UNIDAD DIDÁCTICA 5

ELABORACIÓN DE DIAGRAMAS DE CLASES

MÓDULO PROFESIONAL: ENTORNOS DE DESARROLLO

Índice

RESUMEN INTRODUCTORIO

A lo largo de la unidad se conoceran los conceptos relacionados con los diagramas UML, más concretamente con los diagramas de Clases, relacionados con la programación orientada a objetos.

Se comenzará con los conceptos de programación orientada a objetos, como Clase, Objeto, Herencia, Composición, … y la forma de representar los mismos en un diagrama.

Se continuara con las distintas herramientas que se pueden usar para la generación de diagramas como Draw.io, Umbrello, Dia,…

Por último se pasará a conocer herramientas para la generación automatica de código mediante diagramas, y la de generación de diagramas a traves de código ya escrito.

INTRODUCCIÓN

En el mundo de desarrollo del software existen una serie de diagramas que se utilizan para trasladar a todas las personas que integran el equipo como realizar el software y que es lo que debe hacer. En el caso de esta unidad se estudiarán los diagramas de clase y las herramientas correspondientes para trabajar con ellos. Este tipo de diagramas es indispensable cuando se está realizando software con programación orientada a objetos.

CASO INTRODUCTORIO

En el equipo de proyecto en el que trabajas, existen 2 analistas y 2 diseñadores. Uno de los clientes de la factoría de software decide aumentar la funcionalidad de un software que fue realizado por el equipo de Juan y que se encuentra actualmente en fase de mantenimiento preventivo. Para ello contacta con el equipo y presenta una serie de mejoras y novedades con respecto a la funcionalidad anterior. En este punto, se hace necesario retomar todos los diagramas de clase de la aplicación actual, con el objetivo de analizar de nuevo su funcionalidad y adaptar los cambios propuestos por el equipo en la mayor brevedad posible. Sin la existencia de estos diagramas de clase, no sería viable abordar la actualización del software del cliente en un corto periodo de tiempo.

Al finalizar la unidad serás capaz de conocer que son los diagramas UML, conocer la notación para los diagramas de clases, lo que te permitirá realizar los diagramas e

interpretarlos, identificar las relaciones existentes entre clases, y por último realizar ingeniería inversa en lenguaje JAVA y C++.

1. DIAGRAMAS DE CLASES: NOTACIÓN

Durante el desarrollo de la aplicación de gestión de tiendas de ropa, se han identificado una serie de mejoras que deben llevarse a cabo dentro del proyecto, para ello se retomaran los diagramas que se utilizaron para su desarrollo con el fin de conocer en profundidad el desarrollo del proyecto, y se realizaran modificaciones a los mismos para añadir las nuevas implementaciones. Para ello es imprescindible que el equipo se forme y tenga conocimiento de cómo interpretar y realizar estos diagramas.

El Lenguaje de Modelamiento Unificado (**UML** - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.

PARA SABER MÁS

Información sobre UML y los distintos diagramas que se pueden generar:

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. Por ejemplo, si se quiere diseñar un programa para gestionar un concesionario de coches, se va a tratar con distintos objetos como pueden ser coches, empleados del concesionario, clientes, etc. Cada uno de los coches es un objeto, pero para hacer una programación general necesitamos poner en común lo que tienen los coches y crear un molde con las características de los coches y con las operaciones en común que tienen. Ese molde es el que se conoce como clase en programación orientada a objetos.

En el diagrama de clases, además de representar las clases, se representan los siguientes elementos:

- **Clase**: Atributos, Métodos y Visibilidad.
- **Relaciones**: Herencia, Composición, Agregación, Asociación y Uso.

PARA SABER MÁS

Para ampliar información sobre los diagramas de clases visite la web de Microsoft:

1.1 Notación de Clase

Las **clases** se representan por rectángulos que muestran el nombre de la clase y opcionalmente el nombre de las operaciones y atributos. Los compartimientos se usan para dividir el nombre de la clase, atributos y operaciones. Adicionalmente las restricciones, valores iniciales y parámetros se pueden asignar a clases. En UML, una clase es representada por un rectángulo que posee tres divisiones:

Representación UML de una clase

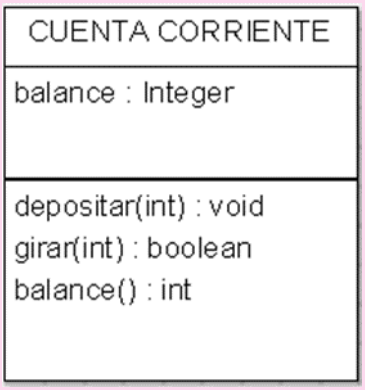
En donde cada una de las partes representa su cometido:

- **Superior**: contiene el **nombre** de la Clase. Los nombres deben de ser descriptivos y no muy largos, con la primera letra de cada palabra que lo forma en mayúsculas.
- **Intermedio**: contiene los **atributos** (o variables de instancia) que caracterizan a la Clase. En el ejemplo del concesionario de coches, para la clase Coche, podemos utilizar los atributos marca, modelo, color, año de fabricación, número de bastidor, entre otros. Estos atributos además tienen un grado de visibilidad que puede ser public, private, protected.
- **Inferior**: contiene los **métodos** u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

EJEMPLO PRÁCTICO

Una Cuenta Corriente que posee como característica "Balance". Además, puede realizar las operaciones de: "Depositar", "Girar" y "Balance".

Solución: Diagrama de clases, en este caso la clase única con nombre, atributos y métodos, en este orden:



La notación que precede el nombre del atributo u operación indica la visibilidad del elemento, si se usa el símbolo **+** el atributo y la operación tienen un nivel **público** de visibilidad, si se usa un símbolo **–** el atributo u operación es **privado**. Además, el símbolo **#** permite definir una operación o atributo como **protegido** y el símbolo **~** indica la visibilidad del paquete.

Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

- **public (+)**: El atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private (-)**: El atributo sólo será accesible desde dentro de la Clase.
- **protected (#)**: El atributo no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven.

Los métodos u operaciones de una clase son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

- **public (+)**: El método será visible tanto dentro como fuera de la Clase, es decir, es accesible desde todos lados.
- **private (-)**: El método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden acceder).

- **protected (#)**: El método no será accesible desde fuera de la clase, pero si podrá ser accedido por métodos de la clase además de métodos de las subclases que se deriven.

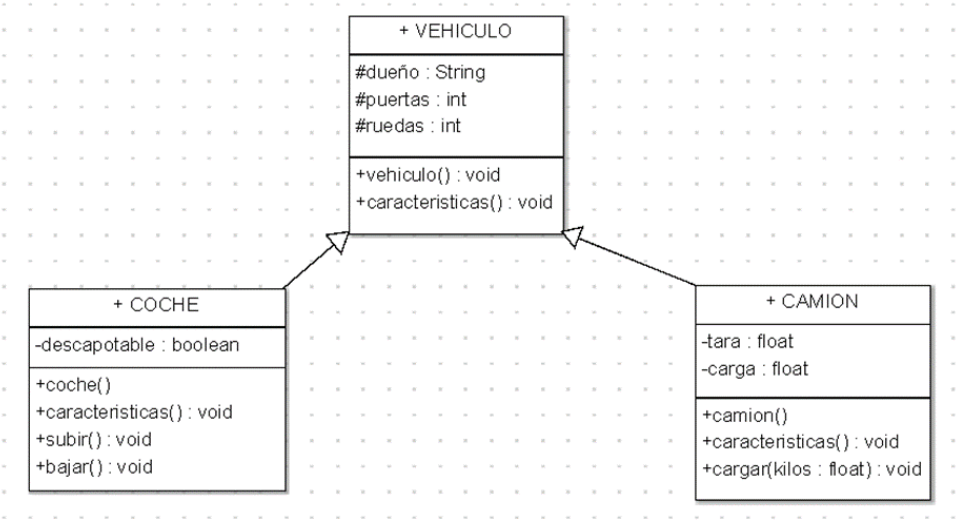
1.2 Relaciones: Herencia, composición, agregación, asociación y uso

Definido el concepto de Clase, se hace necesario detallar cómo se pueden interrelacionar dos o más clases (cada una con características y objetivos diferentes). Previamente, se procede a explicar el concepto de **cardinalidad** de relaciones.
En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, anotándose en cada extremo de la relación y pudiendo ser de tipo:

- **uno o muchos**: 1..* (1..n).
- **0 o muchos**: 0..* (0..n).
- **número fijo**: m (m denota el número).

1.2.1 Herencia (Especialización/Generalización)

Indica que una subclase hereda los métodos y atributos especificados por una superclase, por ende, la subclase además de poseer sus propios métodos y atributos poseerá las características y atributos visibles de la superclase de tipo public y protected (ver ejemplo). En la figura se especifica que Coche y Camión heredan de Vehículo, es decir, Coche posee las Características de Vehículo (dueño, puertas, etc.) además posee algo particular, que es descapotable, en cambio Camión también hereda las características de Vehículo (dueño, puertas, etc.) pero posee como particularidad propia tara y carga. Cabe destacar que fuera de este entorno, lo único visible es el método características aplicable a instancias de Vehículo, Coche y Camión, pues tiene definición pública, en cambio atributos como descapotable no son visibles por ser de tipo privado.



Ejemplo de relación de herencia

Para que exista herencia clara, las subclases deben responder afirmativamente a la expresión “es un”. Por ejemplo, Coche “es un” vehículo y Camión “es un” vehículo. Por tanto, la herencia existe. La notación la marca la flecha hueca.

1.2.2 Agregación

Para modelar objetos complejos, no bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, se presentan dos posibilidades:

- **Por Valor**: Es un tipo de relación **estática**, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada **Composición** (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").
- **Por Referencia**: Es un tipo de relación **dinámica**, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada **Agregación** (el objeto base utiliza al incluido para su funcionamiento).

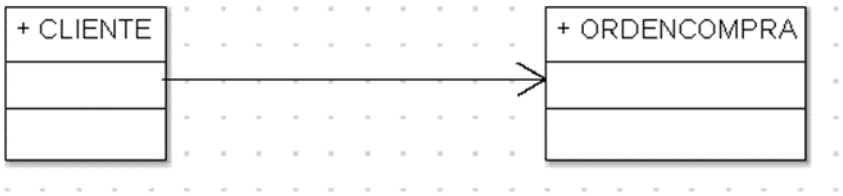
En donde se destaca que:

- Un Almacén posee Clientes y Cuentas (los rombos van en el objeto que posee las referencias).
- Cuando se destruye el Objeto Almacén también son destruidos los objetos Cuenta asociados, en cambio no son afectados los objetos Cliente asociados.
- La composición (por Valor) se destaca por un rombo relleno.
- La agregación (por Referencia) se destaca por un rombo transparente.

La flecha en este tipo de relación indica la navegabilidad del objeto referenciado. Cuando no existe este tipo de particularidad la flecha se elimina.

1.2.3 Asociación

La relación entre clases conocida como Asociación permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

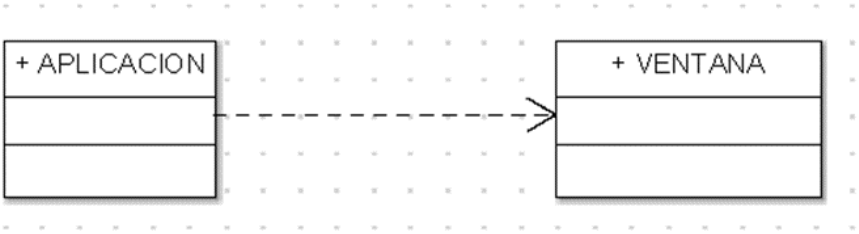


Ejemplo de relación de asociación

Un cliente puede tener asociadas muchas Órdenes de Compra, en cambio una orden de compra solo puede tener asociado un cliente.

1.2.4 Dependencia o Instanciación (uso)

Para comenzar este punto es importante saber que es una instancia, una instancia es un objeto creado a partir de una clase de nuestro programa, por ejemplo, si tengo una clase Coche, ese sería el molde para crear/instanciar objetos. El coche Mercedes, GLA, con 300CV de potencia, color rojo, y número de bastidor EFA3234FSA3342314, es una instancia o un objeto en particular de la clase coche.
La dependencia o instanciación entre distintas clases, representa un tipo de relación muy particular, en la que una clase es instanciada (es dependiente de otro objeto/clase). Se denota por una flecha punteada. El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación gráfica que instancia una ventana (la creación del Objeto Ventana está condicionado a la instanciación proveniente desde el objeto Aplicación):



Ejemplo de relación de uso

Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

ARTÍCULO DE INTERÉS

Mecanismos de herencia y su implementación en Java:

2. HERRAMIENTAS PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASES. INSTALACIÓN

A la hora de realizar o modificar los diagramas que conforman el diseño de nuestra aplicación de gestión de tiendas de ropa se plantea la necesidad de conocer las distintas herramientas de mercado que se pueden utilizar para desarrollar estos diagramas.

UML incluye un conjunto de técnicas de notación gráfica para especificar, visualizar y documentar los modelos de sistemas de software, incluyendo su estructura y diseño, de manera que cumpla con todos estos requisitos. Existen muchas herramientas profesionales de diagramas populares, tanto comerciales como de software libre.

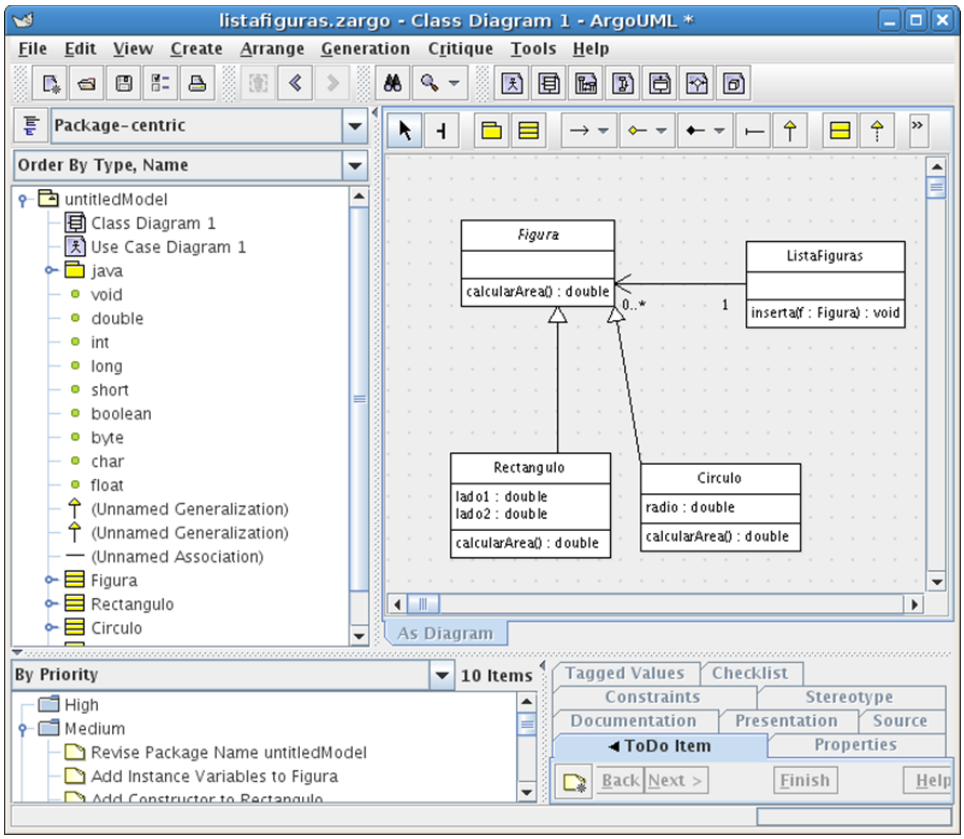
A continuación, se indican las principales herramientas UML de código abierto: StarUML, ArgoUML, Violet UML Editor, Astah Community, BOUML, Dia, UMLet, UMLGraph, MetaUML, Visual Paradigm for UML 10.1 Community Edition, T4 Editor plus modeling tools, NetBeans IDE UML y Eclipse UML2 Tools. Por su parte, algunos generadores Online de Diagramas UML son yUML y zOOML.

PARA SABER MÁS

Listado muy completo de herramientas para la elaboración de diagramas de clases:

2.1. ArgoUML

ArgoUML es una herramienta implementada con la cual se tiene la posibilidad de crear modelos UML que serán compatibles con los estándares de la versión 1.4 de dicho lenguaje. Puede ser utilizado en cualquier plataforma de Java y además se encuentra disponible en 10 idiomas (inglés, italiano, francés y español entre otros). Se pueden crear diferentes tipos de diagramas, entre ellos diagrama de clases, diagrama de estados, diagrama de actividad, diagrama de casos de uso, diagrama de colaboración, diagrama de despliegue y diagrama de secuencia. Se tiene la posibilidad de salvar esos diagramas en los siguientes formatos: PNG, GIF, SVG y PostScript.



Infertaz de ArgoUML

ArgoUML aunque es un programa estable que tiene una gran funcionalidad, es un programa que tiene mucho tiempo, el sitio oficial ha desaparecido, y no tiene soporte, por lo que está en desuso.

Al estar desarrollado en una versión de JAVA inferior, da problemas de compatibilidad con las nuevas Java Virtual Machine.

VÍDEO DE INTERÉS

Instalación de ArgoUML:

2.2. StarUML

StarUML es una herramienta de programación escrita en código abierto y de distribución libre que genera (Diagramas de clases, estructuras, componentes, paquetes, objetos, actividades, módulos, comunicación, estados, actividades, secuencias, etc.). Estos diagramas tienen como función explicar el proceso que hace cada objeto y elemento de la aplicación, de modo que convierte el diseño gráfico en una serie de esquemas y códigos necesarios para el buen funcionamiento del programa. Es compatible con lenguaje C++ o Java. Presenta plantillas predeterminadas que ayudan a entender cómo funciona este sistema, posibilitando crear diseños propios a partir de modelos propuestos.

VÍDEO DE INTERÉS

Crear código Java a partir de diagramas de clases con StarUML:

2.3. Umbrello

Es una herramienta de diagramas UML que resulta de ayuda durante el proceso de desarrollo de software, en la fase de análisis y diseño para analizar y diseñar lo que se quiere crear, en la fase desarrollo para saber lo que hay que desarrollar y como. También se puede usar UML para documentar nuestros propios diseños de software.

Umbrello puede manejar gran parte de diagramas UML, pudiendo crearlos desde cero manualmente, o crearlos automáticamente desde código ya escrito en distintos lenguajes de programación, como pueden ser C++, java, Python, IDL, Pascal/Delphi, Ada, o Perl. Con esto estamos utilizando un código ya escrito y del que en principio no tenemos diagrama, para crear un diagrama y poder modificar el diseño inicial.

También se puede generar código en los lenguajes antes mencionados a partir de los diagramas creados o modificados.

El formato de fichero que utiliza para la creación de diagramas es XMI.

Umbrello también nos va a permitir la distribución de los modelos exportándolos en los formatos DocBook y XHTML, lo cual va a facilitar los proyectos colaborativos donde los desarrolladores no tienen acceso directo a Umbrello, o donde los modelos van a ser publicados vía web.

Si el usuario lo desea, puede agrupar varios diagramas relacionados en un solo fichero XMI. Estos estarán organizados en diferentes vistas (lógica, de casos de uso, de componentes, etc.), que a su vez pueden contener diagramas o carpetas con las que clasificarlos aún más.

Umbrello UML Modeller nos va a permitir usar los siguientes tipos de diagramas:

- Clase
- Secuencias
- Colaboraciones
- Caso de uso
- Estados
- Actividades
- Componentes
- Despliegue
- Entidad relación

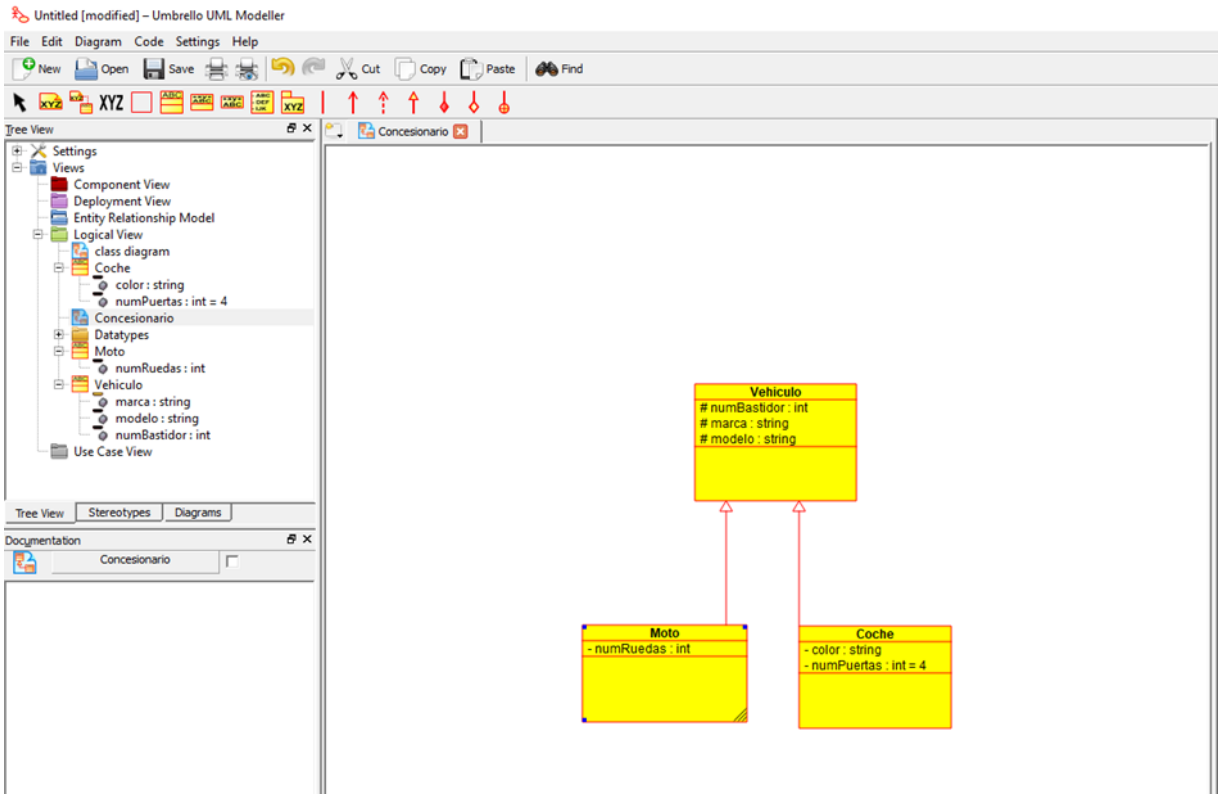


Diagrama de clases con Umbrello

VÍDEO DE INTERÉS

Crear código Java a partir de diagramas de clases con Umbrello:

2.4. Draw.io

Draw.io es una plataforma para la creación y la edición de diagramas libres, entre ellos los diagramas UML.

Una de las características principales de draw.io es que se puede trabajar con esta plataforma sin necesidad de tener instalada ninguna aplicación, puede funcionar directamente en algún navegador.

Está desarrollada principalmente en el lenguaje de programación JavaScript y cuenta con la Apache License en su versión 2.0.

Además, Draw.io se caracteriza por ser un programa de fácil funcionamiento, debido a que facilita el uso de herramientas básicas y sencillas como el sistema drag and drop, también conocido como arrastrar y soltar, y que se utiliza para llevar formas, imágenes, flechas y otros elementos de manera rápida. También facilita la creación de colecciones de diagramas, así como imágenes personalizadas que puedes utilizar en los diagramas diseñados.

Esta aplicación de creación de diagramas también está disponible para la descarga en su versión offline, que permite el trabajo sin conexión, a través de su descarga previa para el uso en plataformas como Windows, Mac Os, Linux y el entorno GNU, Chrome OS, entre otros.

Draw.io está en constante evolución, de momento no se puede generar código a partir de los diagramas creados en esta aplicación, pero viendo la evolución.

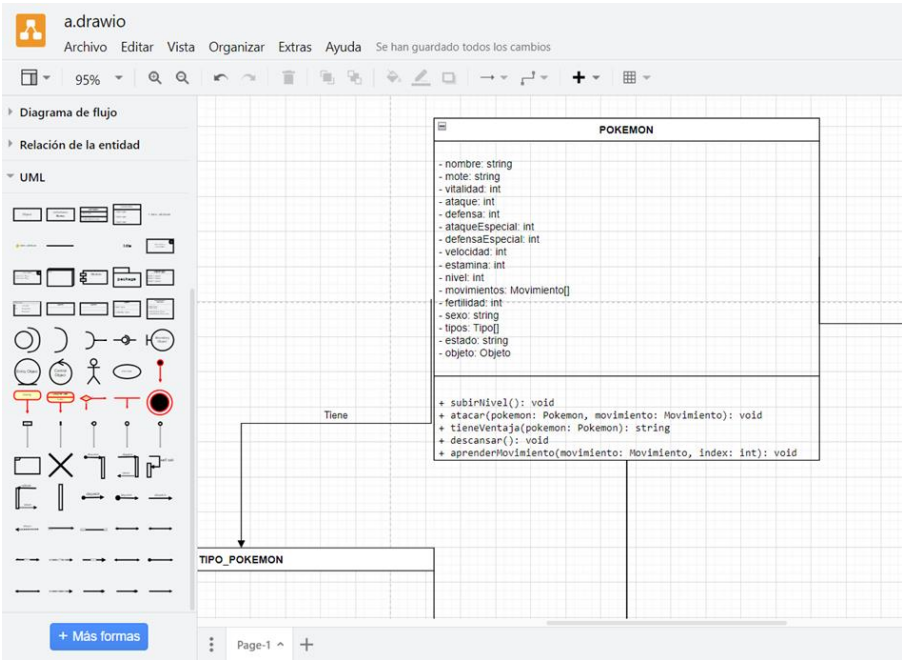


Diagrama de clases con draw.io

VÍDEO DE INTERÉS

Creación de diagrama de Clases:

2.5. Dia

Dia es una aplicación informática para la creación de diagramas. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades.

Este programa se diseñó como un sustituto de la aplicación comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML.2.

Gracias al paquete dia2code, es posible generar el esqueleto del código a escribir, si se utiliza con tal fin un UML.

El formato para leer y almacenar gráficos es XML (comprimido con gzip, para ahorrar espacio). Puede producir salida en los formatos EPS, SVG y PNG.

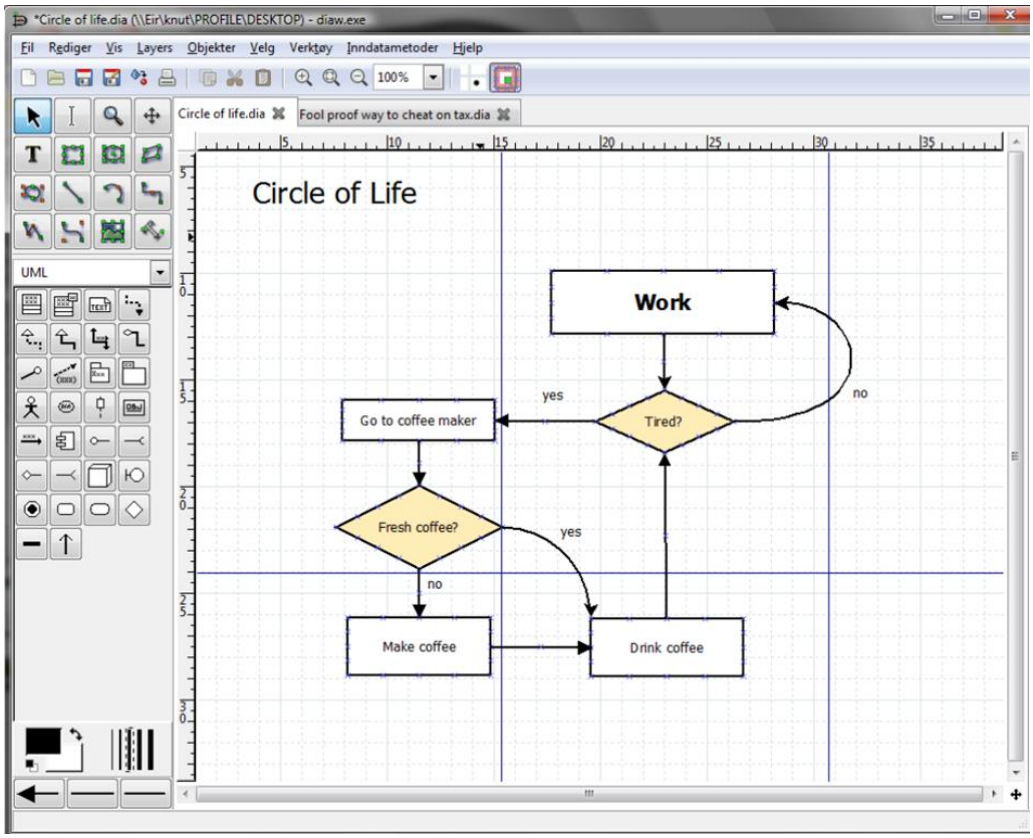


Diagrama de clases con DIA

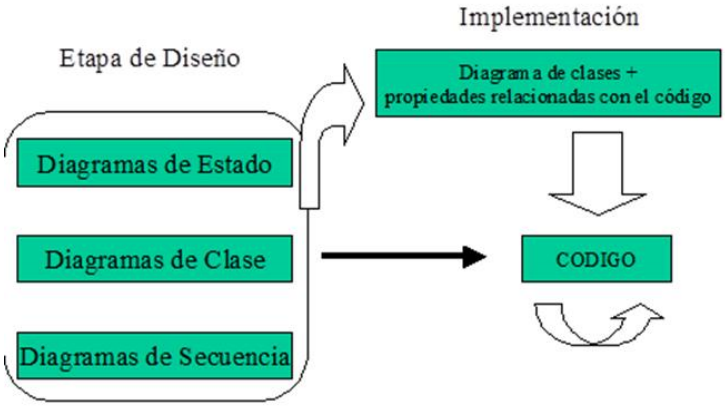
VÍDEO DE INTERÉS

Creación de diagrama de clases con DIA:

3. GENERAR CÓDIGO A PARTIR DE DIAGRAMAS DE CLASES

Al realizar las modificaciones en los distintos diagramas de clases del proyecto de gestión de tiendas de ropa, se han diseñado, nuevas clases, métodos, interfaces, … dentro del diseño. Algunas de las herramientas que se conocen pueden generar parte del código automáticamente por lo que es necesario saber utilizar dichas herramientas en ese sentido.

Son muchas las herramientas que permiten generar código a partir de diagramas de clase y viceversa. A continuación, se presentan un ejemplo con Visual Studio Ultimate.

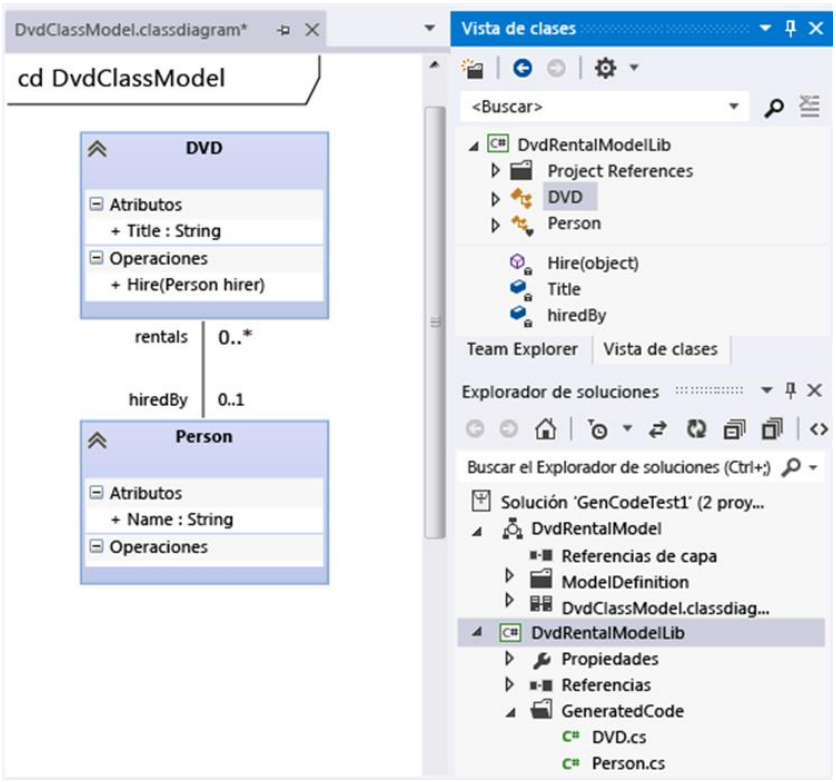


Esquema generación de código a partir de diagramas

3.1. Visual Studio Ultimate

En Visual Studio Ultimate se puede generar código a partir de los diagramas de clases UML mediante el comando de **Generar código**. De forma predeterminada, el comando genera un tipo de C# para cada tipo UML que seleccione. Se puede modificar y extender este comportamiento modificando o copiando las plantillas de texto que generan código. Se puede especificar un comportamiento diferente para los tipos contenidos en los diferentes paquetes del modelo. El comando Generar código es especialmente adecuado para generar código a partir de la selección de elementos del usuario y para generar un archivo para cada clase UML u otro elemento. Por ejemplo, la captura de pantalla muestra dos archivos de C# generados a partir de dos clases UML.

Como alternativa, para generar código en el que los archivos generados no tienen una relación de 1:1 con los elementos UML, se puede considerar la posibilidad de escribir plantillas de texto que se invocan con el comando Transformar todas las plantillas.



Generación de código a partir de diagrama de clases

3.2. Generar código a partir de diagrama de clases en Umbrello

Creamos un diagrama de clases en Umbrello como el que sigue, en este caso simulando un concesionario de coches:

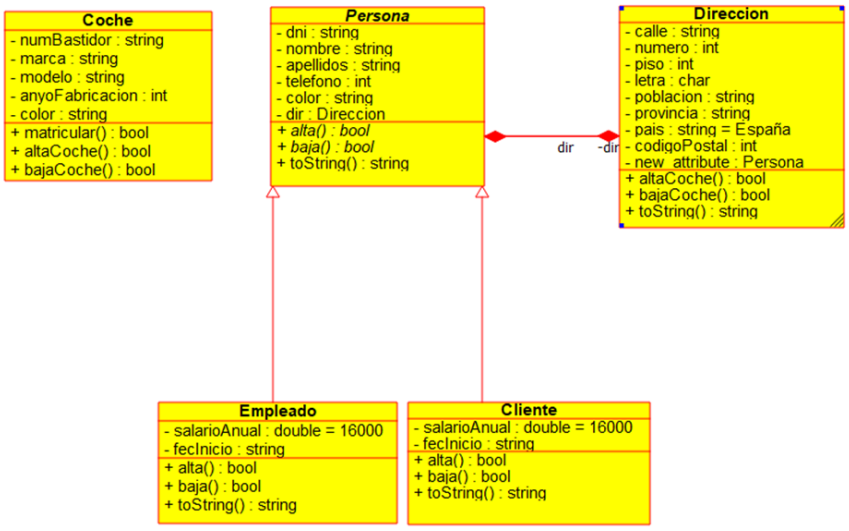
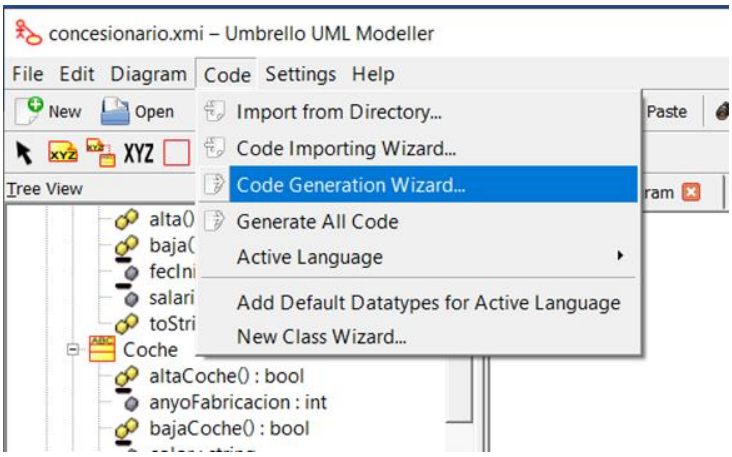


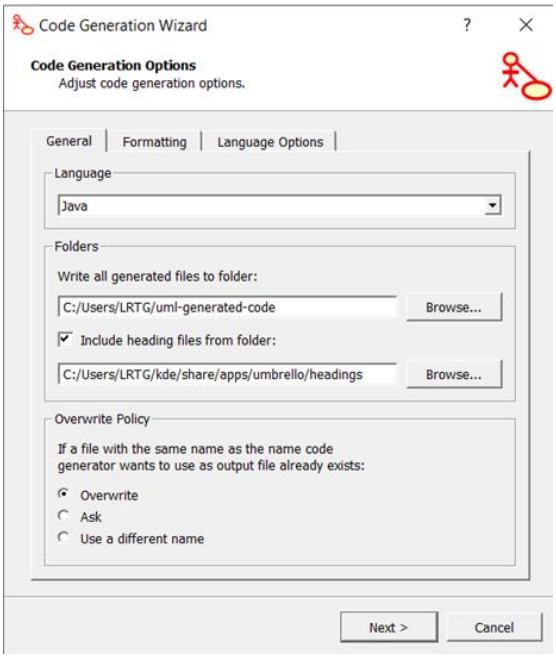
Diagrama de clases de Concesionario de coches

Una vez que tenemos el diagrama hecho con todas las opciones habilitadas, es decir, se han creado las clases, con sus atributos, los métodos, las propiedades de visibilidad, etc, pinchamos en la opción Code:



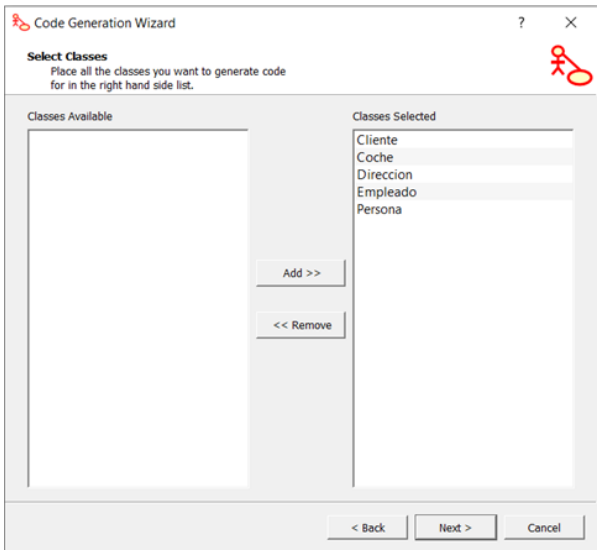
Umbrello para generar código 1

Nos aparece la siguiente venta donde se elegirá el lenguaje de programación, y nos da la opción de si existen métodos con el mismo nombre si deseamos sobreescribir, renombrarlos, o preguntarnos qué es lo que hace. En este ejemplo, se han creado los métodos de alta y baja tanto para la superclase, como para la subclase con el mismo nombre, además el método es abstracto, por lo que deberá sobrescribirlo. También estará la ruta donde se nos generará el código. Se pulsa en Next.



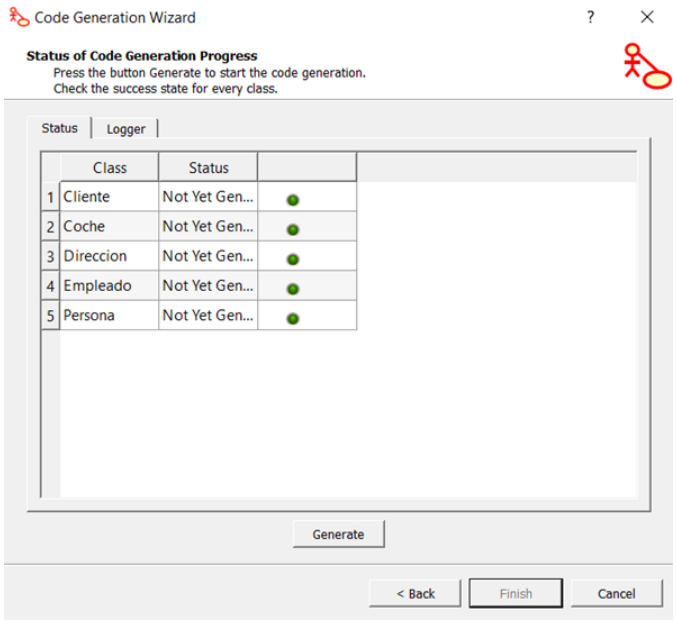
Umbrello para generar código 2

Aparece una pantalla donde se pueden seleccionar las clases de las que se pretende generar el código automáticamente, se seleccionan y pulsa Next.



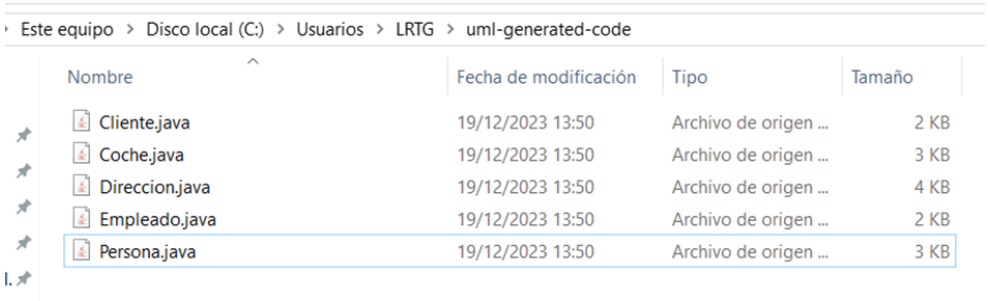
Umbrello para generar código 3

En la siguiente pantalla nos indica si se han generado ya alguna vez, en este caso no es así, presiona en el botón Generate.



Umbrello para generar código 4

Al acabar el estado pasa a verde y se pulsa en finalizar, en la ruta que se ha indicado aparecen los archivos .java que ha generado el programa y que podemos llevar a nuestro proyecto.

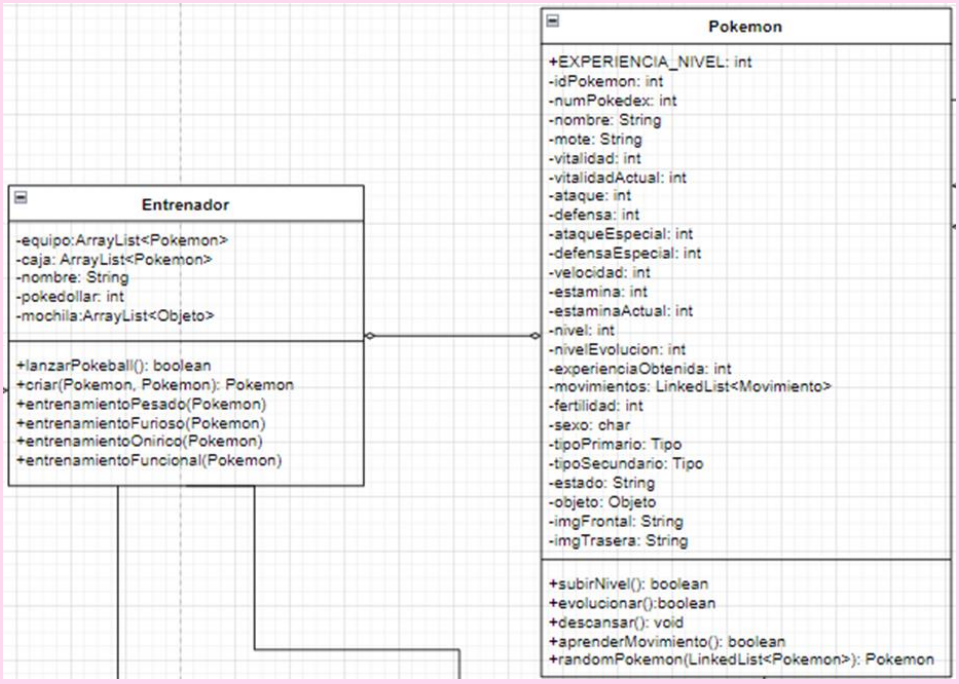


Umbrello para generar código 5

EJEMPLO PRÁCTICO

Dentro de los juegos Pokémon, existen dos objetos sobre los que crear Clases, Pokemon y Entrenador. Crea un diagrama que represente estas dos clases con los métodos que creas convenientes.

Solución: Diagrama de clases, de estas dos Clases, con sus posibles atributos y métodos:



4. GENERAR DIAGRAMAS DE CLASES A PARTIR DE CÓDIGO

Durante la fase final en las modificaciones de la aplicación de gestión de tiendas de ropa, el equipo se ha encontrado con algunas clases, métodos, interfaces, … que no estaban en el diseño, también se han tenido que crear algunos de estos durante el desarrollo de forma espontánea para facilitar el desarrollo del programa, es por ello, que los diagramas generados para el desarrollo de la aplicación están desactualizados, pero han de entregarse al cliente actualizados, y el equipo de soporte debe tenerlos actualizadas para la realización de posibles modificaciones en el futuro, por ello, es necesario conocer herramientas de ingeniería inversa para la generación de diagramas a partir del código ya existente.

Se utiliza el mismo software para realizar el proceso inverso: generar diagramas de clases a partir de código.

4.1. Visual Studio Ultimate

Para agregar clases de C# desde el código al diagrama de clases de UML en Visual Studio Ultimate, se arrastran dichas clases o espacios de nombres desde el Explorador de soluciones, desde un gráfico de dependencias o desde el Explorador de arquitectura al diagrama de clases de UML en cuestión. Las clases de las que dependen también aparecerán en el Explorador de modelos UML.

Para agregar clases desde código de programa a un modelo UML:

- Abrir un proyecto de C#.
- Agregar un diagrama de clases UML a la solución:
    - En el menú de Arquitectura, elegir nuevo diagrama. En el cuadro de diálogo, agregar nuevo diagrama, seleccionar diagrama de clases UML. Se creará un proyecto de modelado si no existiera ninguno.
- Abrir el Explorador de arquitectura:
    - En el menú arquitectura Ventanas, explorador de arquitectura.
- Arrastrar los espacios de nombres o tipos del Explorador de arquitectura a la superficie del diagrama de clases UML.

Para ver un tipo, expandir la vista de clases en la primera columna del Explorador de arquitectura y, a continuación, expandir un espacio de nombres en la columna siguiente. Se observarán los tipos en la tercera columna.

VÍDEO DE INTERÉS

En este videotutorial obtendrá una base para seguir investigando sobre cómo generar diagramas de clases a partir de código Java con NetBeans:

4.2. Netbeans

Se han detallado los procedimientos para realizar a través del IDE VisualStudio. Se procede ahora cómo hacerlo con el software NetBeans y el lenguaje de programación Java. Los diagramas de clases se pueden generar desde NetBeans, y con base en esos diagramas se puede generar el código para el programa. Para esto se debe instalar el plugin UML de acuerdo a las siguientes instrucciones:

- Lo primero es acceder a "Herramientas, Complementos"
- Luego seleccionar la Pestaña Configuración, clic en agregar.
- Y ahora en el nuevo cuadro de dialogo que aparecerá en Nombre poner UML y en URL: http://ea.ddns.com.br:8090/netbeans6.8/UML/catalog.xml
- Ahora ir a Plugins Disponibles y buscar la entrada con el Nombre UML, o con el nombre que se ha escrito en la ventana anterior, marcar e instalar.
- Con esto ya está instalado el plugin, ahora para probarlo crear un nuevo proyecto y en el menú para escoger se tiene la opción de UML.
- Luego ya se puede modelar, solo basta arrastrar las clases al espacio de trabajo y modificarlas con un par de clics, Java automáticamente genera todos los get & set de los campos que se añadan.
- Y ahora para generar el código es mucho más fácil solo se pulsa clic derecho a la clase y pulsar Generate Code, aparecerá un dialogo en donde seleccionar en qué proyecto y en que paquete se desea generar el código, se selecciona, se pulsa en aceptar y listo la clase ya está hecha.
- Una vez instalado el plugin se pueden crear los diagramas y de paso el código asociado a los mismos. A esto se le conoce como ingeniería inversa.

VÍDEO DE INTERÉS

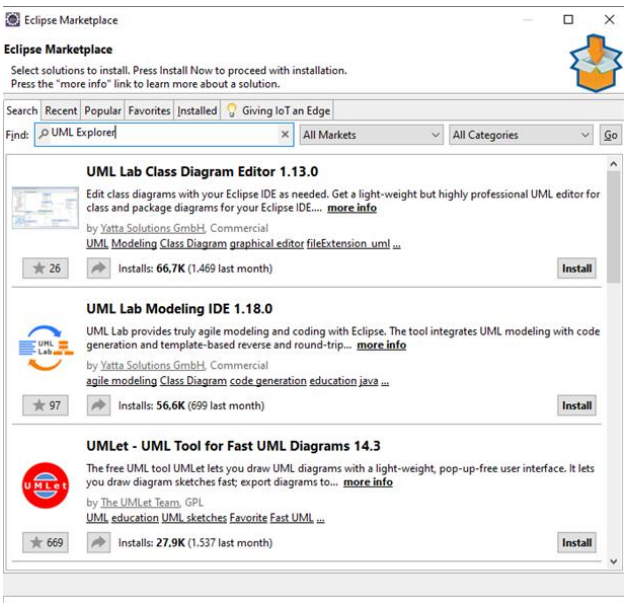
En este videotutorial puede ver el proceso anterior:

4.3. Eclipse – UML Lab Modeling

Eclipse tiene una extensión para poder generar el diagrama de clases de un proyecto que ya esté programado. Esta extensión se llama UML Lab Modeling, se puede descargar desde El Eclipse Marketplace.

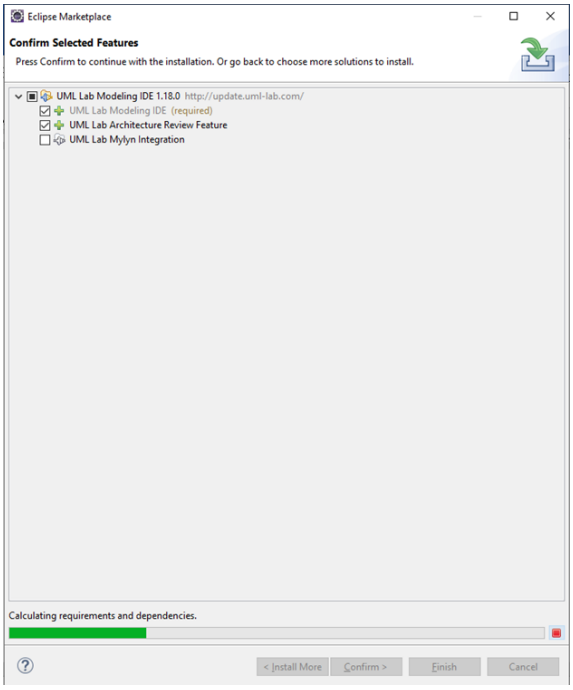
En los siguientes pasos se muestra como poder instalarla dentro de nuestro IDE.
Para generar diagrama de clases en un proyecto ya construido o que estamos construyendo en Eclipse, vamos a utilizar UML Lab Modeling IDE. Para ello instalaremos el siguiente plugin en Eclipse.

1. Nos iremos a Eclipse Marketplace, en el buscador ponemos UML Lab. Instalaremos el UML Lab Modeling IDE.



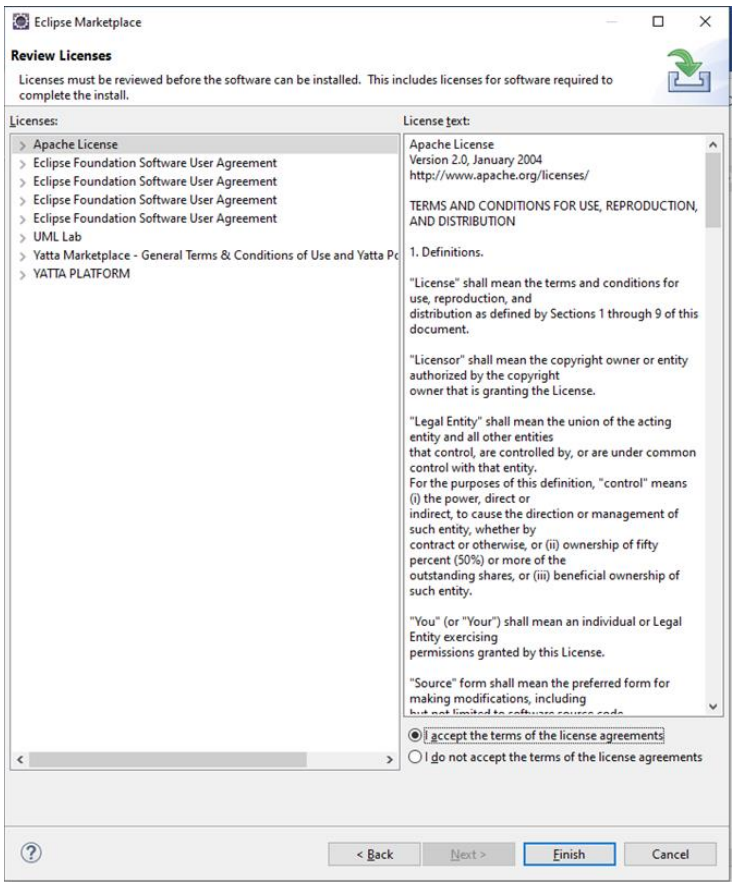
UML Lab Modeling IDE 1

2. En la siguiente ventana confirmamos:



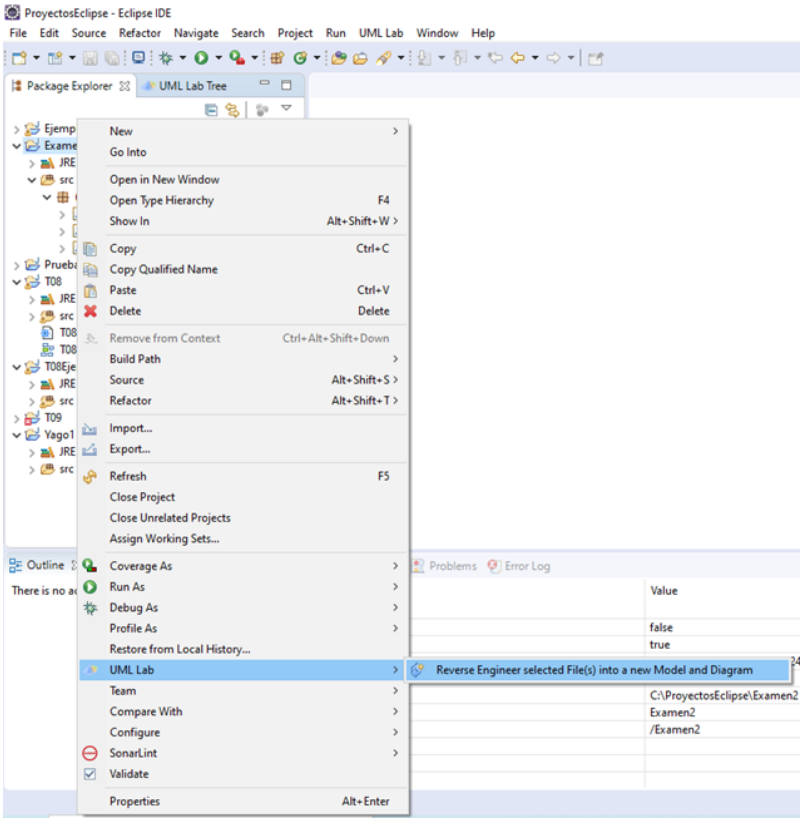
UML Lab Modeling IDE 2

3. Aceptamos los términos y Finalizamos.



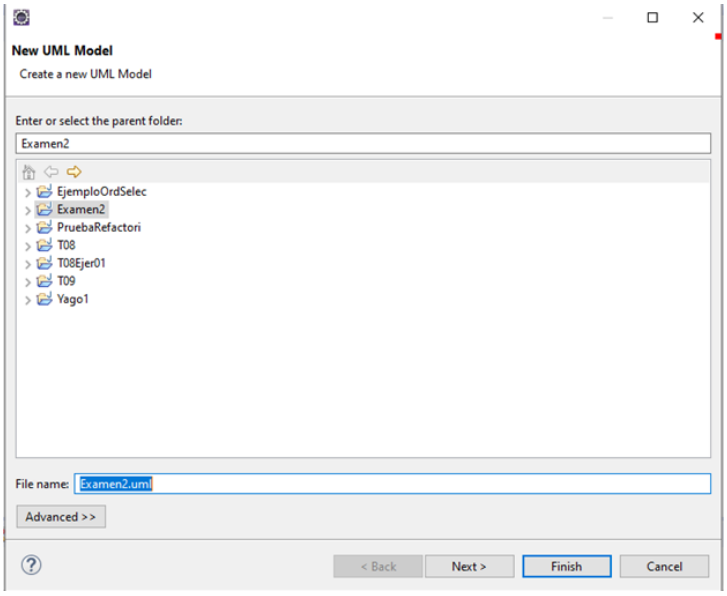
UML Lab Modeling IDE 3

4. Tenemos instalado el UML Lab, para usarlo, solo tenemos que dar encima del proyecto al botón derecho, seleccionar UML Lab -> Reverse…



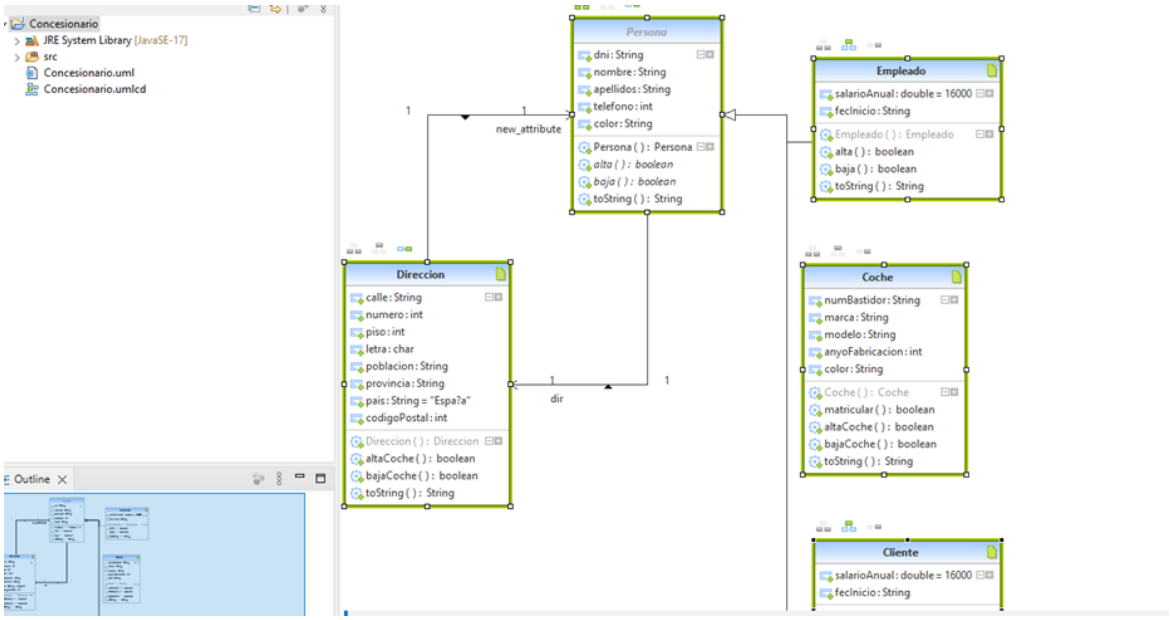
UML Lab Modeling IDE 4

5. Nos aparece el nombre que queremos poner al UML que generaran:



UML Lab Modeling IDE 5

6. Vemos un ejemplo de diagrama generado:



UML Lab Modeling IDE 6

VÍDEO DE INTERÉS

Generar diagrama de clase en Eclipse a partir de código:

RESUMEN FINAL

En esta unidad se pretende inculcar el diseño UML para el modelo de diagramas de clase. Una clase va a permitir crear un esqueleto a través de una abstracción del mundo real que tendrá un nombre, unos atributos y unos métodos. Estas 3 partes son tenidas en cuenta en el diseño UML. Normalmente, entre las clases de un software existen relaciones, que serán de herencia, agregación, asociación, instanciación o/y dependencia, según la funcionalidad de este. A través de una serie de herramientas como ArgoUML o StarUML va a ser posible realizar diseños UML de diagramas de clases. A su vez, existe la posibilidad, en algunas de ellas de generar código a partir de dichos diagramas y viceversa.