Uso de *argoUML* para diseño de aplicaciones

Antes de codificar una aplicación, es necesario sentarse frente al bloc de notas, en lugar de frente al ordenador. Realizar el análisis y el diseño de aplicaciones es una tarea básica para realizar antes de la codificación, pese a la tendencia de todos los programadores (especialmente, los inexpertos), a sentarse a programar.

El uso de herramientas como *NClass* (editor de diagramas), permite realizar un diseño de la aplicación empleando UML, de manera que se puedan generar las clases del diagrama dibujado automáticamente. Así, el diagrama de clases UML no es sólo una ayuda al diseño de aplicaciones, sino que se convierte en un aportación interesante a la programación, lo cuál puede ser un incentivo a tener en cuenta.

Introducción

Para comenzar a trabajar con *NClass*, sólo es necesario arrancar la aplicación, ya que está dedicada integramente al diseño de diagramas UML. Creando un nuevo proyecto, solo es necesario teclear un nombre para el mismo y más tarde hacer doble clic por debajo y teclear un nombre para el diagrama, o bien *Archivo >> Nuevo >> Diagrama C#*.

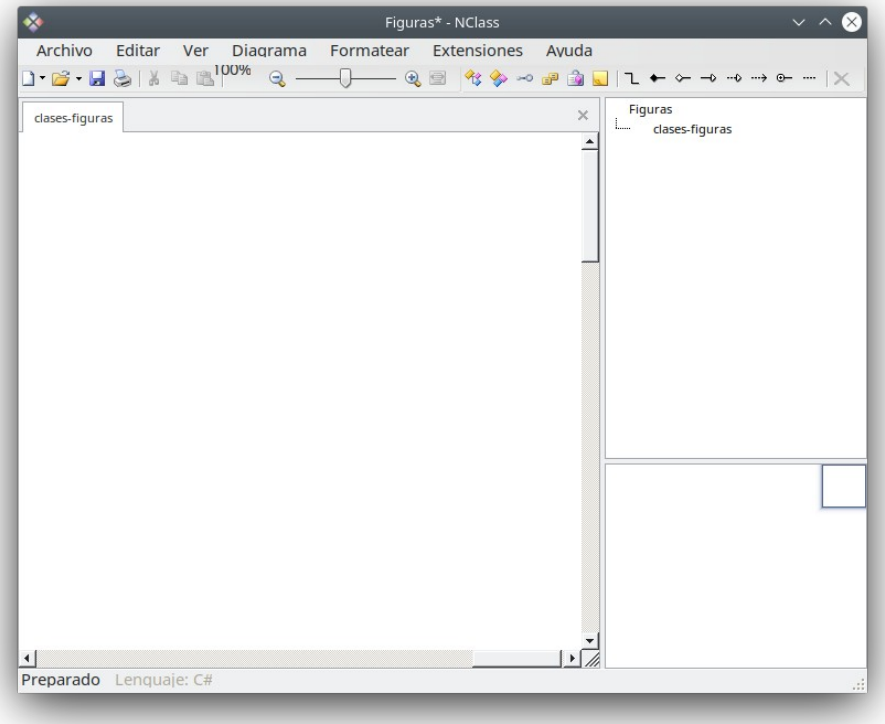

Figura 1: Inicio de un nuevo proyecto

Creación del diagrama de clases

La aplicación mantiene una barra de herramientas desde la que se puede fácilmente abrir, guardar o cerrar un proyecto; utilizar el zoom; e insertar una clase, estructura, interfaz… en el diagrama. De otra manera, se puede utilizar el menú *Diagrama >> Nueva >> Clase*. Una vez hecho clic en algún punto de la superficie de dibujo, aparece la clase de la que deberemos teclear el nombre. A partir de aquí, podemos con esta barra de acceso rápido insertar métodos, propiedades… o bien pulsando con el botón derecho podemos acceder a todos los miembros de la clase.

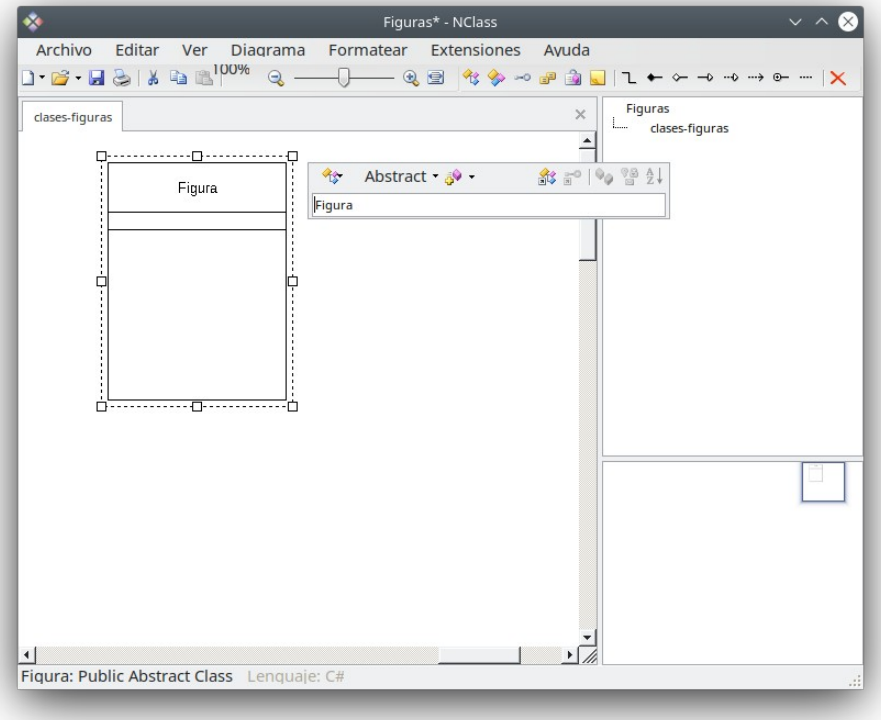

Figura 2: Creando una nueva clase.

En la Figura 2, podemos observar cómo se ha creado la nueva clase, y además se le ha atribuido el carácter *abstract*.

Al hacer clic con el botón derecho, y seleccionar “Editar miembros”, accedemos a las características de la clase que es posible modificar a nuestro gusto (Figura 3). Se pueden crear miembros desde aquí o desde la barra de herramientas que aparece al seleccionar una clase. Con teclear el tipo de retorno, nombre del método, y los parámetros (con sus tipos) entre paréntesis, el programa ya interpreta todas las características.

Al ser una herramienta específica para C#, no solo es posible crear métodos y atributos como en otras aplicaciones, sino también propiedades.

También podemos fácilmente establecer relaciones de herencia o de composición utilizando la barra de herramientas o *Diagrama >> Nueva >> Generalización*, por ejemplo, para herencia.

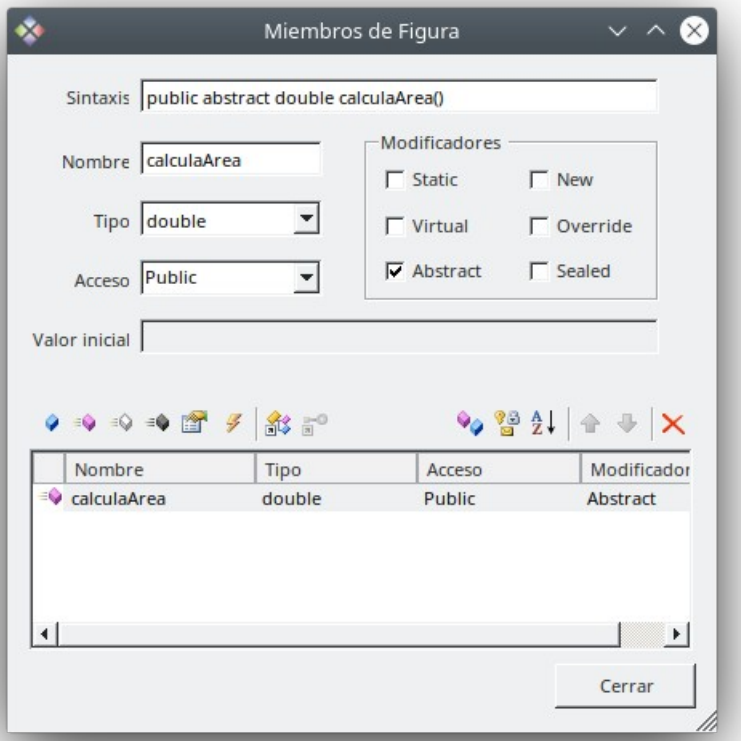

Figura 3: El diálogo con todos los miembros de la clase.

En los apartados de “propiedades”, "atributos" y "métodos", colocamos todo aquello (estático, abstract...) relativo a las propiedades del miembro. Es posible, si se desea, especificar el nombre del atributo o método, su visibilidad, su tipo, y sus parámetros (en el caso del método), por separado.

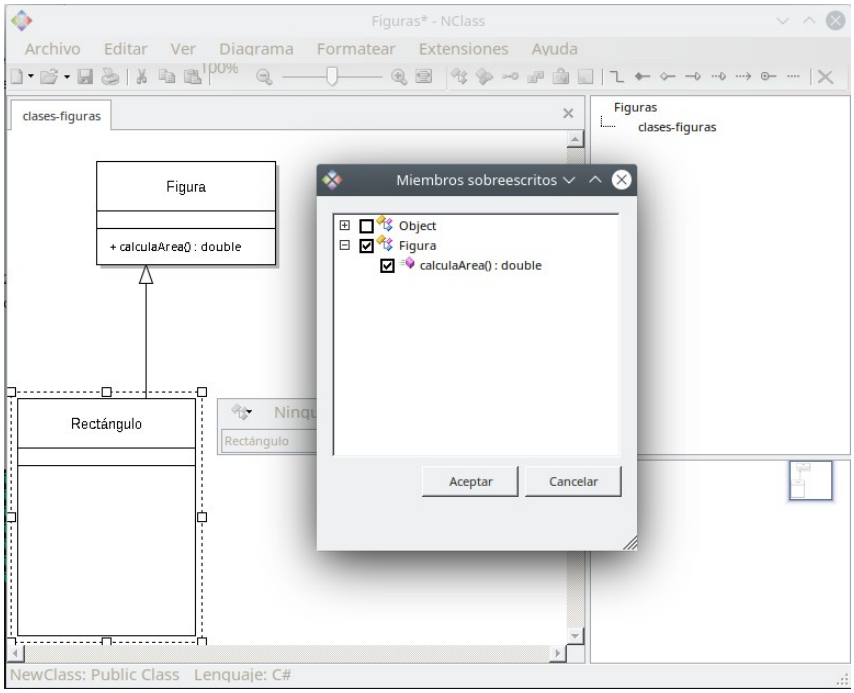

Figura 4: Reescribiendo los métodos de la clase padre.

En la Figura 4, se aprecia como, tras establecer una generalización entre dos clases, se puede pulsar en “miembros sobreescritos” para automáticamente generar los miembros (típicamente, métodos), que aparecerán en la clase derivada, sin tener que volver a introducirlos.

Es interesante la posibilidad de crear propiedades, pues hace más natural en C# el futuro código. Además, se pueden incluir también constructores, e incluso destructores.

En la Figura 5, aparece el diagrama final de clases tal y como lo genera la herramienta al seleccionar *Diagrama >> Generar como imagen*, empleando el formato PNG.

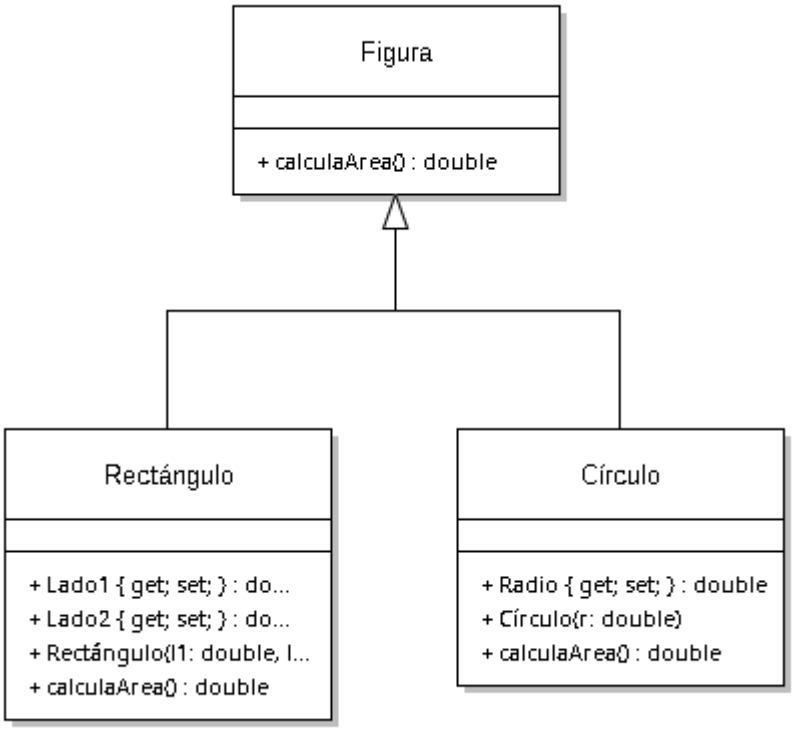

Figura 5: Diagrama final generado por la propia herramienta.

Generando el código

Una vez terminado el diagrama, para generar el código fuente sólo es necesario seleccionar *Diagrama >> Generar código*, con lo que aparece el diálogo que se muestra en la Figura 6. En la parte de arriba se selecciona la carpeta en la que se va a crear el código generado. En la izquierda, las importaciones que se desea que aparezcan automáticamente en todos los archivos. A la derecha hay ciertas opciones de formato. Cuando todo esté listo, se selecciona “Generar” para obtener el código.

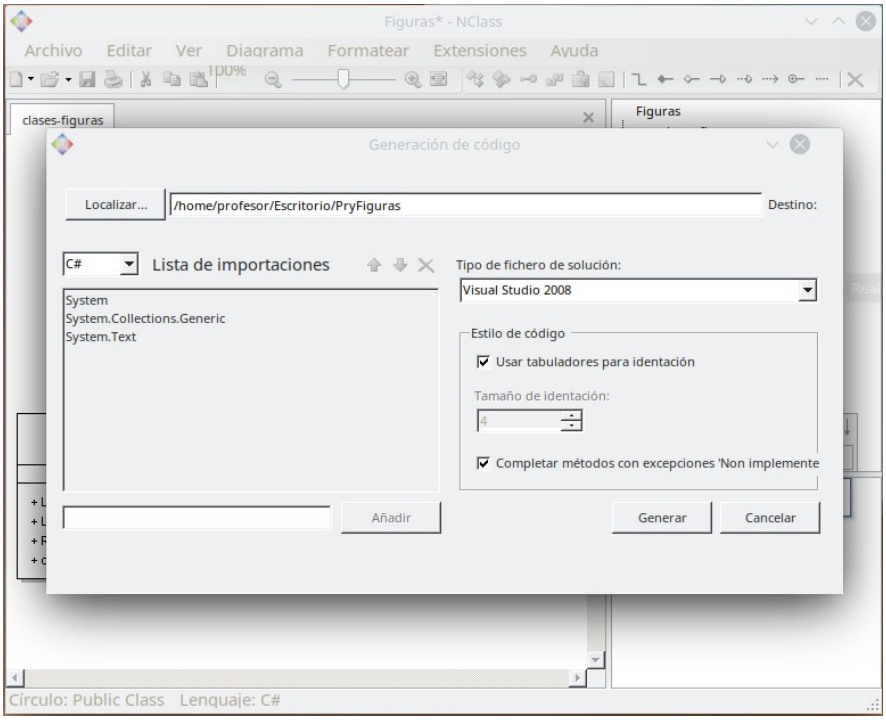

Figura 6: Generando el código a partir del diagrama.

Un ejemplo del código generado aparece a continuación.

```
using System;
using System.Collections.Generic;
using System.Text;


namespace Figuras.clases-figuras
{
	public abstract class Figura
	{
		public abstract double calculaArea();
	}
}
```