ENTORNOS DE DESARROLLO

ciclos formativos de grado superior de
desarrollo de aplicaciones multiplataforma

CAPÍTULO 5

Carlos Casado Iglesias

Ra-Ma®

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

1. INTRODUCCIÓN A UML

Para realizar labores de diseño de software, es vital realizar modelados o diagramas representando gráficamente la estructura de la aplicación y su funcionalidad, de una manera fácil de entender y rápida de crear. Antiguamente, los diagramas de cada diseñador eran únicos, ya que, salvo un par de diagramas conocidos por todos (como los diagramas de flujo o los diagramas de entidad-relación), cada diseñador o analista realizaba los diagramas de la manera que mejor los entendía, por lo que podría suponer un problema si varias personas tenían que entender los diagramas que uno o varios diseñadores les presentaban para definir el software.

Con el fin de evitar ese problema se creó **UML (*Unified Modeling Language*)**, un conjunto unificado de estándares para las diferentes necesidades y usos que un diseñador pudiera tener a la hora de plantear una representación gráfica de un programa. Son diagramas de propósito general que se presuponen conocidos por todos, con unas técnicas de notación conocidas, de modo que cualquiera pueda crear diagramas entendibles por todos.

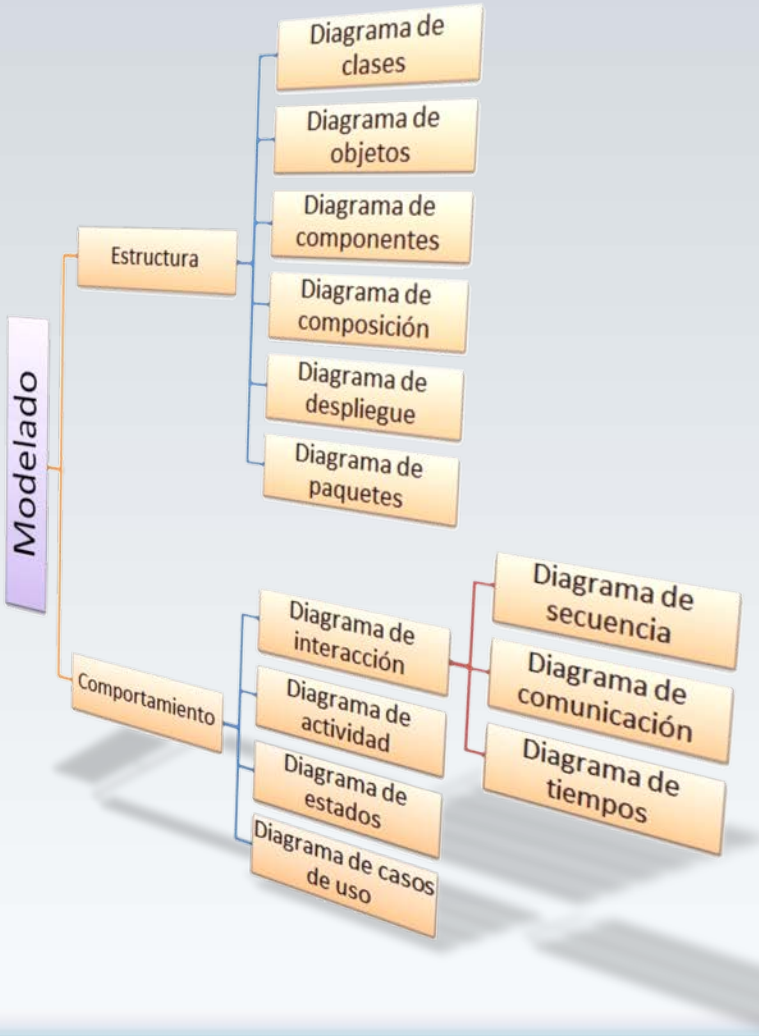
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

1. INTRODUCCIÓN A UML

- **UML 2.0**

En la versión 2.0 de UML se definió un completo árbol de superestructuras donde se relacionaban y complementaban todos los tipos de diagramas UML a la hora de definir un software por completo.

Teniendo presente la superestructura de diagramas de UML, podríamos pensar que la tarea de definir y realizar dichos diagramas (y que por supuesto sean coherentes entre sí) sería una tarea abrumadora, pero no es necesario ni se suelen realizar todos los diagramas para modelar un software, por norma general se utilizan solo una serie de diagramas para modelarlo, lo más clásico sería la tríada diagrama de clases, de secuencia y de casos de uso.

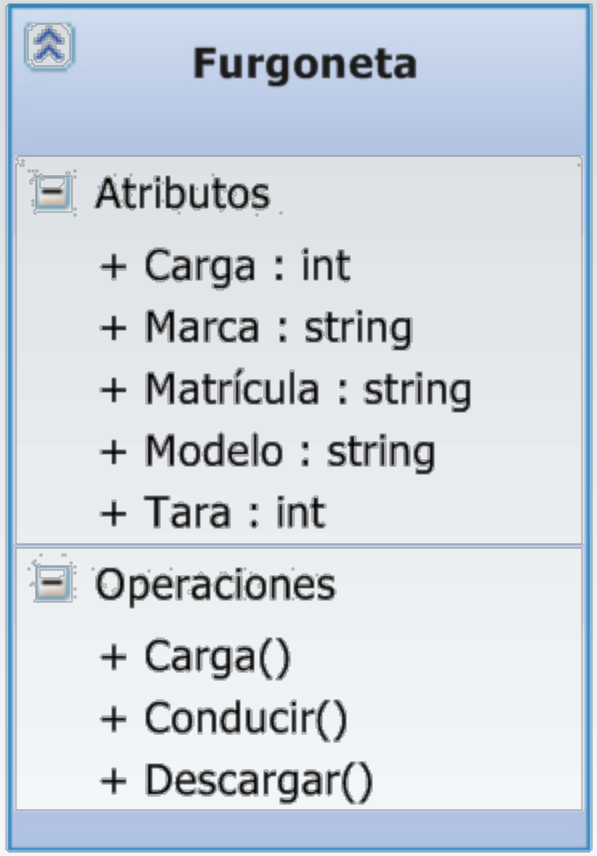


DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

Cada bloque dentro del diagrama representa a una clase; una clase, como sabemos, dispone de unos atributos y de unos métodos asociados. Representaríamos las clases como cajas en donde escribiríamos sus atributos y sus métodos.

Las clases representan a nuestros objetos, los atributos definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.



DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

Estos diagramas son particularmente útiles en el desarrollo de la capa del modelo, para mapear de un modo gráfico las entidades y sus relaciones en nuestra aplicación, ese tipo de diagramas tienen la particularidad de que no muestran las acciones que se realizan sobre cada objeto, ya que solo representan la entidad y se encuentran separadas las acciones de los datos, por lo que nuestro diagrama solo mostraría las clases y sus atributos. Dependiendo del lenguaje utilizado, las clases sí que tendrían operaciones, como por ejemplo los métodos set y get utilizados para obtener y establecer los atributos de los objetos siempre y cuando los atributos estén encapsulados. En C# realizaríamos esa operación definiendo los atributos como propiedades.

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

➢ NOTACIÓN

En primer lugar, debemos percatarnos de que las clases se definen mediante cuadrados, y dichos cuadrados se dividen en tres segmentos horizontales.

Primero tendríamos el nombre de la clase y por tanto de nuestro objeto, seguidamente irían los atributos y, al final, los métodos de la clase. Es importante resaltar que si alguna clase no posee atributos propios o métodos propios, se deberá seguir dibujando el segmento que le corresponde, incluso si está vacío.

El carácter que precede al nombre del atributo o método se corresponde con su grado de comunicación.

- \+ **Público**, el elemento será visible tanto desde dentro como desde fuera de la clase.
- \- **Privado**, el elemento de la clase solo será visible desde la propia clase.
- \# **Protegido**, el elemento no será accesible desde fuera de la clase, pero podrá ser manipulado por los demás métodos de la clase o de las subclases.

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

En el diagrama de clases, además de dibujar y representar las clases con sus atributos y métodos, también se representan las relaciones.

Las relaciones se representan con flechas con una forma determinada, y, además, poseen una cardinalidad. La cardinalidad es un número o símbolo que representa al número de elementos de cada clase en cada relación, pudiendo usar el comodín "*" para definir un número indeterminado de elementos.

Cardinalidad	Significado
1	Uno y solo uno
0..1	Cero o uno
X..Y	Desde X hasta Y
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios

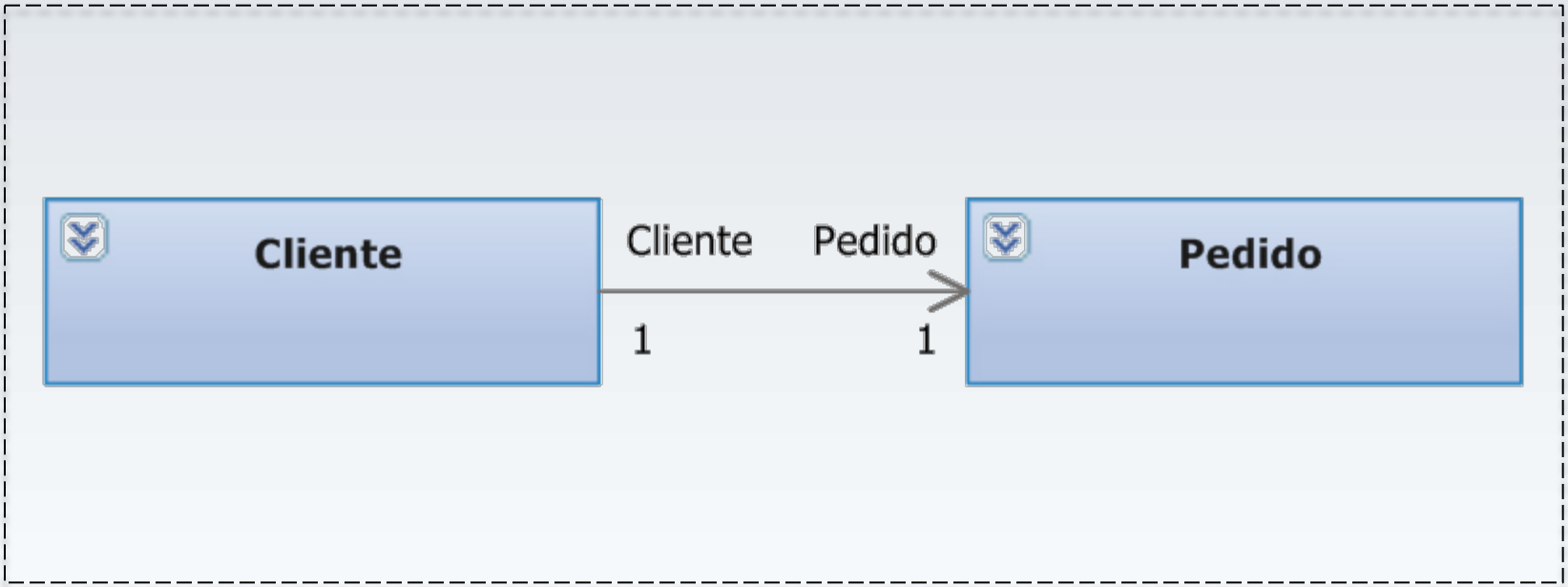
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

- **RELACIONES**

Asociación

La asociación es la más básica de las relaciones, no tiene un tipo definido y puede ser tanto una composición como una agregación, además una asociación podría implicar únicamente un uso del objeto asociado. Se representan mediante una flecha simple que también puede tener una cardinalidad.



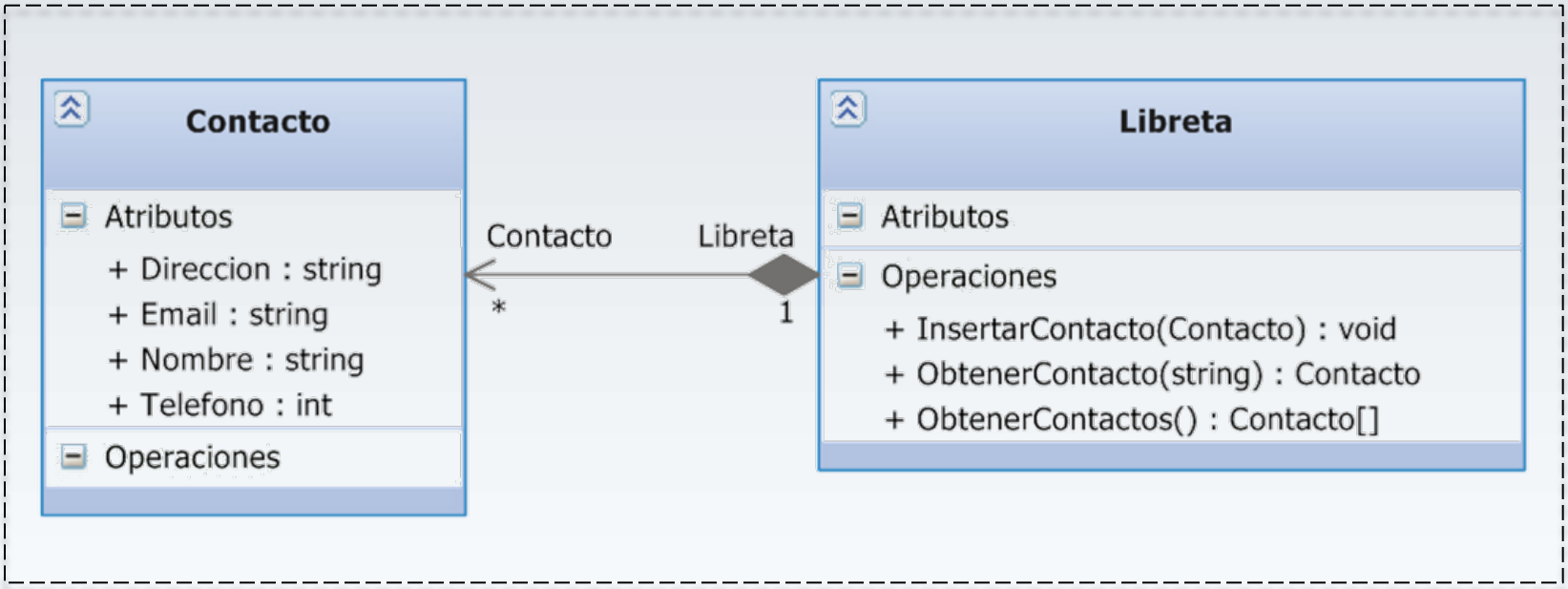
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

1. DISEÑO DE CLASES EN UML

- **RELACIONES**

Composición

La composición define los componentes de los que se compone otra clase, se define además que la clase que contiene la composición no tiene sentido de existencia si la(s) agregada(s) desaparece(n). Mediante esta relación, denotada por una flecha con un rombo relleno en una de sus puntas, se indica la presencia de las clases origen en la clase destino, donde la clase destino es apuntada por el rombo de la relación.



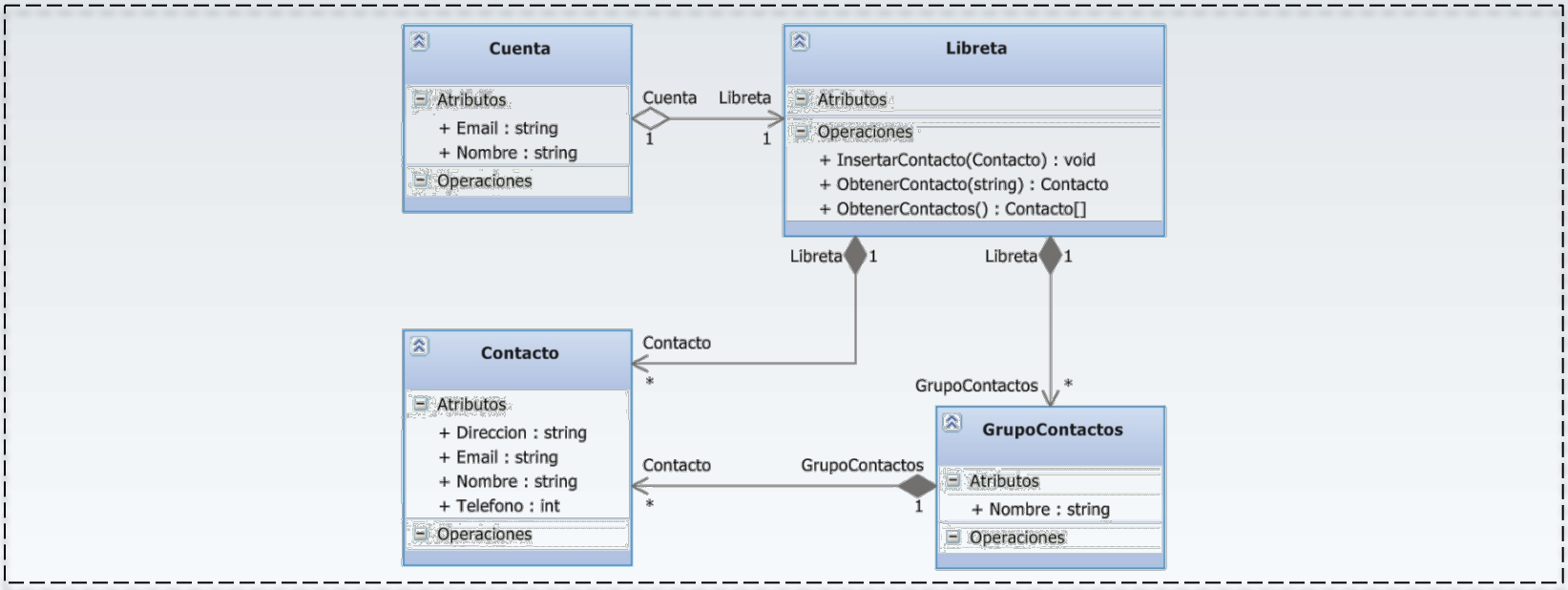
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

1. DISEÑO DE CLASES EN UML

- **RELACIONES**

Agregación

Nuestra libreta sigue complicándose, ya que ahora se define la posibilidad de tener una cuenta de usuario para acceder a nuestra libreta de contactos. Las cuentas tendrán un nombre de cuenta y un email, y podrán acceder a la libreta de direcciones. Para representar la relación de esta nueva clase, vamos a utilizar una agregación.



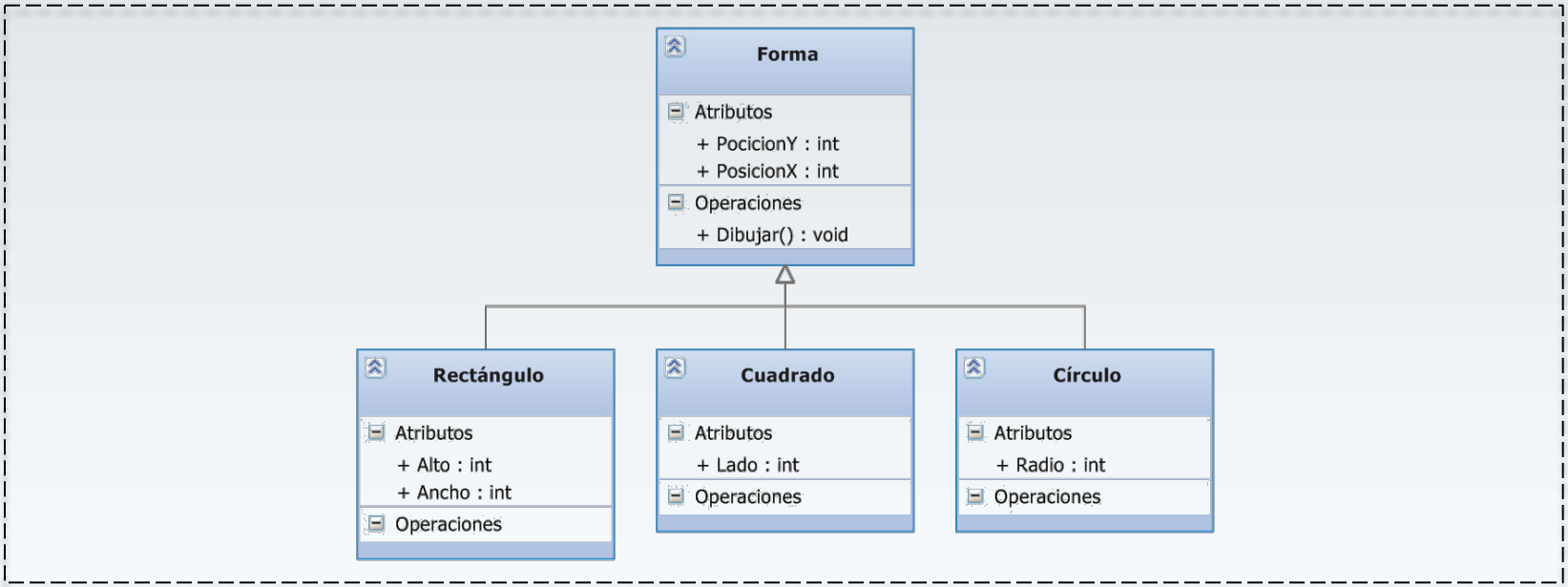
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

2. DISEÑO DE CLASES EN UML

- **RELACIONES**

Herencia

La herencia es un modo de representar clases y subclases, es decir, clases más específicas de una general, se representan mediante una flecha con una punta triangular vacía.



DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

➢ HERRAMIENTA DE MODELADO DE VISUAL STUDIO

Integrado en nuestro entorno de desarrollo, tenemos una potente herramienta **CASE (*Computer Aided Software Engineering*)** que utilizaremos para crear los diagramas de clases y posteriormente usaremos para modelar diversos tipos de diagramas **UML** diferentes.

En cualquier proyecto de **Visual Studio** podemos agregar un diagrama como se haría con cualquier otro componente, pero podemos crear en su defecto un proyecto específico para crear los diferentes modelados que necesitemos.

Para aprender el funcionamiento de la herramienta de modelado utilizaremos un proyecto nuevo de modelado. Para ello, iremos a la entrada de menú **Arquitectura > Nuevo Diagrama** para que nos aparezca el cuadro de diálogo de creación de diagramas. Elegiremos la plantilla **Diagrama de clases UML** y haremos clic en **Aceptar**. Nos creará una pantalla en blanco con las instrucciones necesarias para empezar a trabajar.

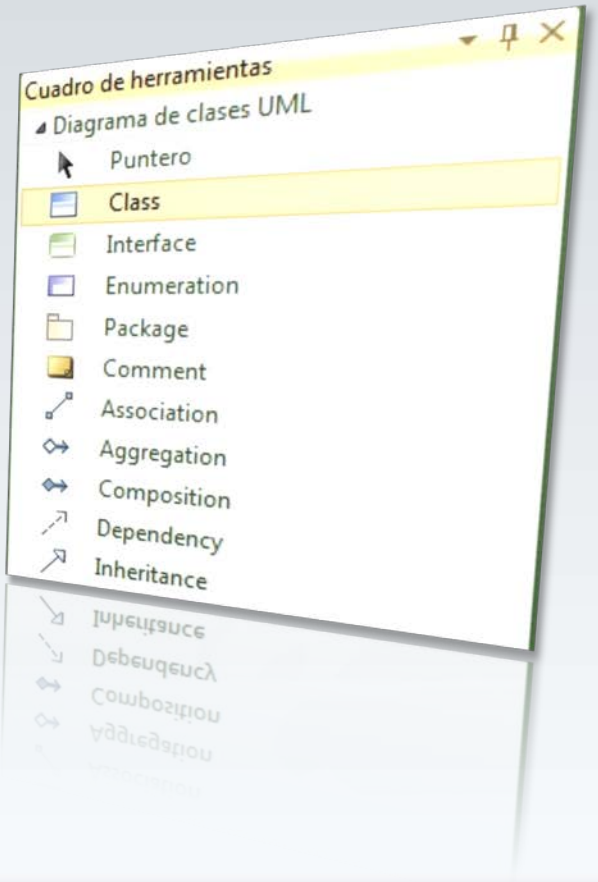
DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

- **HERRAMIENTA DE MODELADO DE VISUAL STUDIO**

Podemos añadir cualquier herramienta haciendo clic primeramente en el elemento del cuadro de herramientas y posteriormente haciendo clic en el cuadro de trabajo para agregar el elemento seleccionado.

Las relaciones utilizan el mismo proceso con la peculiaridad de que se tienen que arrastrar desde un elemento al elemento que queremos relacionar.



DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

➢ GENERAR CÓDIGO A PARTIR DE DIAGRAMAS DE CLASES

Esta funcionalidad nos la ofrece la herramienta de modelado, para ello crearemos un nuevo proyecto de aplicación de consola que llamaremos *ClasesyCodigo*. Nos generará la plantilla básica con una clase *program* y su método *main*. Esa clase no la vamos a tocar, en su lugar vamos a ir añadiendo las clases de nuestro programa, tal y como haríamos normalmente, pero en vez de crear las clases según el método tradicional lo vamos a hacer mediante un diagrama de clases, por ello, una vez creado el proyecto, haremos clic con el botón derecho en él y elegiremos la opción **Agregar > Nuevo elemento**. Elegiremos la plantilla diagrama de clases y nos aparecerá la misma vista de dibujo que hemos visto hasta ahora.

Vamos a modelar una aplicación muy sencilla, donde tendremos solamente dos clases: una clase **Operario** y una clase **Trabajos**, donde representamos los trabajos que puede desempeñar un operario. Primero vamos a crear nuestra clase Operario, para ello iremos al cuadro de herramientas del diagrama y arrastraremos el elemento "**Clase**" hasta el cuadro de dibujo.

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

- **GENERAR CÓDIGO A PARTIR DE DIAGRAMAS DE CLASES**

Según vayamos añadiendo elementos al diagrama la herramienta de modelado irá creando el código asociado. De éste modo y de forma muy rápida podemos tener la estructura de clases de nuestro programa directamente desde la etapa de diseño.

Además de ser una manera rápida de crear una estructura de clases, tenemos la ventaja de ver gráficamente la estructura y obtener una mejor visión sobre lo que estamos haciendo.

No hay que olvidar también que facilitaría la tarea del diseñador o del analista ya que con el propio diagrama de clases podría tener también la estructura de clases para que el programador pueda implementarla sin que se produzcan errores o confusiones. Los cambios realizados directamente en el código también se ven representados en el diagrama.

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

➢ INGENIERÍA INVERSA

Visual Studio nos permite realizar una **ingeniería inversa** de un proyecto con su herramienta de modelado, pero, ¿qué es la ingeniería inversa en el contexto de los diagramas de clase? La respuesta es bien sencilla: consiste en obtener de forma automática el diagrama de clases de un proyecto mediante su código.

El procedimiento de esta poderosa herramienta es muy sencillo, lo único que tenemos que hacer es seleccionar un proyecto de los que hemos realizado hasta ahora, donde hayamos utilizado en código relaciones de asociación o herencia. Una vez abierto, haremos clic con el botón derecho en el proyecto y elegiremos la opción **Ver diagrama de clases**, automáticamente, Visual Studio nos creará un diagrama con las clases utilizadas en el proyecto, aunque sin relacionar, nosotros solo tendremos que establecer correctamente las relaciones entre las clases para dejarlo correctamente.

Es también un dato interesante la posibilidad de ver la **clase base** de una clase solo tenemos que hacer clic con el botón derecho en la clase de la que queremos sacar la **derivada** y elegir la opción **Mostrar clase base**, automáticamente, nos creará y relacionará la clase base en nuestro diagrama de clases.

DISEÑO ORIENTADO A OBJETOS: DIAGRAMAS DE CLASE

3. HERRAMIENTAS

- **UMLPAD**

UMLPad es una herramienta muy sencilla y ligera que nos permite crear diagramas de clases. Es una herramienta externa y portable y su manejo es muy sencillo, muy similar al de la herramienta de modelado de Visual Studio.