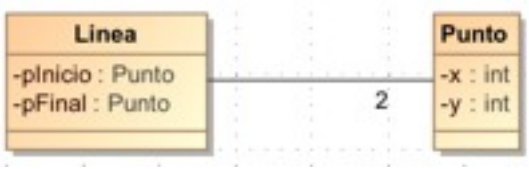
Diagramas de Clases
Definición de Atributos (in-line vs en la relación) y Navegabilidad

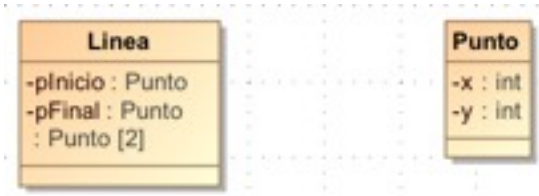
Vamos a utilizar como ejemplo el siguiente ejercicio simple visto en clase.
Modelar mediante un diagrama de clases la estructura de un sistema que contiene una colección de líneas en dos dimensiones y que permite la siguiente funcionalidad:
- Dada una línea, buscar todas las líneas que la cortan.
- Dado un punto, buscar todas las líneas que pasan por él.

Definición de Atributos

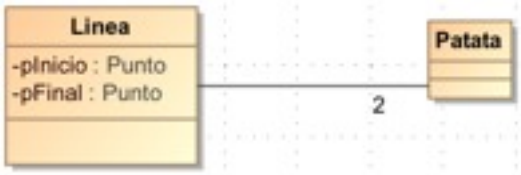
Dado el siguiente diagrama, ¿cuántos puntos harían falta para definir una línea?



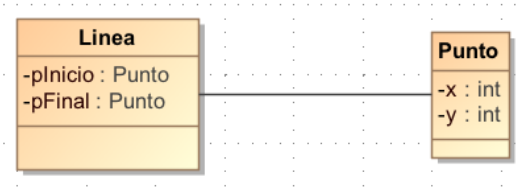
La respuesta es 4.
Hay dos atributos del tipo punto definidos in-line que son: pInicio y pFinal.
Además hay una lista de 2 puntos, que es un atributo definido en la asociación, al cuál no se le ha dado nombre (como en sí es un atributo, lo suyo es darle nombre en la asociación, como +puntos).
El diagrama anterior sería equivalente al siguiente.



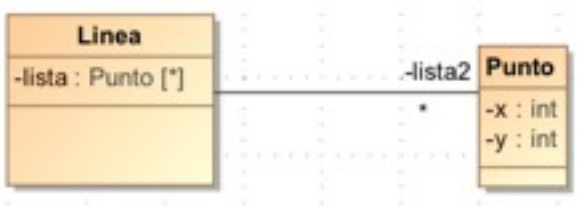
Con este ejemplo quizás se vea más claro. Cada Línea tiene 4 elementos (2 puntos y 2 patatas), si en vez de patatas son puntos, tenemos lo de antes.



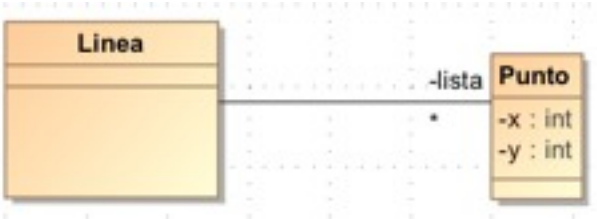
Lógicamente, si no ponemos multiplicidad en la asociación, estaríamos expresando que cada línea tiene 3 puntos (2 attributos in-line y 1 attributo de la relación).

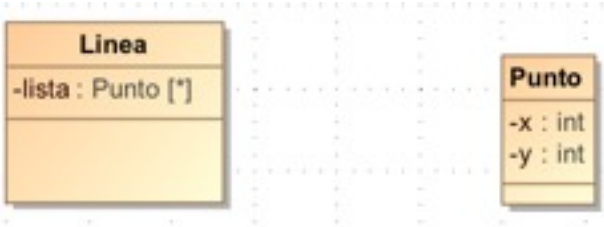


Hay que tener cuidado con esto para evitar duplicar información. Por otro lado, es probable que encontréis situaciones en las que se utiliza el mismo nombre para el atributo in-line que para el de la relación, con idea de expresar que es el mismo atributo.
*en el siguiente diagrama he usado “lista” y “lista2” porque directamente la herramienta detecta que se están definiendo 2 atributos con el mismo nombre y no me deja.



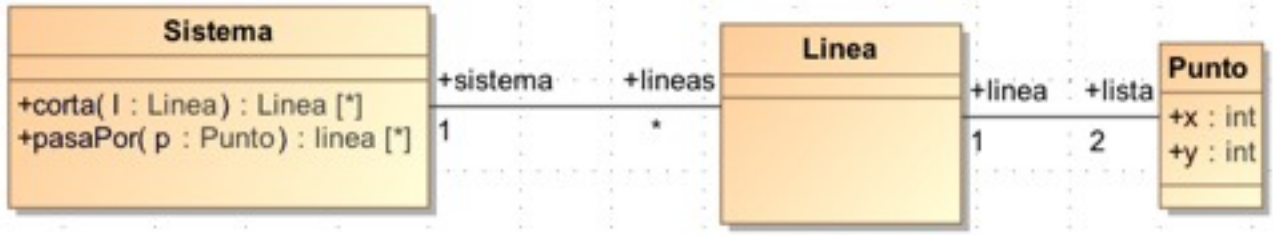
La forma correcta de representar esto sería una de las siguientes dos opciones:





Navegabilidad

Veamos ahora cómo afecta la navegabilidad a distintas soluciones del problema.



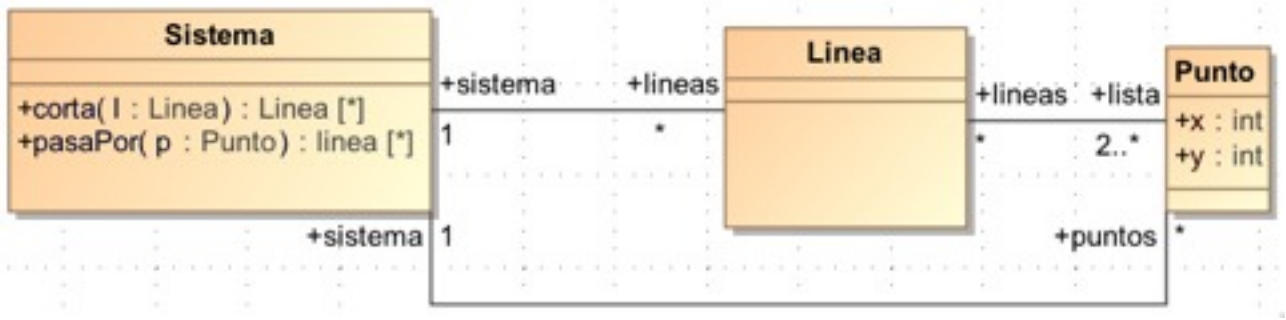
Dada la especificación del problema, esta solución sería correcta, ya que podemos almacenar las líneas definiéndolas mediante dos puntos y calcular lo que se nos pide.

Aunque si nos fijamos bien, no podríamos definir 1 misma instancia de punto que fuera común a varias líneas (ya que cada punto definido en el sistema está asociado a 1 sola línea). Sí podríamos hacerlo definiendo dos instancias, por ejemplo P1{x=1; y=2} y P2{x=1; y=2}.

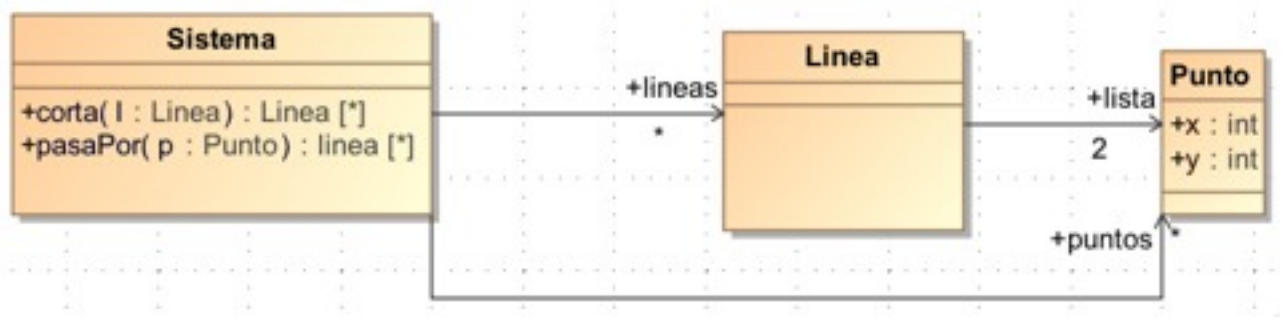
Aunque la especificación del problema no pedía "almacenar" los puntos de corte, o puntos aislados en el plano, si quisiéramos hacer esto no nos valdría lo de antes, ya que según el modelo, todos los puntos que definamos obligatoriamente pertenecen a una línea.

Si quisiéramos hacer esto tendríamos varias opciones:

Una de ellas sería tocar la multiplicidad de las asociaciones de modo que cada línea pueda tener varios puntos y cada punto pueda asociarse a 0 o varias líneas. También sería recomendable añadir una lista de puntos al Sistema (asociación de Sistema a Punto), para tener conocimiento de todos los puntos, sean sueltos, de corte, o que definen a una o varias líneas.



Otra opción podría ser restringiendo la navegabilidad.
Si hacemos que Punto no conozca a Linea, al definir puntos "sueltos" no tenemos ningún atributo del tipo línea al que haga falta asociarle ningún valor.



En el modelo se ha puesto también que las líneas tampoco conozcan al sistema y además se ha añadido una asociación de Sistema a Punto para que el sistema pueda acceder directamente a una lista de puntos que almacena todos los puntos (sueltos, de corte, que definen líneas).

Si optáis por este tipo de soluciones en el que restringimos la navegabilidad habría que tener bien en cuenta dónde definimos las operaciones de modo que sea posible que accedan a los datos que necesitan. En este caso si las hubiéramos definido en la clase Línea, estas operaciones no podrían acceder a toda la colección de líneas (ya que Linea no es navegable hacia Sistema que es quien tiene la lista de lineas).