bg=green
--
-nif: String
-nome:String

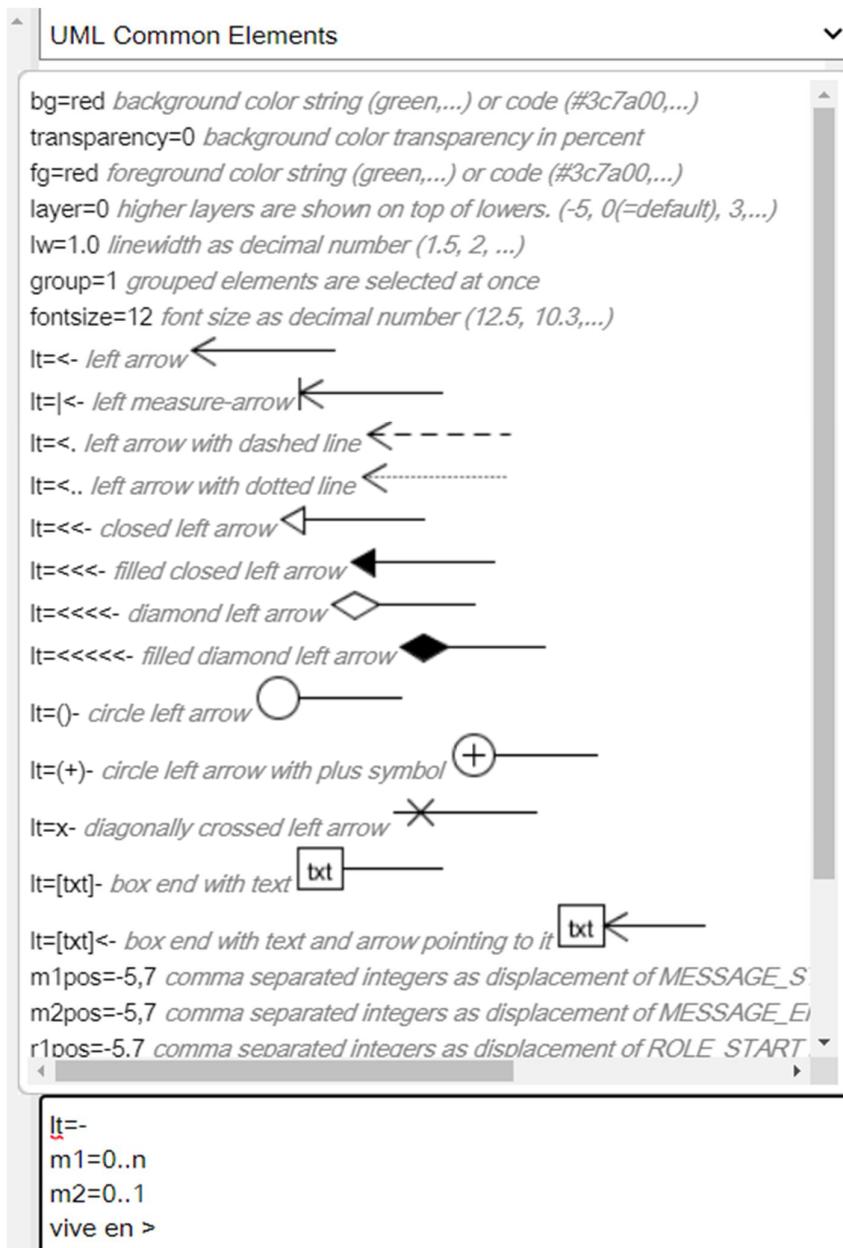
```

Para saber as opcións disponibles dependendo do elemento que esteamos editando debemos teclear Ctrl+barra de espazo nunha nova liña da rexión de edición.

No caso dunha clase, veremos as seguintes opcións:



No caso dunha relación veremos as opcións:



Plugin easyUML de Apache NetBeans

O contorno de desenvolvemento NetBeans permite instalar a extensión easyUML para o traballo con diagramas UML que, ademais de aportarnos ferramentas gráficas para a representación do diagrama de clases, vai darnos a posibilidade de xerar código a partir dos diagramas que teñamos feito e, incluso, poderemos facer “enxeñaría inversa” a partir do noso código, é dicir, xerar diagramas UML a partir do código fonte do noso proxecto.

Actualmente non está dispoñible no portal de plugins de NetBeans.

Esta extensión podemos instalala desde <https://github.com/mgeeee35/easyUML>.

En <https://github.com/ossdcfos/easyuml> temos o código fonte do proxecto.

O plugin explícase en <https://branislav1993.gitbooks.io/easyuml/content/>.

Se temos o plugin comprimido, unha vez descargado, descomprimímolo.

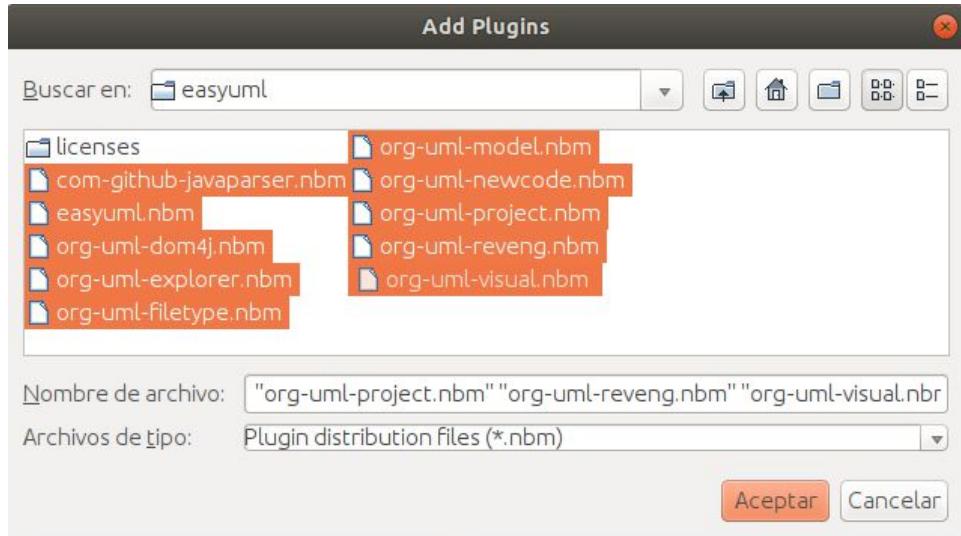
No caso dunha distribución Ubuntu e se o temos en formato zip podemos usar o comando:

```
unzip 1499813117_easyuml.zip -d easyuml
```

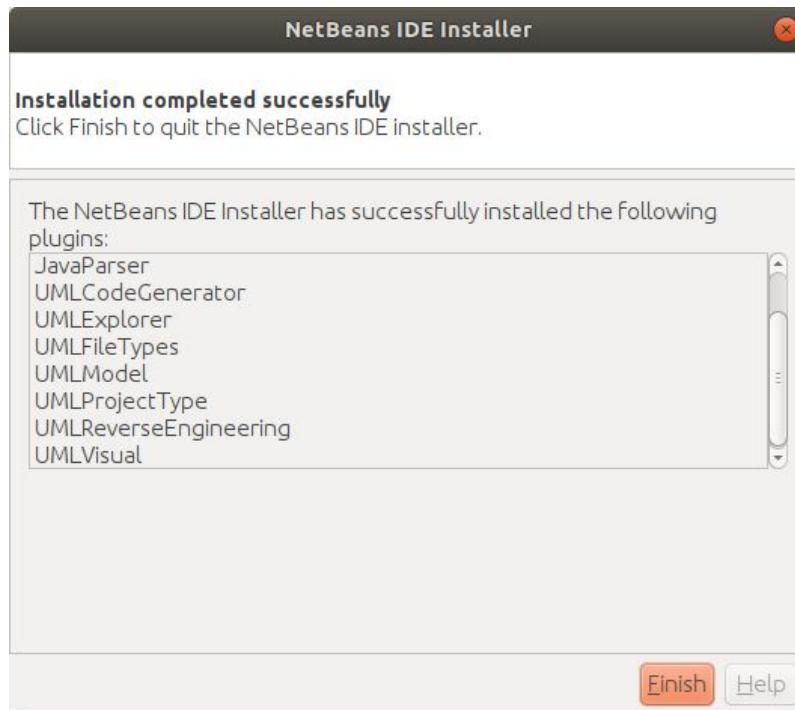
Para deixar no directorio easyuml o resultado da descompresión.

En Apache NetBeans, seleccionamos **tools, plugins** e a pestana **downloaded**.

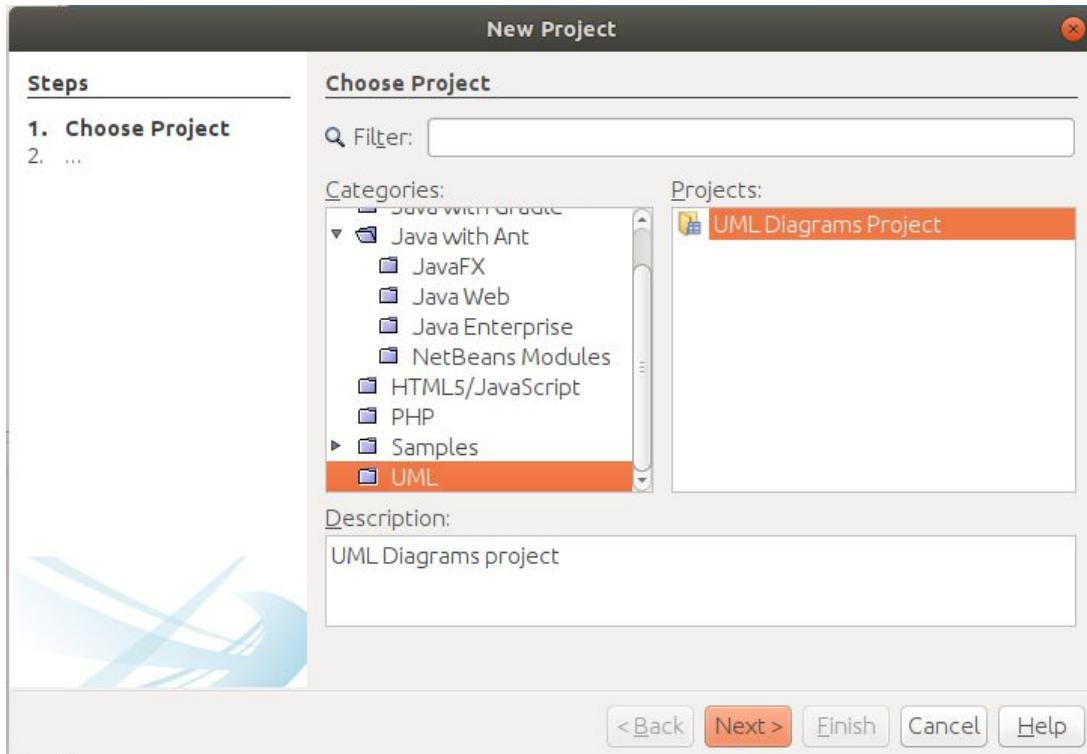
Facemos clic en **add Plugins...** e navegamos ata o directorio onde se descomprimiu eo zip e seleccionamos todos os arquivos **.nbm**.



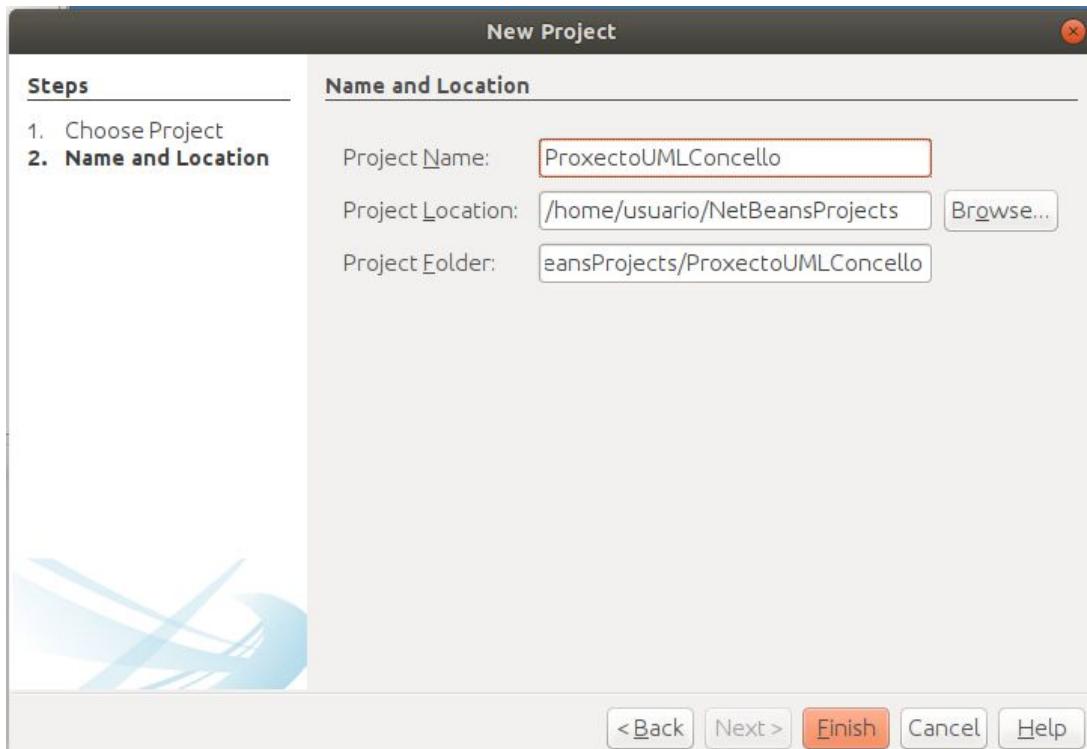
Pinchamos no botón **Install** e procedemos á instalación.



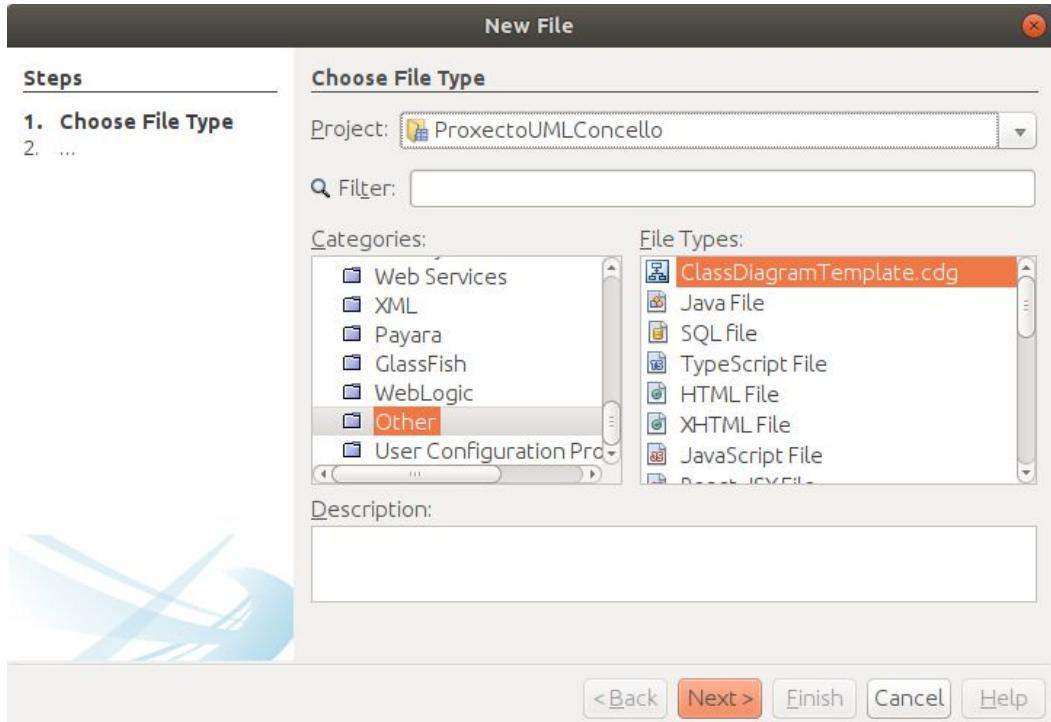
Despois da instalación comprobaremos que na opción de crear un proxecto novo aparece agora a posibilidade de crear un proxecto UML:



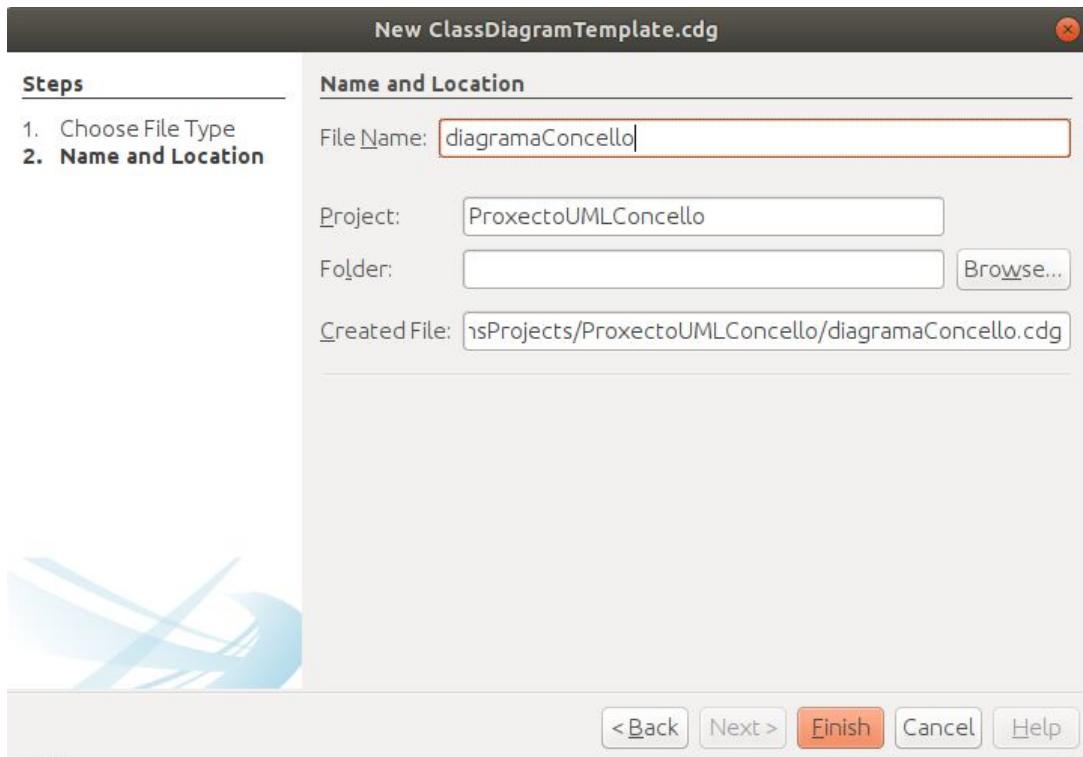
Creamos o proxecto UML co nome que consideremos.



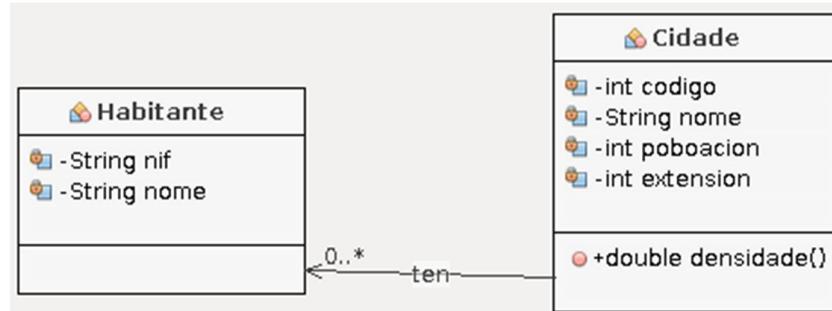
Unha vez creado o proxecto, para agregar un novo diagrama UML facemos clic co botón derecho do rato sobre o cartafol **Class Diagrams** e seleccionamos **New -> Class diagram** para elixir finalmente **ClassDiagramTemplate.cdg**.



A continuación escribimos o nome do diagrama e xa podemos comezar a deseñar o noso diagrama.



Podemos crear o diagrama arrastrando os componentes e relacións desde a paleta (situada á dereita) ou facendo clic dereito sobre a folla de deseño.



Para agregar novos atributos e métodos podemos facer dobre clic nas seccións correspondentes das clases ou clic dereito sobre estas.

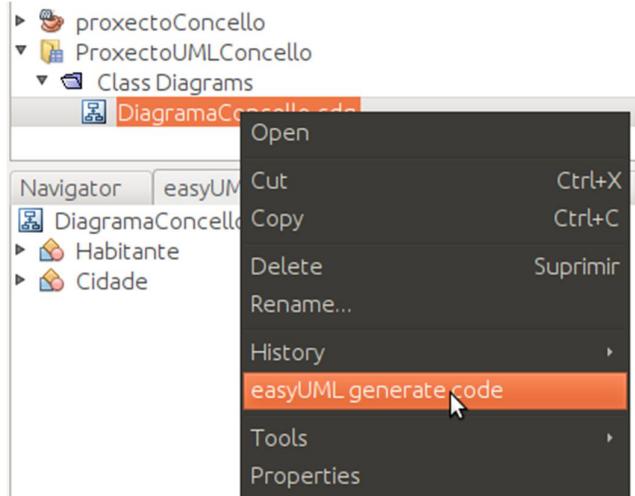
Xeración de código a partir de diagramas de clases

Imos ver como se pode xerar código a partir dos diagramas de clases UML xerados con easyUML.

En primeiro lugar abrimos o proxecto que contén o diagrama de clases a partir do cal queremos xerar código.

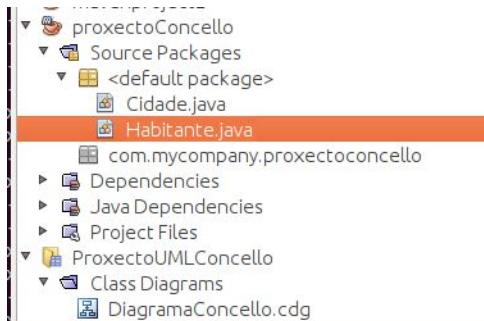
A continuación, creamos un proxecto Java ao que non lle engadimos nada.

Seguidamente, facemos clic dereito sobre o diagrama UML e seleccionamos **easyUML generate code**.



Na ventá que apareza, debemos buscar o proxecto Java que acabamos de crear e pinchamos no botón **Generate code**.

As clases xéranse no proxecto Java. Se non indicamos clase, crearanse no paquete por defecto (default package).



Abrimos o proxecto e vemos o código xerado.

```

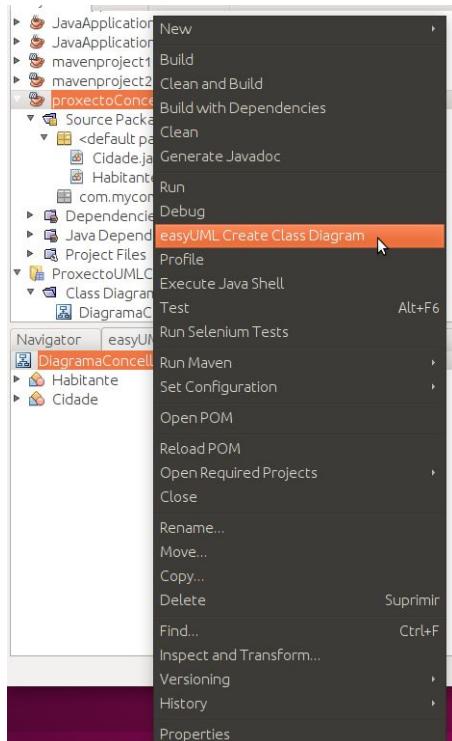
1  public class Habitante {
2
3      private String nif;
4
5      private String nome;
6
7

```

Xeración de diagramas de clase a partir de código (enxeñería inversa)

Tamén podemos facer o proceso contrario ó visto no apartado anterior, é dicir, si temos un proxecto con ficheiros de código fonte podemos crear os diagramas de clases que representen ese código.

Só hai que seleccionar as clases, paquetes ou o proxecto completo e facer clic dereito seleccionando **easyUML Create Class Diagram**.



Na ventá que apareza, debemos buscar o proxecto onde queremos crear o diagrama de clases e pinchamos no botón **Create class diagram**.

Visual Paradigm

Instalación de Visual Paradigm Community Edition

Visual Paradigm é unha das ferramentas mais potentes para o traballo con diagramas UML. É software propietario pero dispón dunha versión gratuíta, a Community Edition, con moitas das funcionalidades da versión de pago.

Para a descarga da versión gratuíta iremos á páxina de descargas e seleccionaremos a Community Edition para o sistema operativo que estamos empregando (<http://www.visual-paradigm.com/download/community.jsp>).

Non hai que confundir esta versión coa versión de proba (FREE Trial) , xa que esta última é unha versión completamente funcional pero limita o seu uso a 30 días.

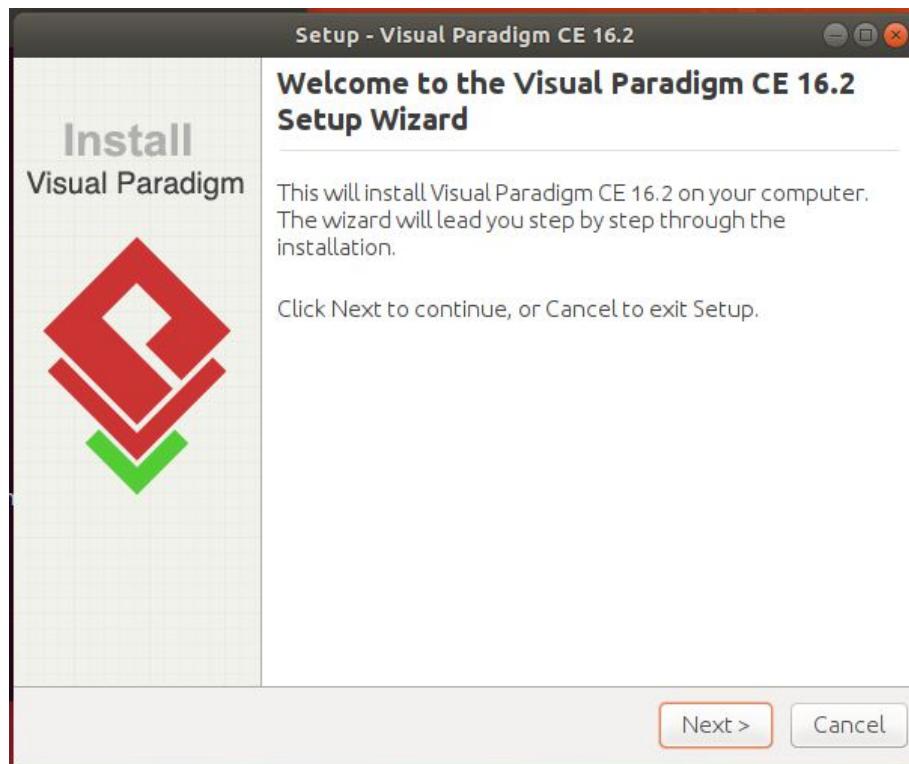
Para a instalación podemos seguir as instrucións que aparecen na URL

https://www.visual-paradigm.com/support/documents/vpuserguide/12/14/6008_windows2000n.html.

En Ubuntu lanzamos a instalación desde o cartafol onde se descargou facendo:

```
bash ./Visual_Paradigm_CE_16_2_20210101_Linux64.sh
```

Aparece o proceso de guía da instalación que comeza coa seguinte pantalla:



A instalación é sinxela.

Unha vez instalado, necesitamos un código de activación que conseguiremos rexistrándonos. Envíano ao correo que se indique no rexistro. A activación non é obligatoria pero si recomendable para que non estea recordándonos continuamente que estamos traballando cunha versión sen activar.

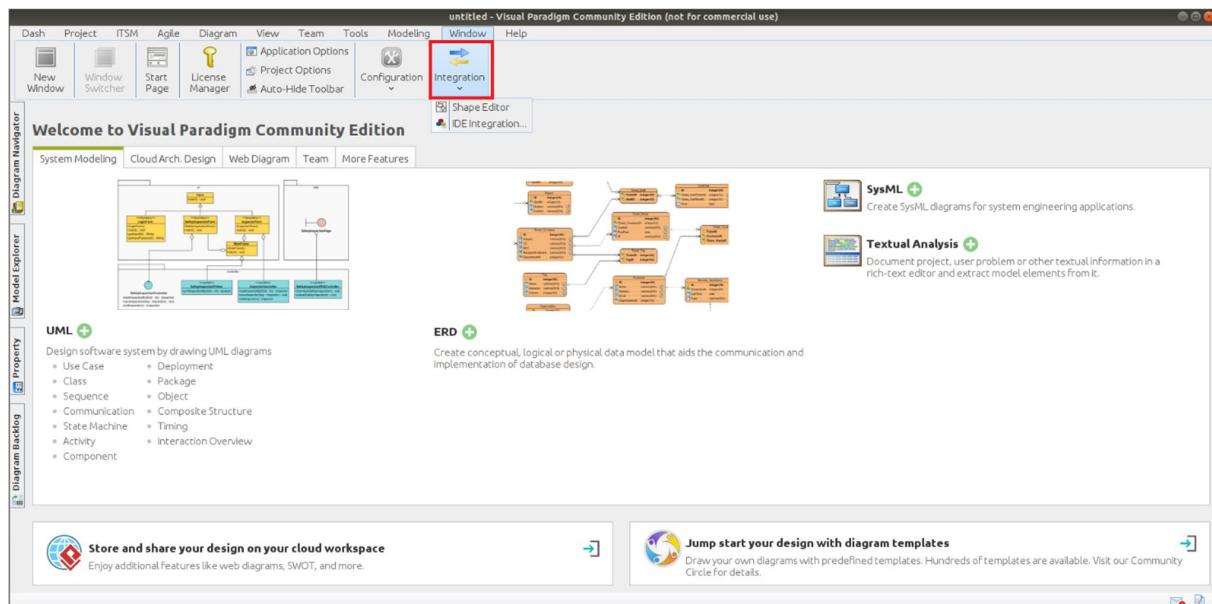
A versión Community Edition inclúe algunas das funcionalidades da versión completa entre as que non está a xeración de código nin a enxeñaría inversa.

Cando abrimos o programa, pregúntanos o directorio por defecto para gardar os proxectos.

Podemos deixar a opción por defecto ou seleccionar o noso propio directorio.

Visual Paradigm permite integrar o contorno de modelado visual con NetBeans proporcionando soporte completo do ciclo de vida do desenvolvemento de software.

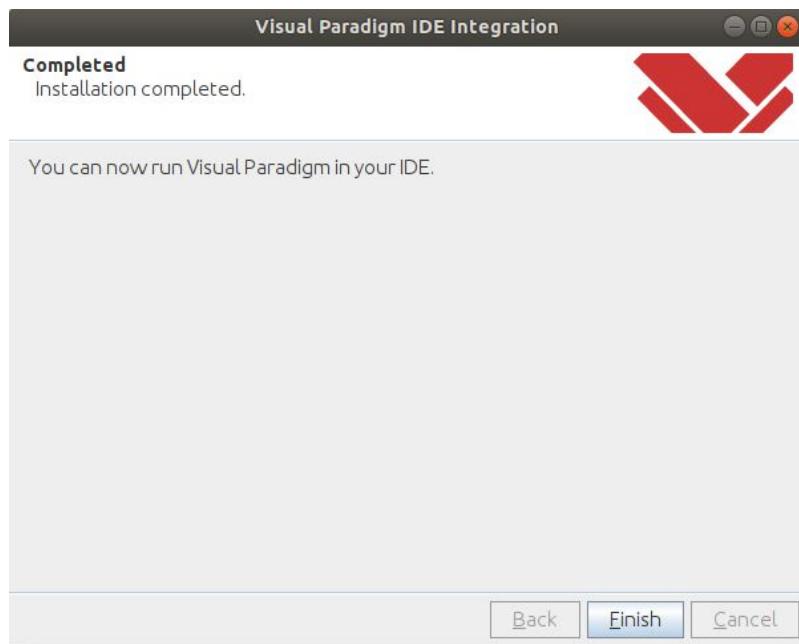
Para instalar a integración de Netbeans desde Visual Paradigm debemos seleccionar Window e pinchar en Integration na barra de ferramentas.



Posteriormente eliximos **IDE Integration...** e, na pantalla que aparece eliximos **Netbeans Integration**, eliximos o directorio onde está instalado o Netbeans e deixamos que o asistente faga a instalación.



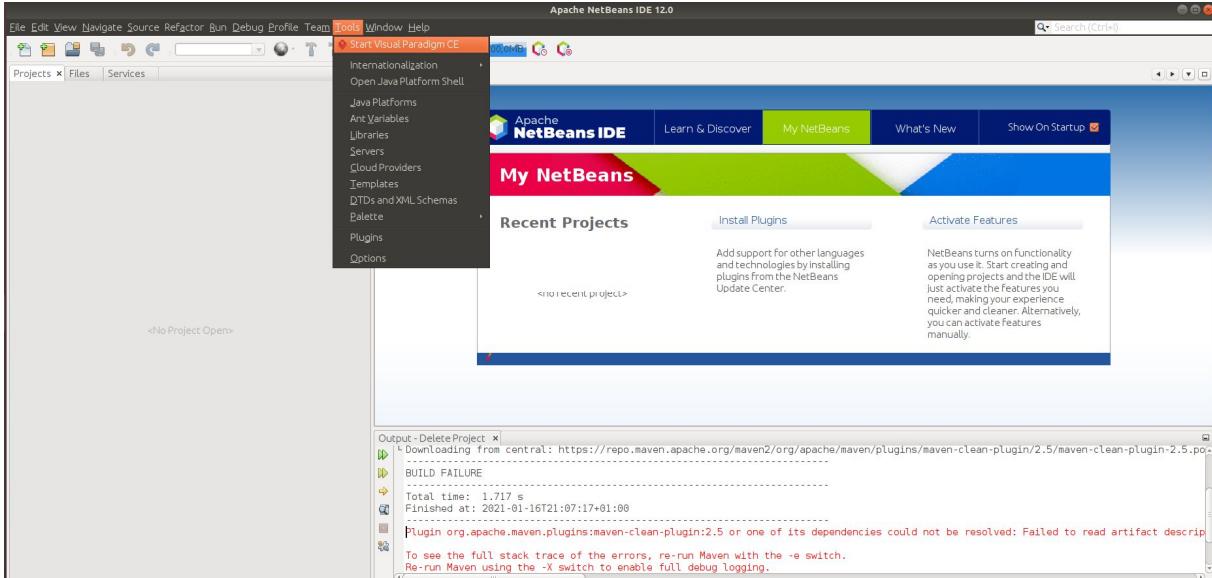
A pantalla que aparece ao rematar a instalación é a seguinte:



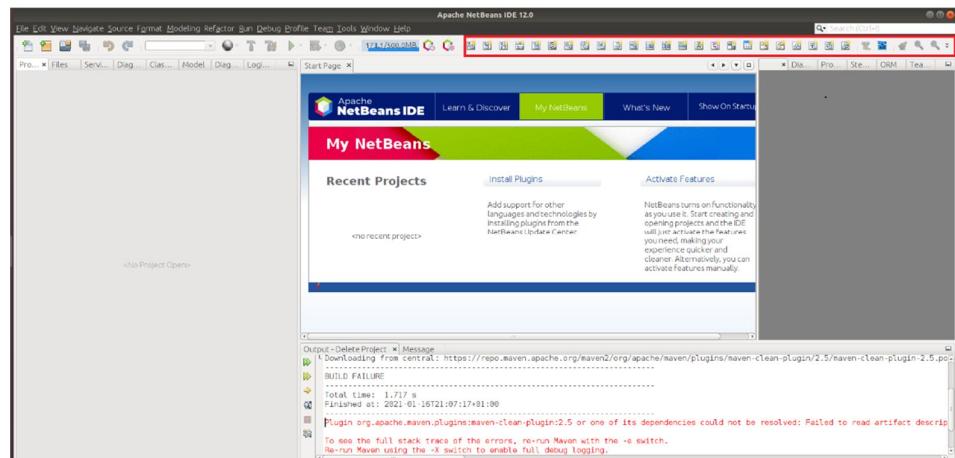
O proceso de instalación modificará/agregará arquivos no directorio de instalación de Netbeans.

Se abrimos o Netbeans, podemos ir a **Tools** e seleccionar **Start Visual Paradigm CE**.

Cando nos pida a ruta se seleccionamos **Create in default path**, gardará o proxecto UML nun novo directorio vpproject dentro do directorio do proxecto. Se seleccionamos **Create or select existing project at external location** debemos especificar a ruta do proxecto que se desexe.

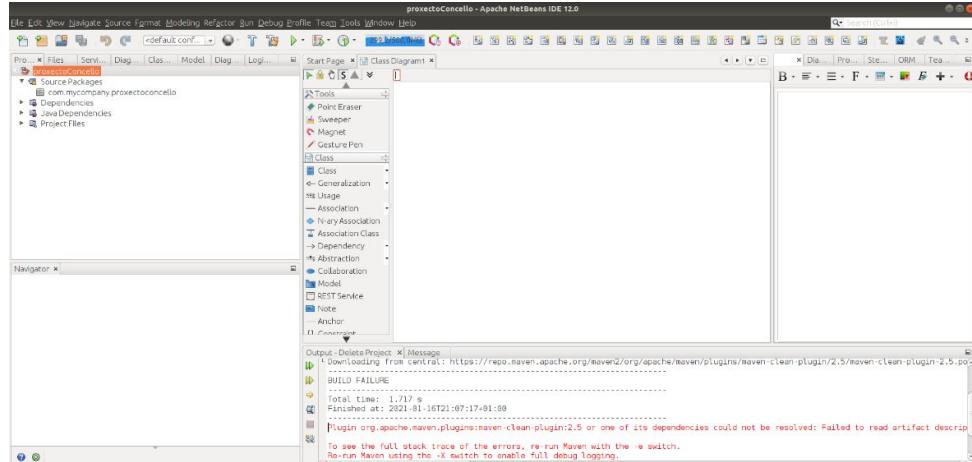


Unha vez iniciado Visual Paradigm no IDE, aparecerán unha serie de botóns na zona superior dereita que nos permitirán crear os diferentes diagramas de UML.

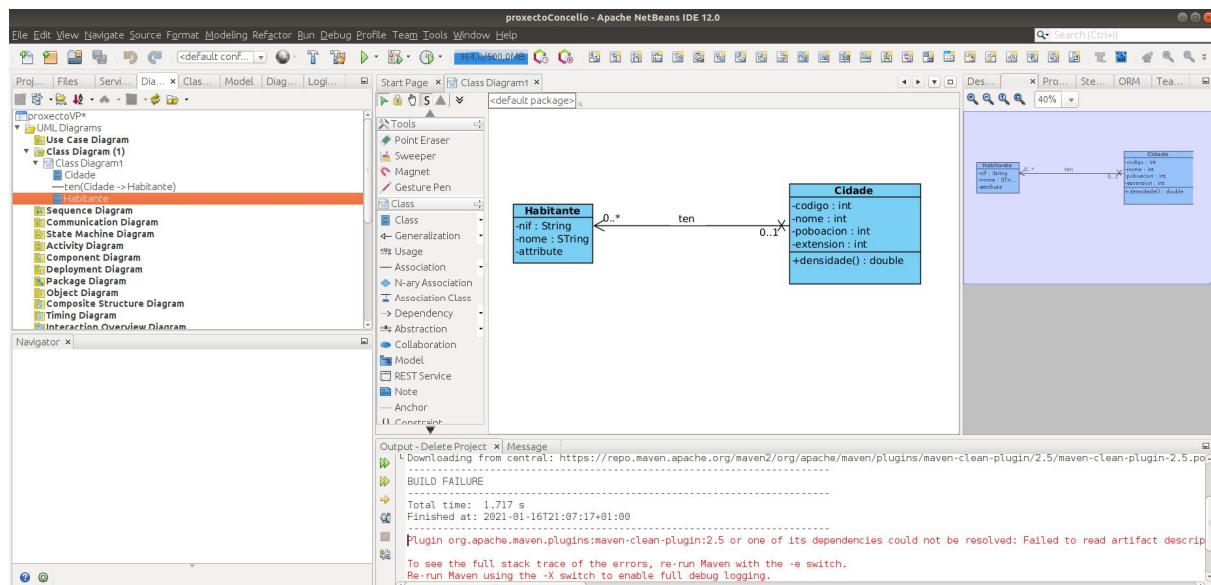


Podemos crear un proxecto UML para calquera dos proxectos Java que temos en Netbeans. Un proxecto Java pode asociarse como máximo cun proxecto UML e non pode crearse un proxecto UML sen asocialo a ningún proxecto Java.

Seleccionamos diagrama de clases no menú e xa podemos comenzar a deseñar o noso diagrama de clases.



Un exemplo de diagrama de clases sería o seguinte:



Xeración de código a partir de diagramas de clases

Para poder xerar código a partir de diagramas de clases imos descargar a versión de avaliación de 30 días de Visual Paradigm.

Dende a propia versión *Community Edition* permítesenos facer o cambio (non e necesario descargar e instalar outra versión).



Download Visual Paradigm

No risk. No obligation. No registration. 30-day FREE Trial

Version: 16.2

Build number: 20210101

Download Visual Paradigm
FREE Trial

SSL Secure Connection

For Windows 64bit

[More Options](#) | [System Requirements](#) | [End User License Agreement](#)

[Download Features List](#)

[Product Leaflet](#) | [Brief Feature List](#) | [Full Feature List](#)

Other download

[Get Community Edition
FREE for non-commercial use](#)

[Get VP-Viewer
FREE VP project browser](#)

[Download Old Versions](#)

[Uninstall Guidelines](#)



Teaching with Visual Paradigm is a pleasure. It is easy-to-use, it is intuitive, and above all it does not get in the way of conveying the semantics of object oriented modeling.

- Dr. Ajaz Rana, Temple University, Philadelphia

Visual Paradigm

Product

[Features](#)
[Editions](#)
[Try Now](#)
[Pricing](#)
[Visual Paradigm Online](#)

Support

[Forums](#)
[Request Help](#)
[Customer Service](#)

Learn

[Community Circle](#)
[Know-how](#)
[Demo Videos](#)
[Tutorials](#)
[Documents](#)

About Us

[Visual Paradigm](#)
[Newsroom](#)
[YouTube Channel](#)
[Academic Partnership](#)

All rights reserved | [Legal](#) | [Privacy statement](#)

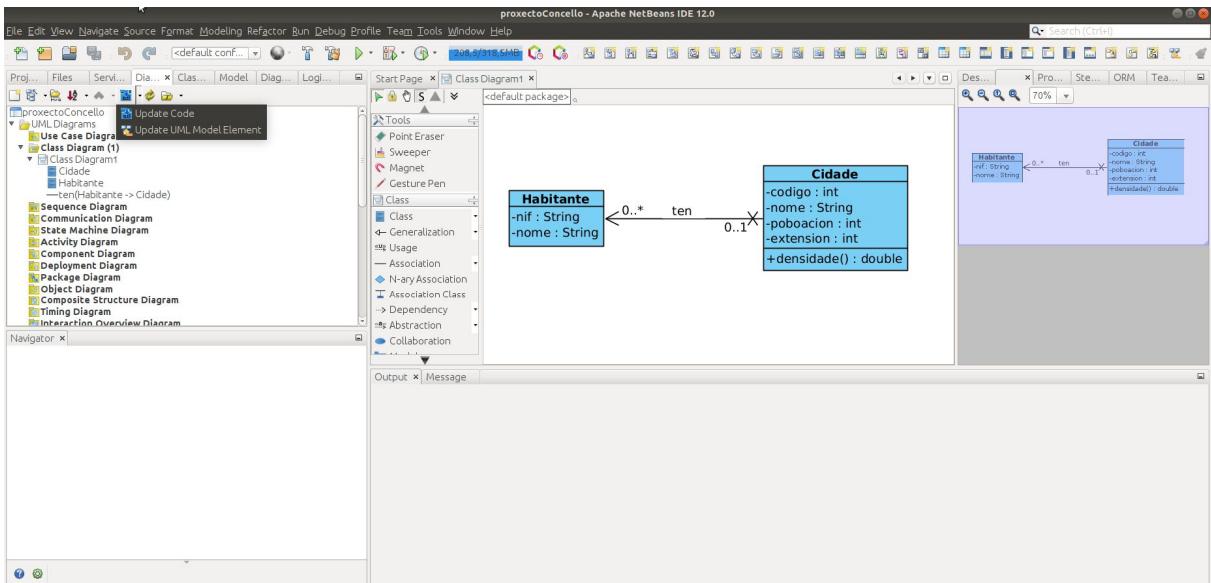
[Facebook](#) [Twitter](#) [Google+](#) [Email](#)

A instalación faise da mesma maneira que a indicada para a Community Edition.

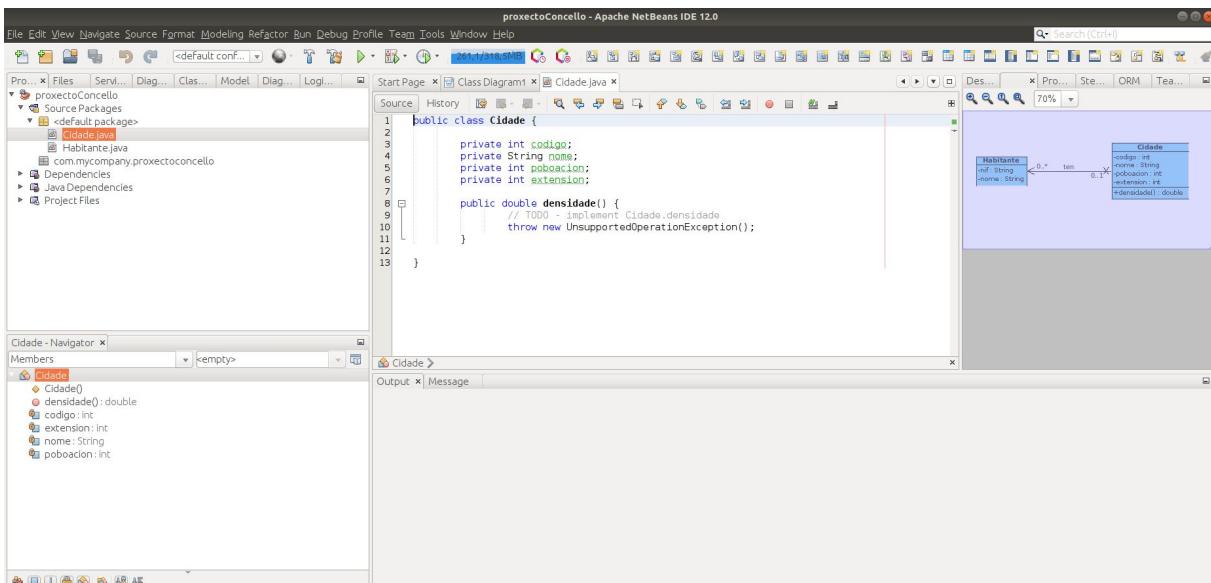
Unha vez instalado o Visual Paradigm, abrímolo e indicamos que queremos avalialo.

Cando o programa teña arrancado, facemos a integración con Netbeans.

Abrimos o proxecto para o que queremos xerar código. A xeración pódese facer de moi diversas maneiras. Unha delas é ir ao panel de Navegación de Diagramas (Diagram Navigator) e facer clic no botón **Update Code** (Actualizar código).



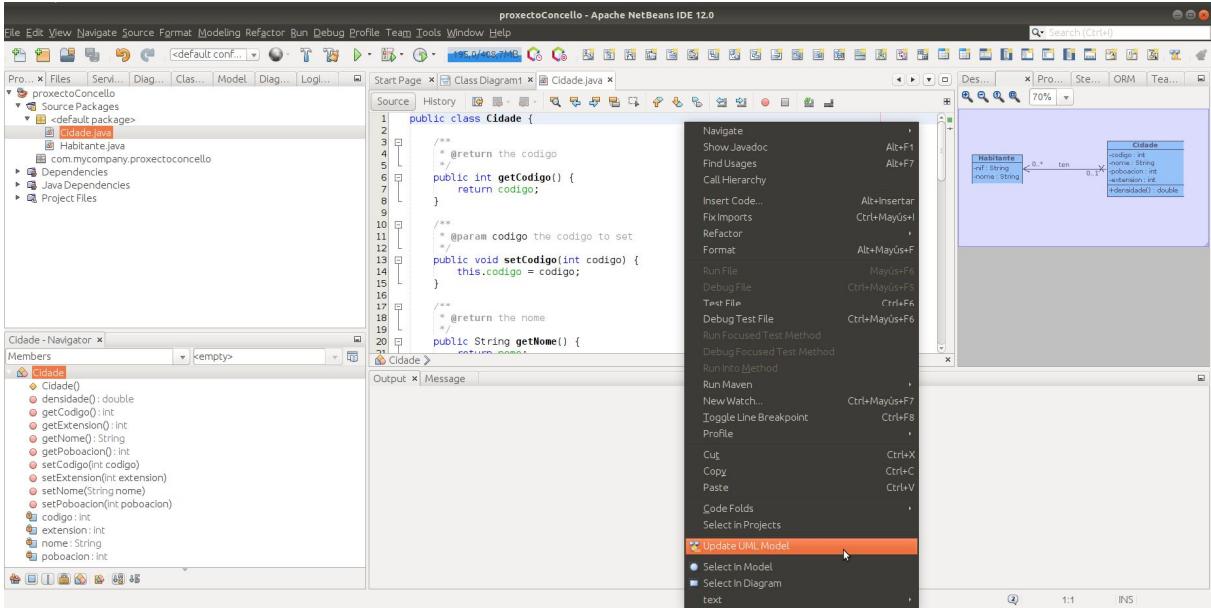
No panel Proxectos (Projects), expandimos o nodo do proxecto e marcamos o paquete de orixe. As clases Cidade.java e Habitante.java aparecen aí e pode verse o seu código.



Xeración de diagramas de clase a partir de código (enxeñería inversa)

Podemos facer as modificacións no código que consideremos a un proxecto xa creado ou crear un novo proxecto Java e xerar o diagrama de clases correspondente.

Polo tanto, cando teñamos feito os cambios que queremos no código e desde a zona onde se edita o código, pinchamos co botón dereito do rato e seleccionamos Update UML Model (Actualizar modelo UML).



Tras a actualización, debe aparecernos o diagrama de clases cambiado (tendo en conta as modificaciós feitas no código).

