ACTIVIDAD 5 - ED

DIAGRAMAS E INGENIERÍA INVERSA

Francisco Jesús Díaz Martínez
IES AGUADULCE | 1B DAM

Parte 1: Diagramas estructurales

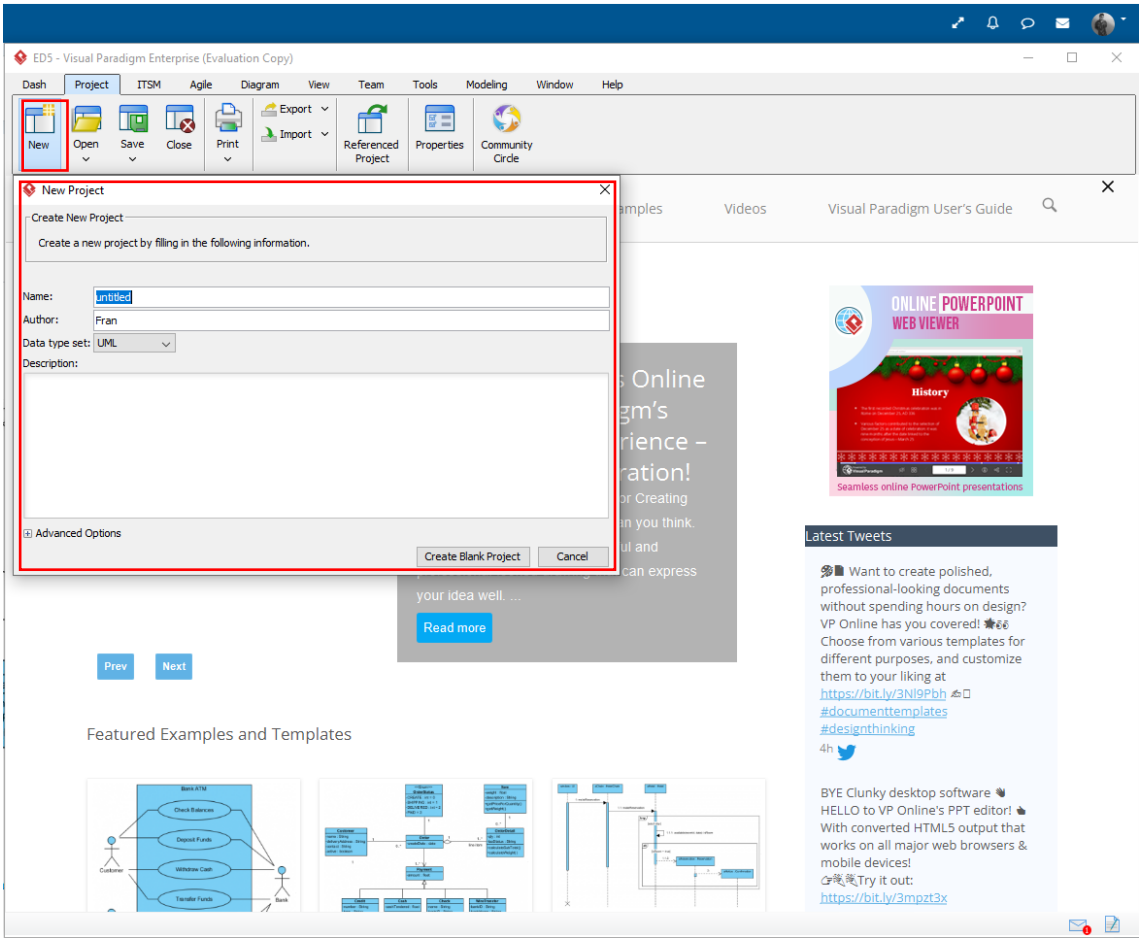
- **Ejercicio 1: Elabora el siguiente diagrama de clases mediante el programa Visual Paradigm. Debes crear el proyecto VP-UML.**

En la actividad nos proporcionan la imagen de un diagrama de clases que cuenta con 5 clases, cada una de ellas con sus correspondientes atributos y operadores.

Para la resolución de esta primera actividad deberemos de instalar el programa "**Visual Paradigm"** una herramienta de creación de diagramas, entre ellos de tipo UML.

La herramienta es de pago, pero podemos descargar una versión de prueba por 30 días.

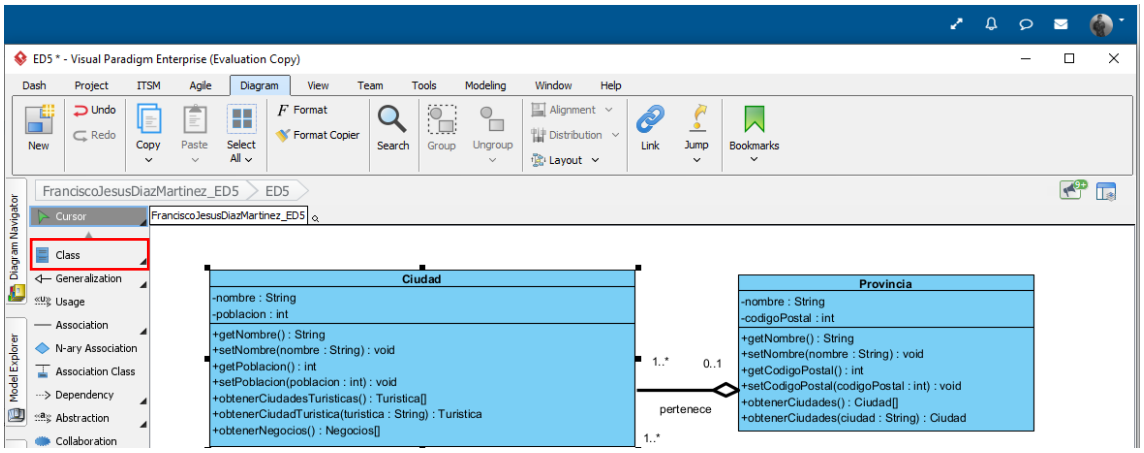
Una vez instalada deberemos de crear un nuevo proyecto UML y empezar con la creación de las respectivas clases y su contenido.



En esta primera captura podemos ver como se crea un proyecto, podemos indicar el nombre, el autor, el tipo de dato y una breve descripción como opciones básicas. Una vez configuradas las preferencias de nuestro proyecto podemos crearlo. A continuación, un menú con pestañas nos proporcionará plantillas predefinidas que podremos elegir según nuestras necesidades.

Una vez creado nuestro proyecto, deberemos de rellenar los campos de cada clase y establecer sus relaciones. En el ejemplo a seguir se dan varias situaciones. Nos centraremos en la confección de las relaciones entre las clases “ciudad” y “provincia”, así como la creación de las mismas.

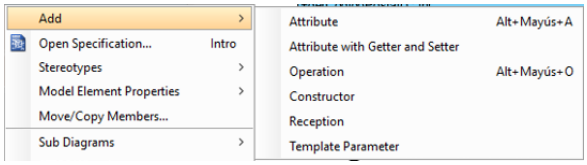
El proceso es sencillo, y lo describiré textualmente, aportando capturas de los apartados que considere más significativos.



En el rectángulo remarcado en la captura, podemos observar una herramienta llamada “Class” se trata del creador visual de las cajas azules que vemos en el diagrama. Pinchando en esta herramienta, solo deberemos de trazar con nuestro ratón el área que deseemos que ocupe nuestra clase.

Por supuesto, el campo estará vacío y deberemos de completarlo indicando el nombre de la clase, los atributos así como los getters-setters y operadores. Podemos rellenar esta tabla de dos formas, o bien haciendo clic derecho sobre ella, se abrirá un desplegable y añadiendo el parámetro que deseemos.

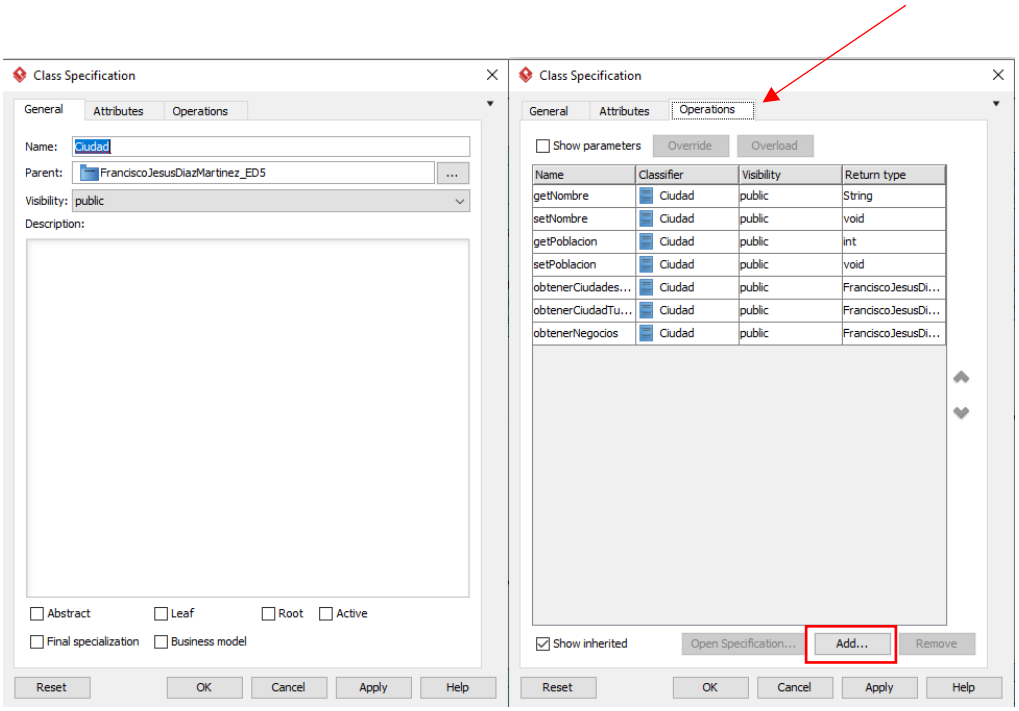
O bien pulsando “enter”, dónde se nos mostrará una ventana en la que podremos añadir los campos necesarios.



Esta captura corresponde a lo que se ha descrito primero, añadiendo los campos que necesitemos mediante este menú desplegable.

Personalmente lo he encontrado útil a la hora de crear las clases más sencillas, pues una vez que se usa el creador de “Attribute with Getter and Setter” los métodos se autocompletan con la descripción de los atributos.

En la siguiente página se muestran dos capturas de la segunda opción, en la que nos encontramos una ventana emergente para configurar nuestras nuevas clases.



Aquí podemos observar, un formulario en el que encontramos a la izquierda los nombres que informan de lo que debe de ir en ese campo.

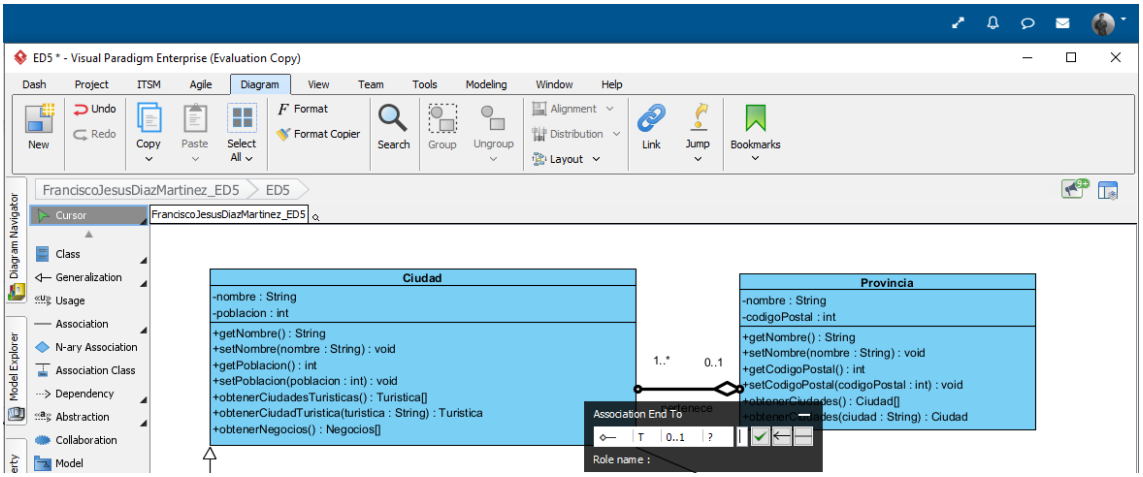
Y, por otro lado, en la parte derecha, en la pestaña correspondiente a los operadores, una lista con los operadores que hemos definido. Si quisiéramos añadir otro más solo deberíamos pulsar dónde se señala en el cuadro rojo, y se nos abriría otra ventana en la que deberíamos de definir los valores que se pueden observar en las columnas (name, classifier, visibility, etc.).

Con esto cubriríamos por completo, la creación de las clases y sus campos, obviamente debemos de tener conocimientos del lenguaje que vayamos a usar (en este caso JAVA) para completar correctamente las tablas.

Ahora solo nos queda por comentar la relación entre las clases, también poseemos herramientas para definirlas gráficamente, y se encuentran en la barra lateral izquierda del software.

Aquí, una captura de la sección que corresponde a las herramientas para definir la relación entre clases.

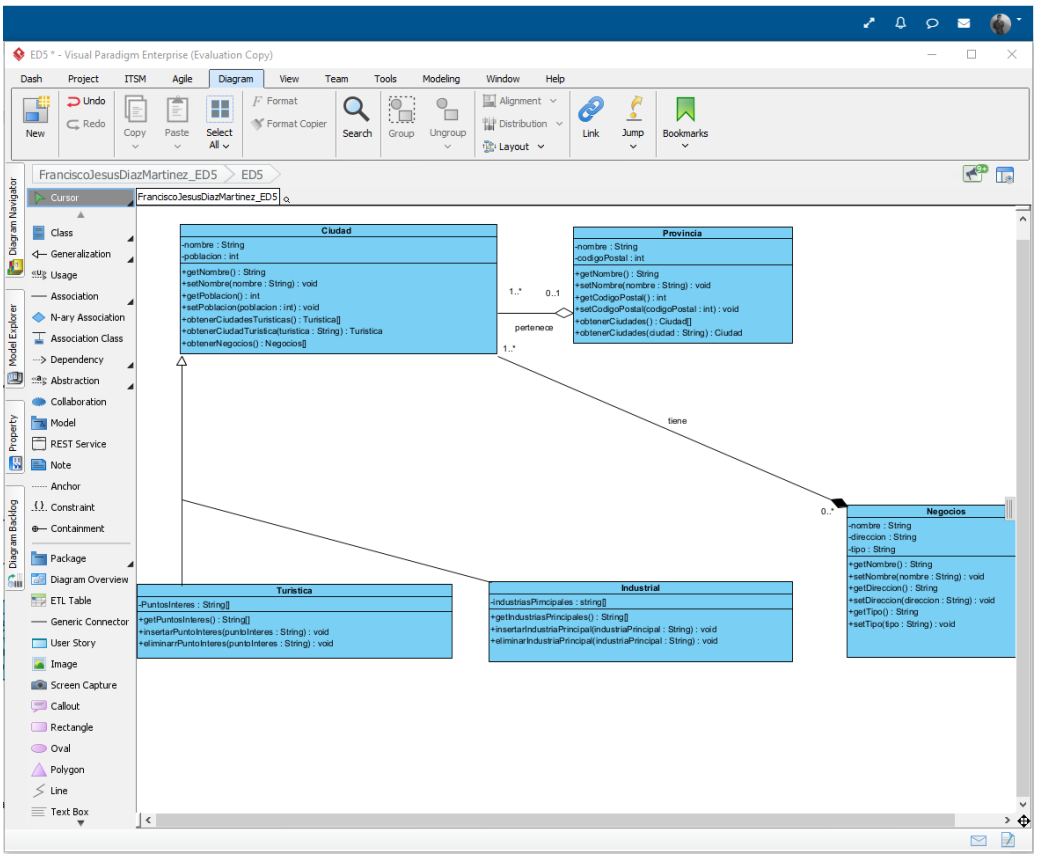
Por supuesto, como he comentado antes hay maneras diferentes de conseguir el mismo resultado, como pinchando en la parte superior derecha de las clases, para acceder a un menú contextual en el que definir la relación que existe entre ellas, aquí expondré solo una de estas opciones, la que se ha mostrado en la anterior captura.



Como he expuesto al comenzar la actividad, creo que esta relación es una de las más completas para poder explicar todas las acciones que hemos tomado. En este diagrama, nos han indicado que existe una relación de asociación entre estas clases. Para indicar esto, deberemos de seleccionar la herramienta “Association” en el panel izquierdo, y pinchar por orden las clases que intervienen en la relación.

Hecho esto, se creará la línea que se puede observar, y se nos mostrará un menú contextual en el que podremos indicar el tipo relación y cardinalidad (entre otros parámetros).

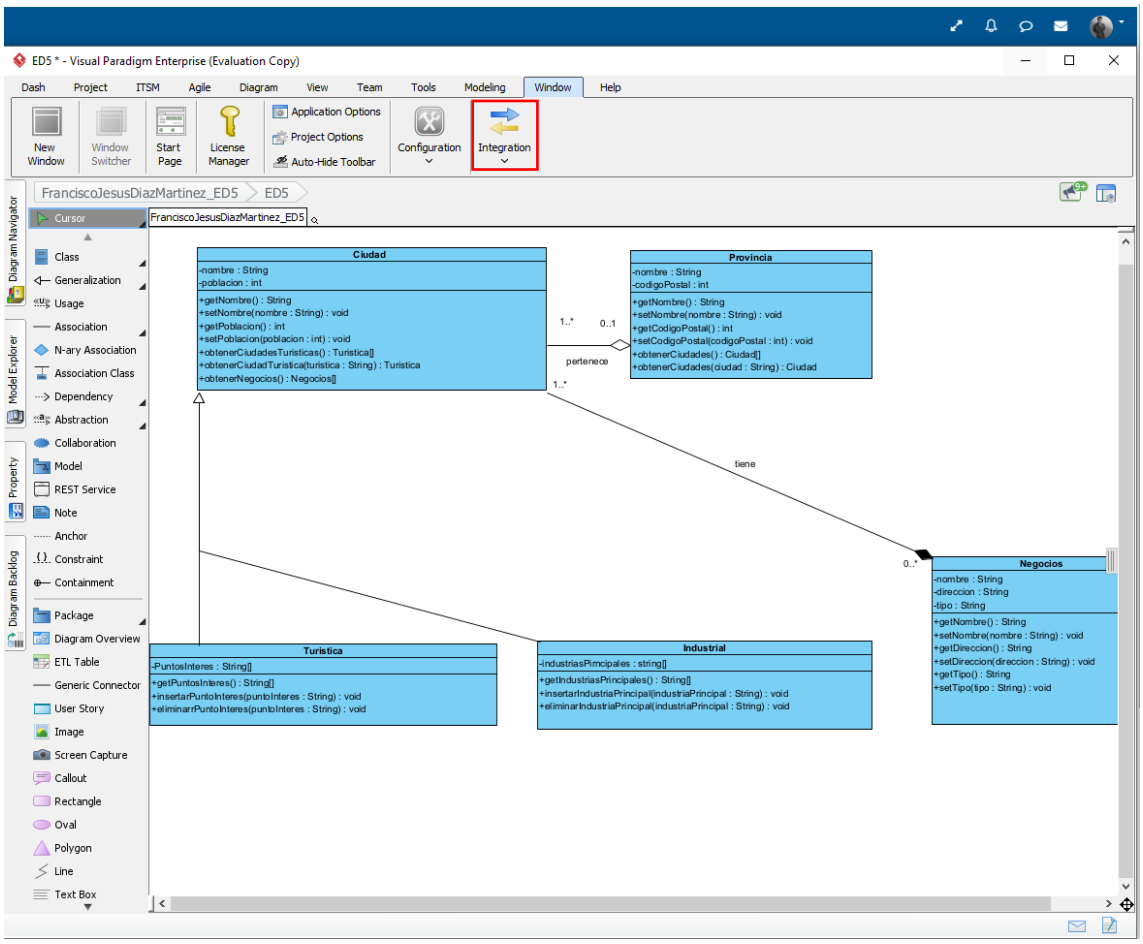
Deberemos de configurar ambos extremos, así como aprovechar la posibilidad de introducir textualmente la naturaleza de la relación.



Este es el resultado final, después de seguir las instrucciones comentadas previamente.

Ejercicio 2: Importación del proyecto VP-UML en un proyecto de NetBeans.

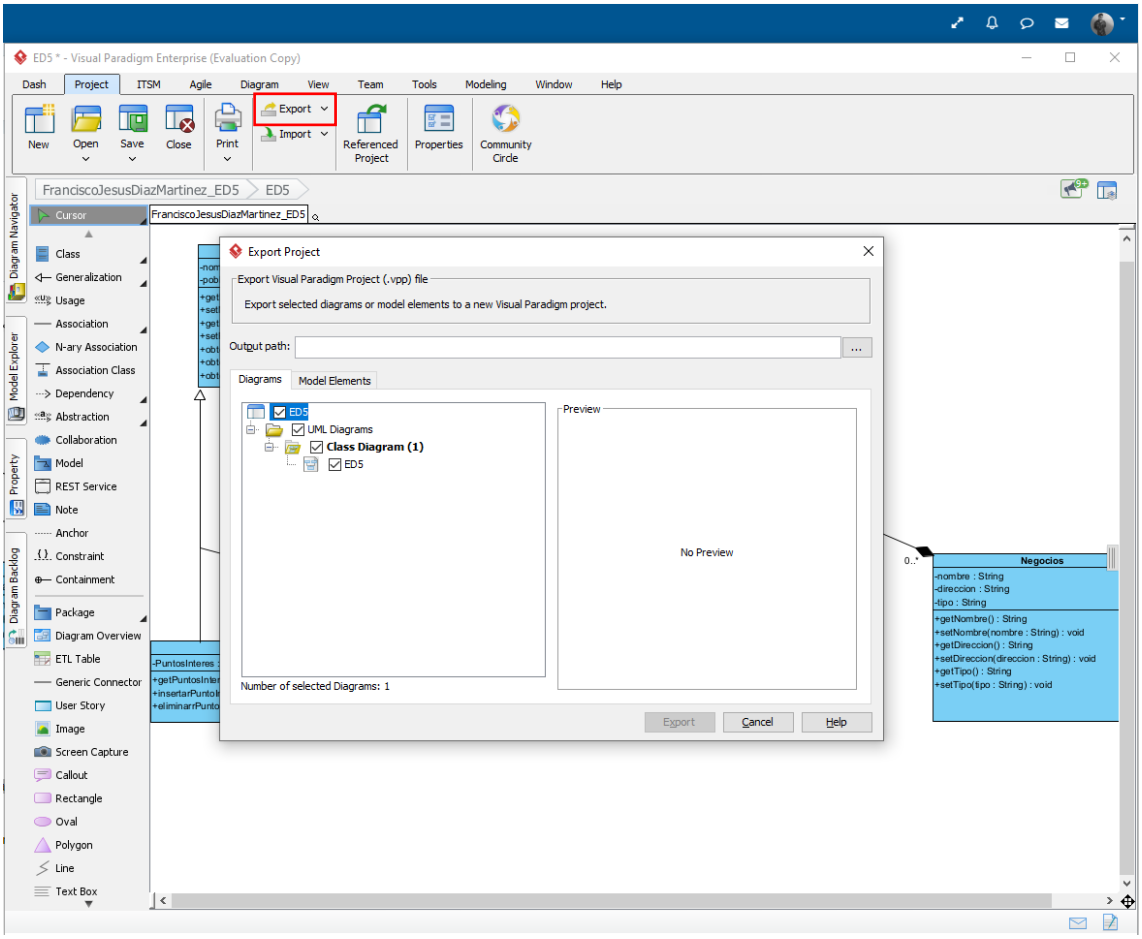
En este apartado deberemos de enlazar el software Visual Paradigm con el IDE NetBeans, para este fin, el primero de ellos tiene una función especifica que cumple con este objetivo.



En la pestaña “Window” deberemos de pulsar sobre “Integration”, se abríra un desplegable que muestra “IDE Integration”, al escoger esta opción se nos abrirá una ventana emergente en la que deberemos de seleccionar el IDE (en nuestro caso NetBeans) y su ruta de instalación. Una vez finalizado el proyecto ya tendremos ambos programas vinculados.

Como se puede apreciar en la captura anterior, ya aparece el add-on del software en nuestro IDE. Deberemos de activarlo para poder acceder a la función de importar el archivo .vpp de nuestro diagrama. En la siguiente captura mostramos como exportar el archivo que necesitamos.

Pulsamos en “Export”, seleccionamos el diagrama en cuestión e indicamos la ruta en la que queremos que se guarde el archivo.

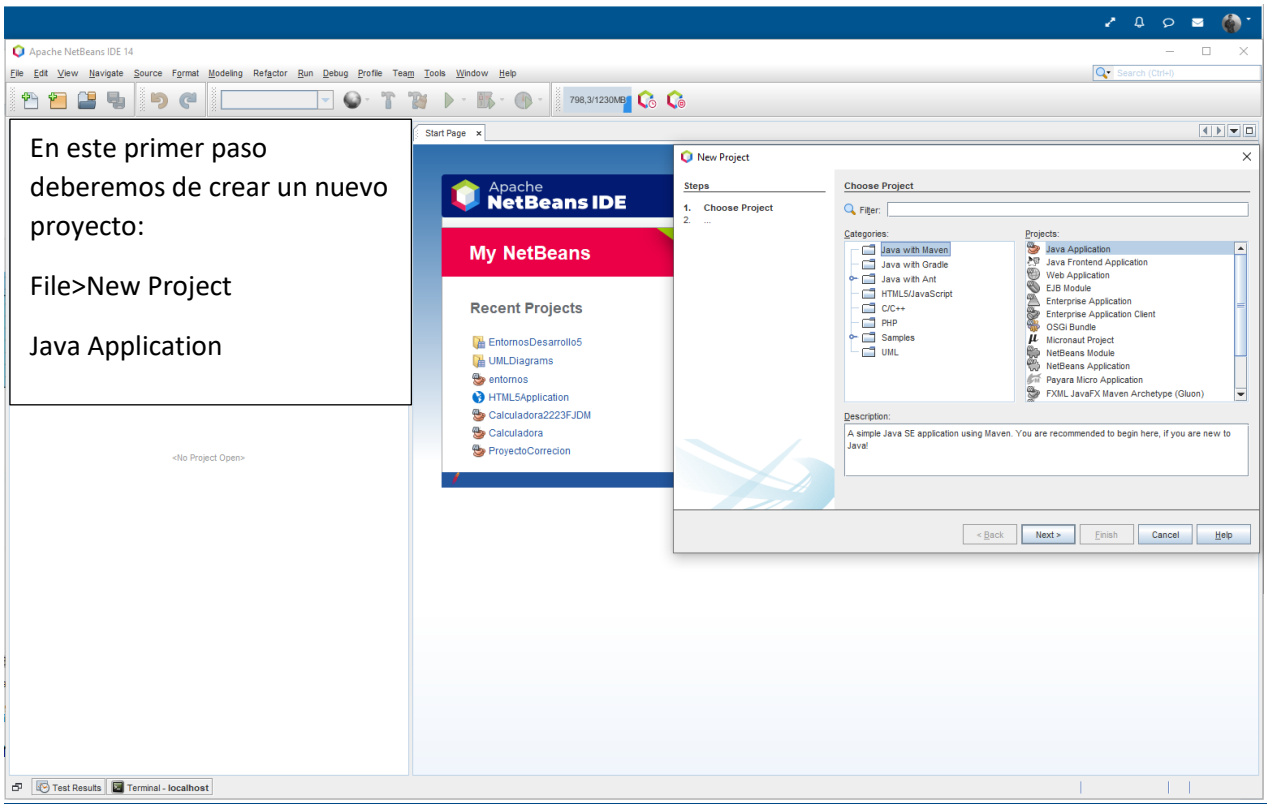


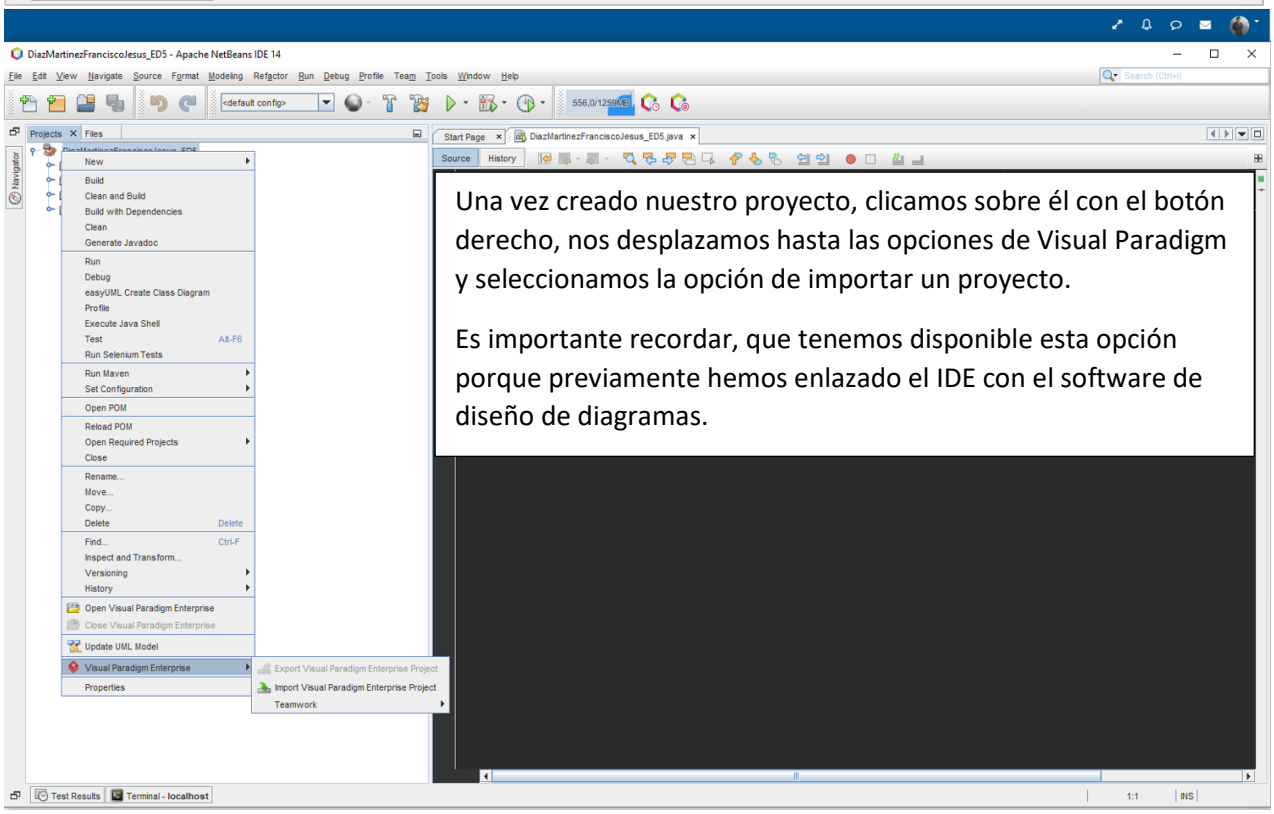
Una vez ya tenemos el archivo, debemos de continuar con el siguiente paso, debemos de importarlo a NetBeans.

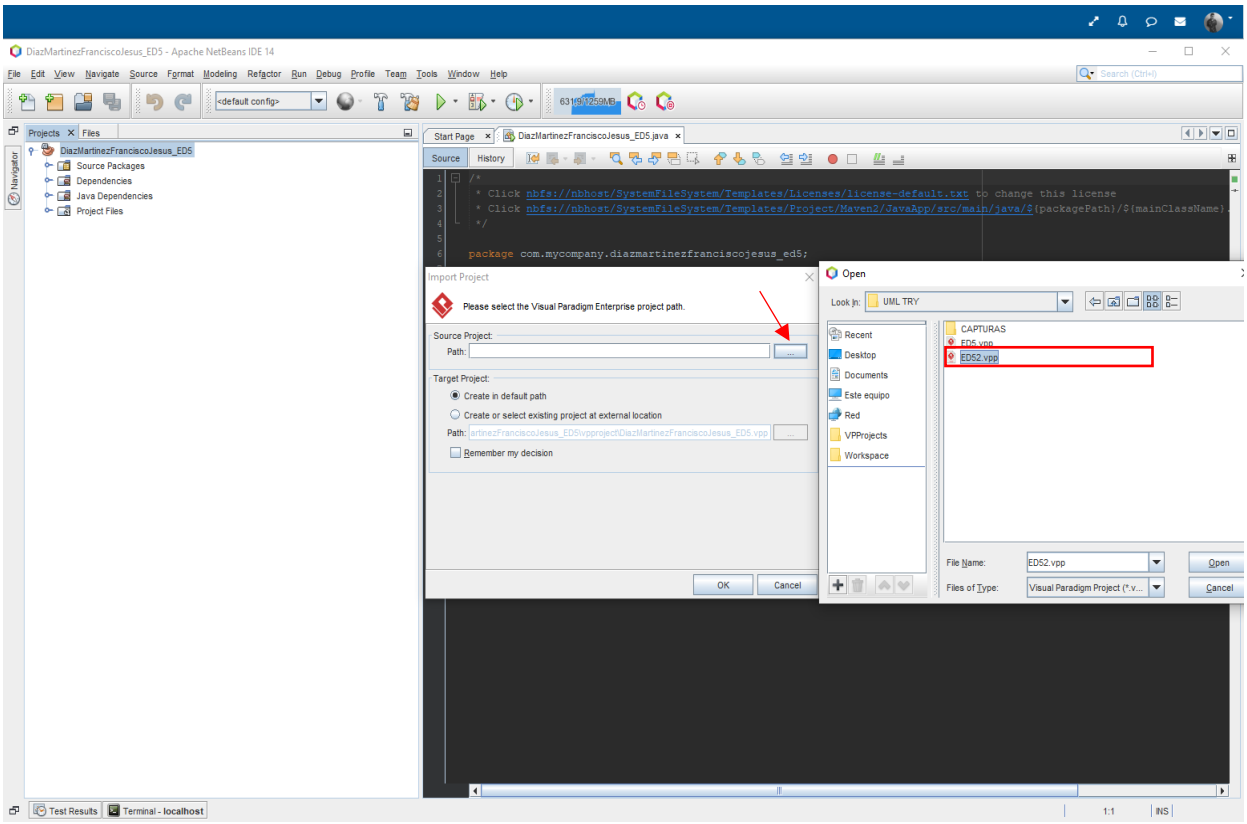
Como el lenguaje que hemos usado para el diagrama es JAVA es muy importante que tengamos esto en cuenta. Los pasos a seguir son:

- Crear un proyecto JAVA
- Una vez creado, hacer clic derecho sobre la raíz del proyecto
- Bajar en el desplegable hasta el add-on que de Visual Paradigm
- Seleccionarlo y pulsar sobre la opción de importar el proyecto VPP

A continuación, se ejemplificarán los pasos indicados mediante capturas en el entorno de NetBeans.


En este primer paso deberemos de crear un nuevo proyecto:
File>New Project
Java Application
Apache NetBeans IDE 14
New Project
Steps
Choose Project
My NetBeans
Recent Projects
Java Application
Next >
Finish
Cancel
Help


Una vez creado nuestro proyecto, clicamos sobre él con el botón derecho, nos desplazamos hasta las opciones de Visual Paradigm y seleccionamos la opción de importar un proyecto.
Es importante recordar, que tenemos disponible esta opción porque previamente hemos enlazado el IDE con el software de diseño de diagramas.
Visual Paradigm Enterprise
Import Visual Paradigm Enterprise Project

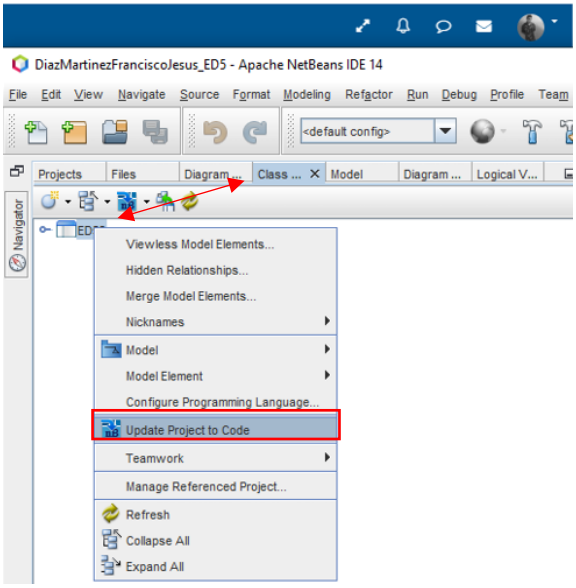


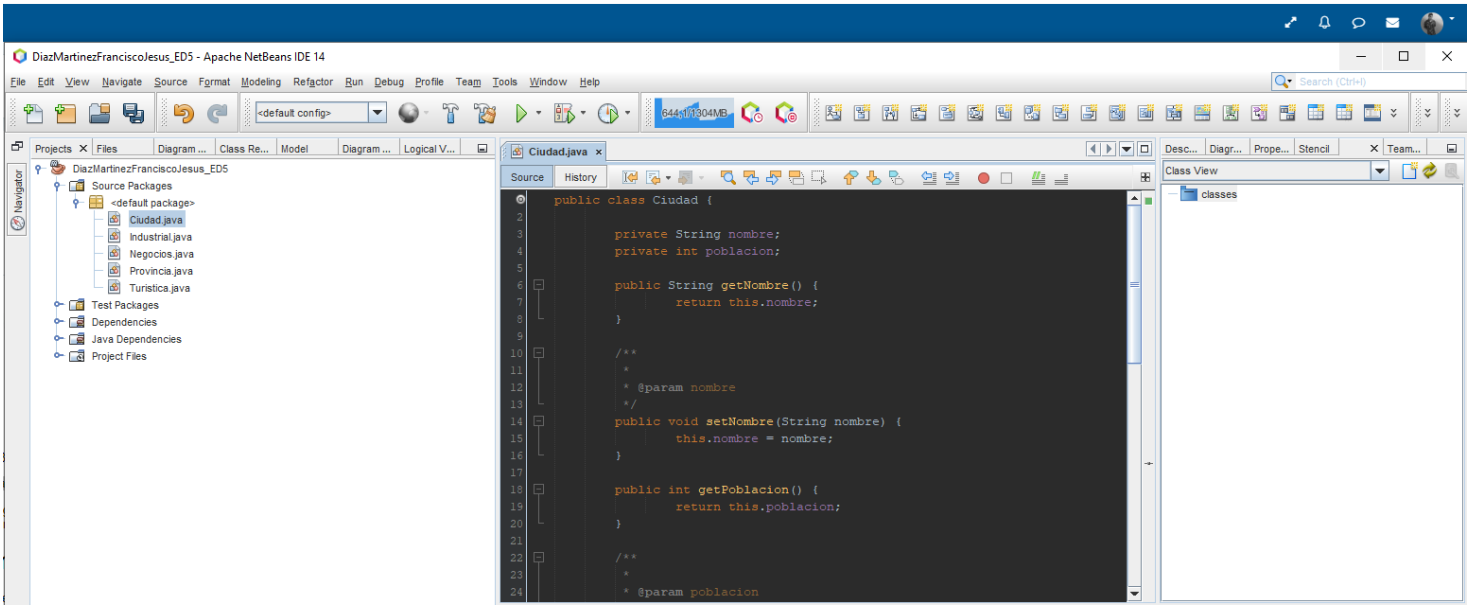
Por último, deberemos de escoger la ruta donde previamente habíamos guardado el proyecto VPP y seleccionarlo para que el IDE lo pueda importar.

Ejercicio 3: Generación del código a partir del diagrama de clases realizado.

Para culminar con el proceso, podemos comprobar si hemos definido bien el diagrama pasándolo a código. Esto es sencillo, una vez que tenemos el IDE preparado para ello.

Después de haber importado el archivo VPP, tan solo deberemos de dirigirnos a la pestaña “Class Repository” hacer clic derecho sobre el archivo que se muestra y seleccionar “Upgrade Project to code” como se muestra en la captura:





Aquí podemos ver, como una vez finalizado el proceso, tenemos las clases que habíamos definido en el diagrama, y también su contenido, como se muestra en el centro.

<u>Ejercicio 4:</u> Mediante el proceso de ingeniería inversa, genera el diagrama de clases a partir del siguiente proyecto Java

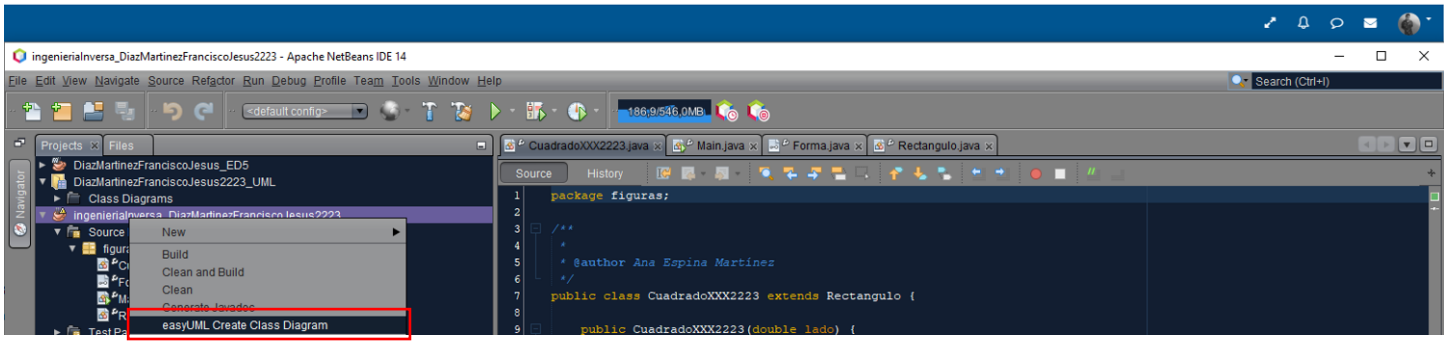
Antes de ir directos a la resolución del enunciado, hay algo que debemos de exponer. Para realizar el proceso de ingeniería inversa, disponemos del plugin EasyUML, deberemos de instalarlo en el IDE NetBeans para poder resolver lo que nos proponen.

En mi caso he tenido que instalarlo manualmente porque no estaba disponible en la base de datos de mide IDE, he descargado el repositorio proporcionado desde la tarea, y después mediante: **Tools>Plugins>Downloaded>Add Plugins**, he instalado y activado el paquete al completo.

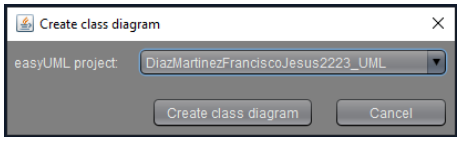
Hecho esto, deberemos de crear un proyecto UML para depositar la información del diagrama extraído del proyecto que se nos proporciona.

En las siguientes páginas explicaré mediante capturas el proceso más detalladamente, así como las modificaciones necesarias para que todo este optimizado.

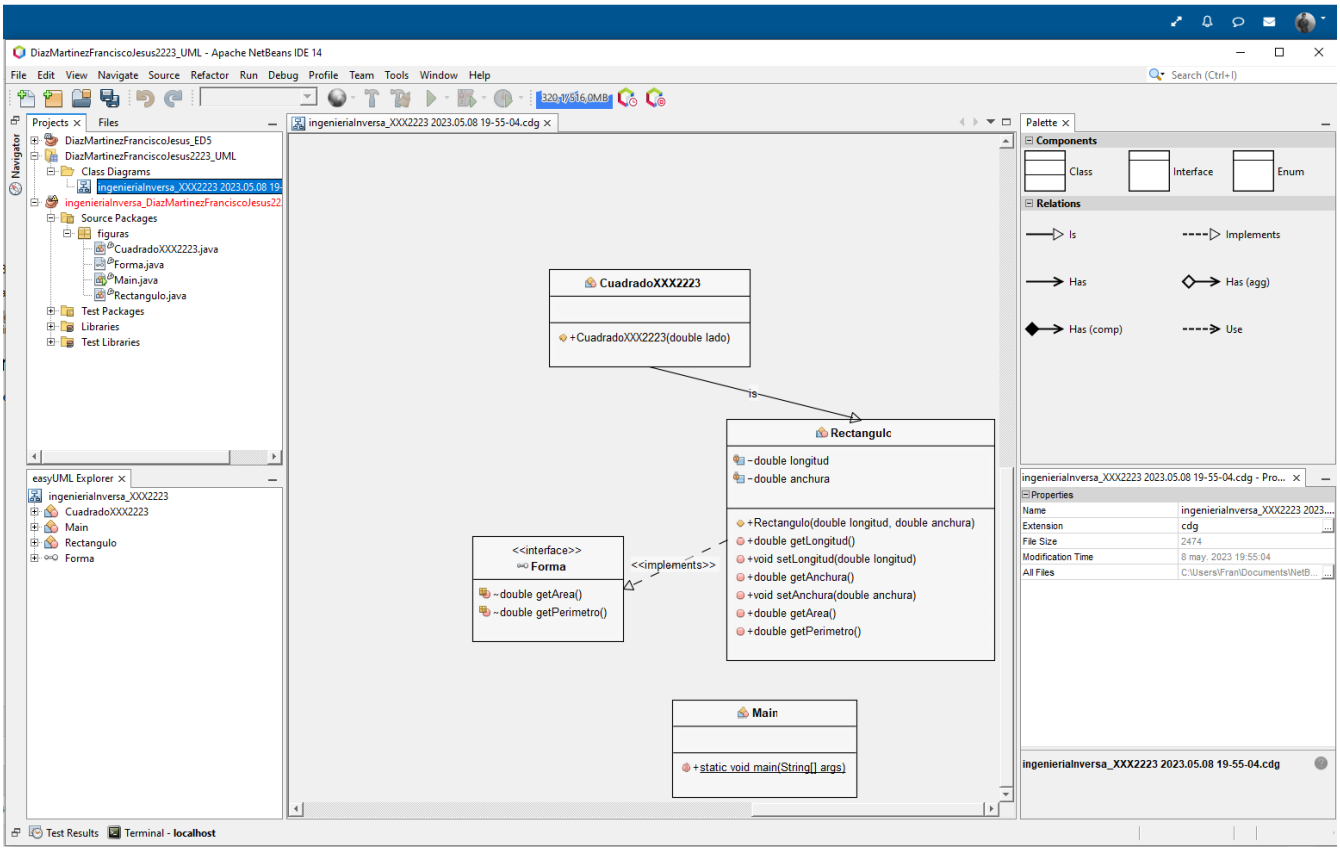
Obviaré la creación del proyecto UML ya que es algo que debemos de hacer con cierta soltura y algunos pasos como el de indicar la ruta del directorio donde debe de guardarse el proyecto, ya que no aportan información de valor a la resolución del ejercicio.



Aquí observamos como en el desplegable se muestra la opción de crear el diagrama de clases con la herramienta easyUML.



A continuación, indicamos el proyecto que usaremos para depositar la información del diagrama.



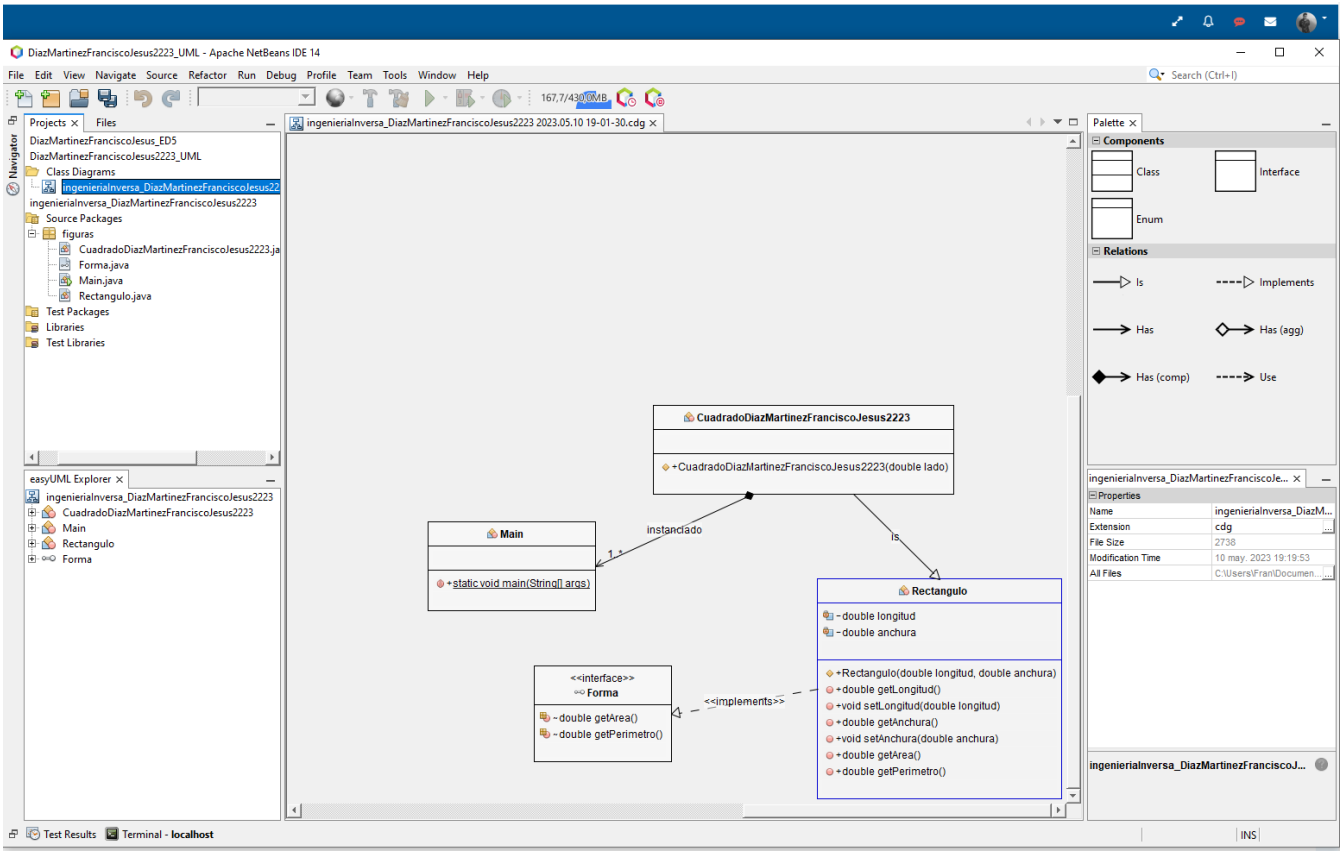
Al finalizar el proceso, esta es la primera versión que nos muestra el IDE. Como se puede ver automáticamente traza las relaciones, las etiqueta y define las clases junto a sus atributos y operadores.

Si prestamos atención a la parte superior izquierda, dónde se muestran los proyectos, se ha generado un archivo CDG. Ahora deberemos de revisar las clases del proyecto original para revisar si todo se ha implementado correctamente o faltan relaciones.

Después de revisar las clases proporcionadas, y de refactorizar el proyecto, he llegado a la conclusión de que podemos añadir una relación extra que EasyUML no ha tenido en cuenta, pero que si se ha tratado en la unidad.

Esta relación se trata de la creación y posesión de la instancia creada por “Main” respecto a “CuadradoXXX23” (nombre sin refactorizar), la clase main instancia la anterior clase para dar una salida por pantalla. Por lo tanto, podemos relacionarlas mediante una relación de composición.

He optado por una cardinalidad de 0 a 1, porque la clase de cuadrado puede existir sin la clase main, pero sin embargo main no puede funcionar sin instanciar a la clase cuadrado.



Ya hemos refactorizado nuestra clase, hemos ordenado el diagrama y hemos establecido la relación de composición entre las clases main y cuadrado.

Este sería el resultado final después de revisar las clases y ver las relaciones que tienen entre sí.

Podemos establecer que un cuadrado siempre se define por la clase rectángulo, que la implementación de la interfaz forma en la clase rectángulo es la que define el área y el perímetro y que la clase main instancia a la clase cuadrado para comprobar las medidas obtenidas a través de la clase rectángulo.

Parte 2: Diagramas de comportamiento

Ejercicio 5: Identifica el tipo de diagrama e interpreta lo que está representando sobre pedidos de pizza.

En el diagrama propuesto, mediante los elementos que podemos observar, nos indica que es un diagrama de estados.

- Diagrama de estados del proceso de pedido de una pizza:
  - Los estados por los que va a pasar el proceso son: pedido, elegir tipo de pizza, añadir ingredientes, elegir método de entrega, dirección de entrega, pago domicilio y pago recogida.
  - El estado inicial se activa cuando se realiza un **pedido**
  - Al realizar el pedido se solicita que se **indique el tipo de pizza** que quiere el cliente
  - A continuación, el cliente puede **añadir ingredientes** adicionales a su pizza
  - Completada la personalización, se debe de elegir el método de entrega
  - En este instante, se produce un **evento que bifurca los posibles estados** en el diagrama, Domicilio/Recogida
  - Si el cliente decide que se le entregue la pizza a domicilio, se le requerirá que indique una **dirección de entrega**, posteriormente se solicitará el **pago en el domicilio** y se llegará al **estado final.**
  - Por otro lado, si el cliente decide ir a **recoger la pizza**, pasará por el estado de **pagar la pizza en la recogida**, y se llegará al **estado final**.

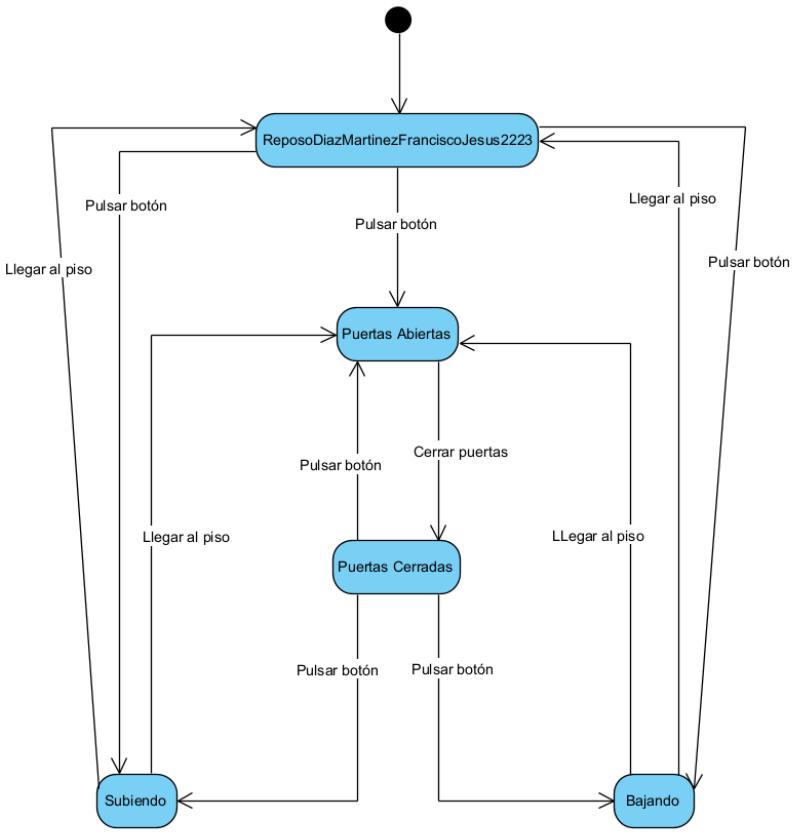
Ejercicio 6: Realiza el diagrama de estados para la siguiente operatoria de un ascensor:

El ascensor dispone de los siguientes estados: ReposoXXX2223, Subiendo, Bajando, Puertas abiertas y Puertas cerradas.

La posición inicial parte del estado Reposo, en el cual se mantiene hasta que se pulsa un botón, pasando al siguiente estado subiendo o bajando, dependiendo de la dirección del piso en el que se ha pulsado el botón o abriendo la puerta si se pulsa el botón.

Cuando llega al piso, vuelve a la posición de Reposo, y abre las puertas, cambiando a este estado. A continuación, cierra las puertas, cambiando a dicho estado, y vuelve a estar en Repos a la espera de que se pulse un botón.

De este enunciado extraemos que el ascensor tiene un esta inicial, pero no un estado final. También extraemos que tendremos tres acciones entre estados: Pulsar botón, llegar al piso y cerrar puertas, a continuación el diagrama correspondiente al planteamiento del ejercicio.


ReposoDiazMartinezFranciscoJesus2223
Pulsar botón
Pulsar botón
Llegar al piso
Llegar al piso
Pulsar botón
Puertas Abiertas
Pulsar botón
Cerrar puertas
Llegar al piso
LLegar al piso
Puertas Cerradas
Pulsar botón
Pulsar botón
Subiendo
Bajando