&f7('5$

.RKIRMIV´E HI SJX

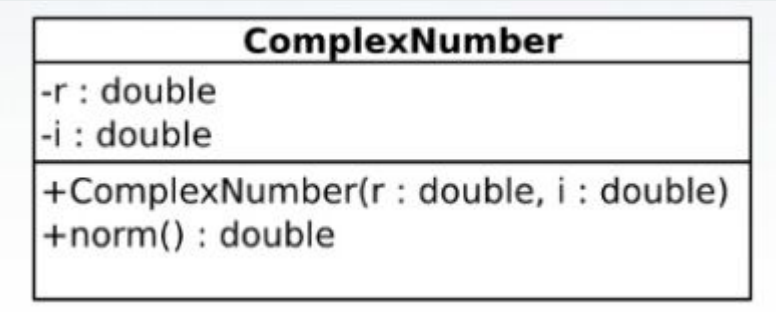
Pendiente de clase anterior….

Retomando el proyecto en grupo realizado en Ingeniería de software I:

1. Relea el trabajo
2. Relacione las tareas que realizó con las fases de UP.
3. Realice el diagrama de actividad
4. Basado en su conocimiento del proceso indique una mejora o un problema.

Clases bien creadas… recordamos los puntos fundamentales para avanzar:

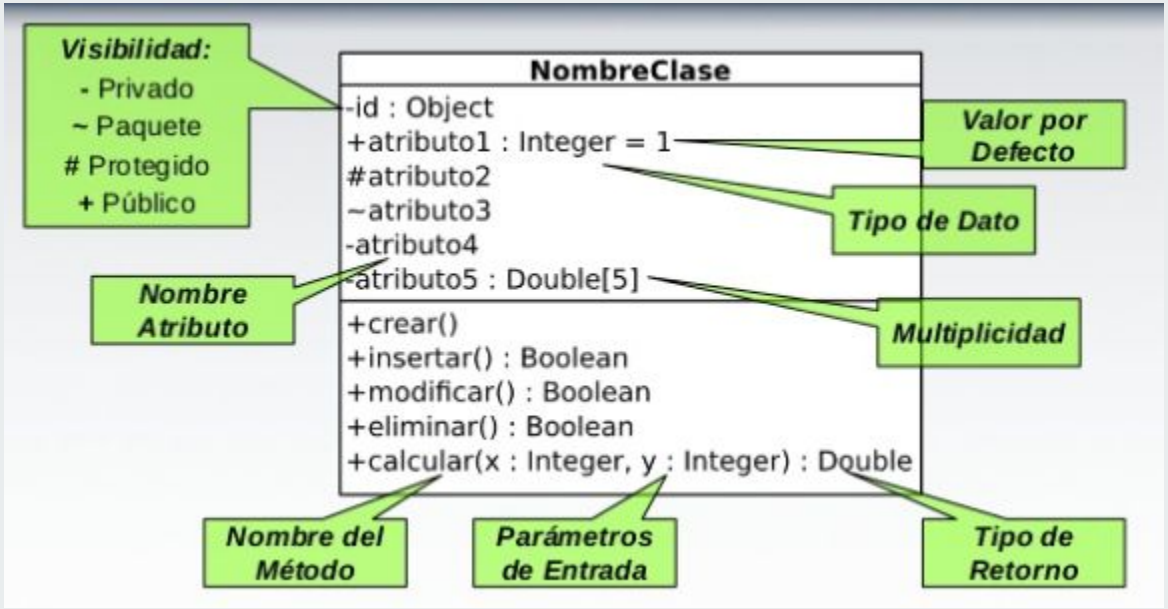
Una clase define la estructura y el comportamiento de un conjunto de objetos que tienen el mismo patrón estructural y de comportamiento.



Atributos: propiedades relevantes de una clase. Representan su estructura.

Métodos: comportamiento asociado a una clase.

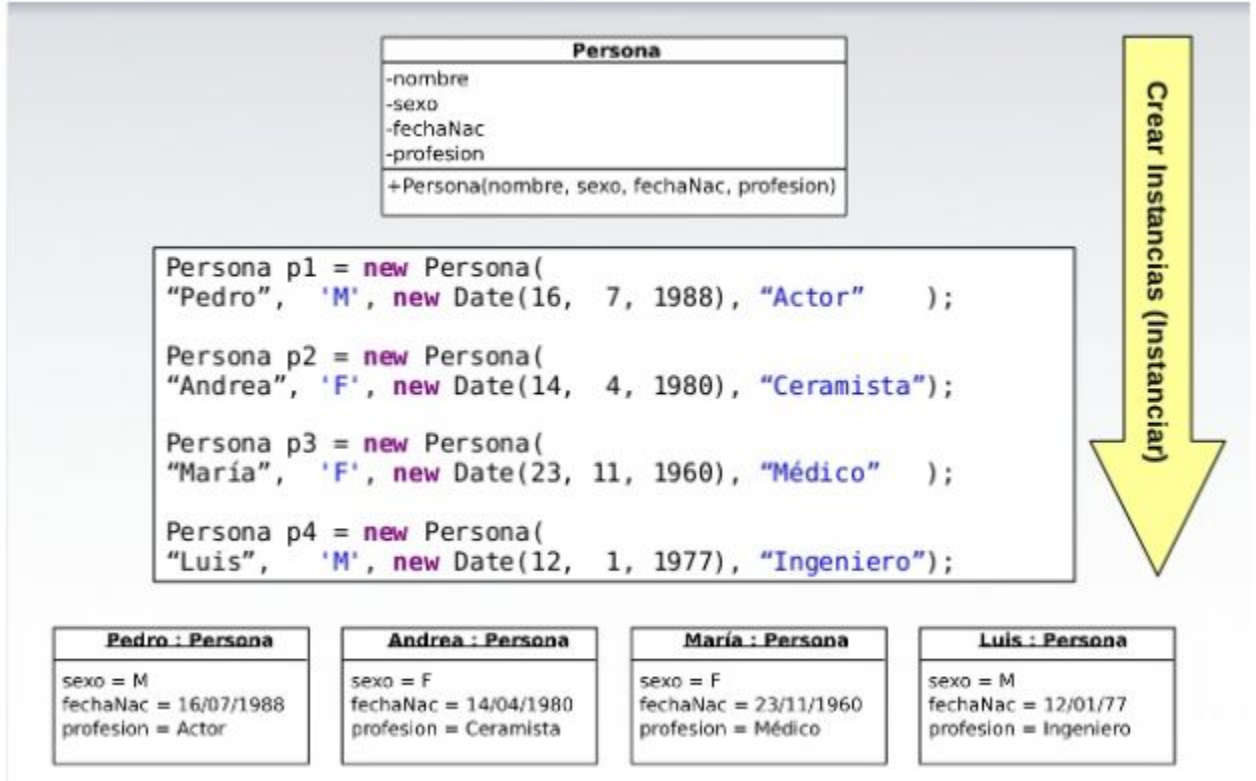
UNIDAD 2: Workflow de diseño



Visibilidad	Public	Private	Protected	Default*
Desde la misma clase	✓	✓	✓	✓
Desde una subclase	✓	✓	✓	⊘
Desde otra clase (no subclase)	✓	⊘	⊘	⊘

Clases???.... instancias???

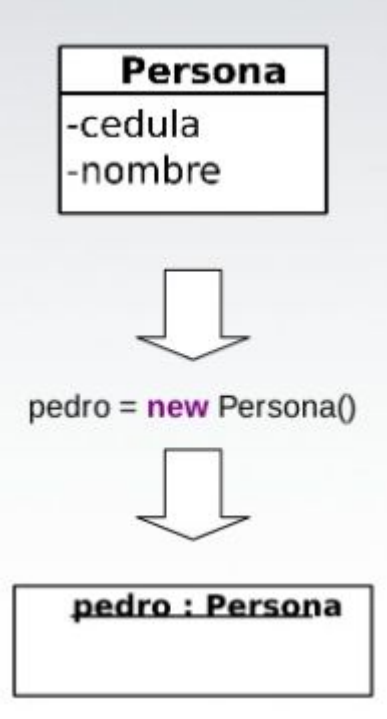
UNIDAD 2: Workflow de diseño



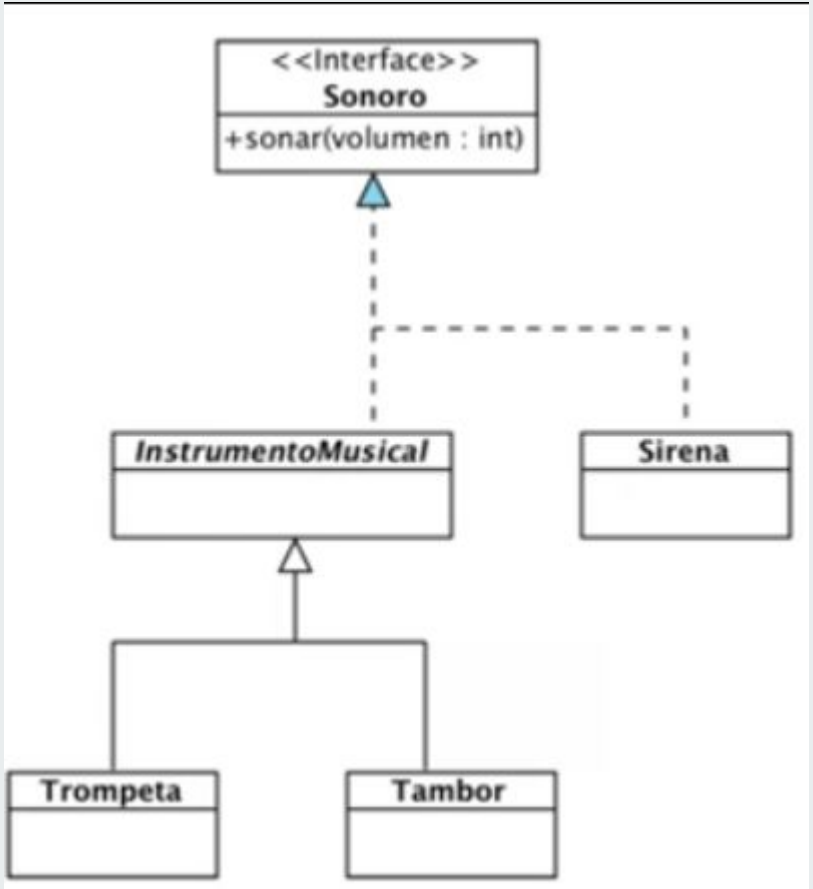
Instancia Es la particularización, realización específica u ocurrencia de una determinada clase, entidad o prototipo.

Instanciar es el proceso de generar instancias de una clase.

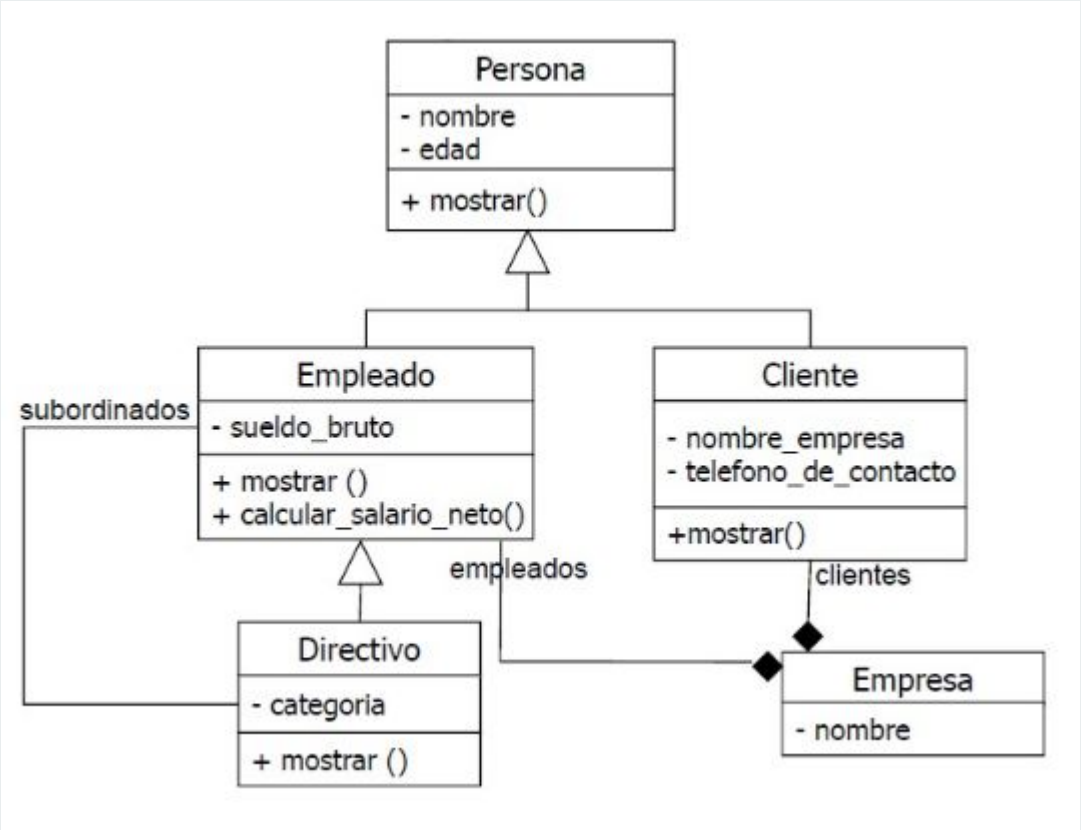
Objeto es una instancia de una clase



Interfaz: es un conjunto de operaciones y/o propiedades que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto.



Herencia: se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad a un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre.



Las clases son bloques de construcción del modelo de diseño y son vitales entender cómo se modelan de forma eficaz.

Las clases de diseño son clases cuyas especificaciones se han completado hasta tal nivel que se pueden implementar.

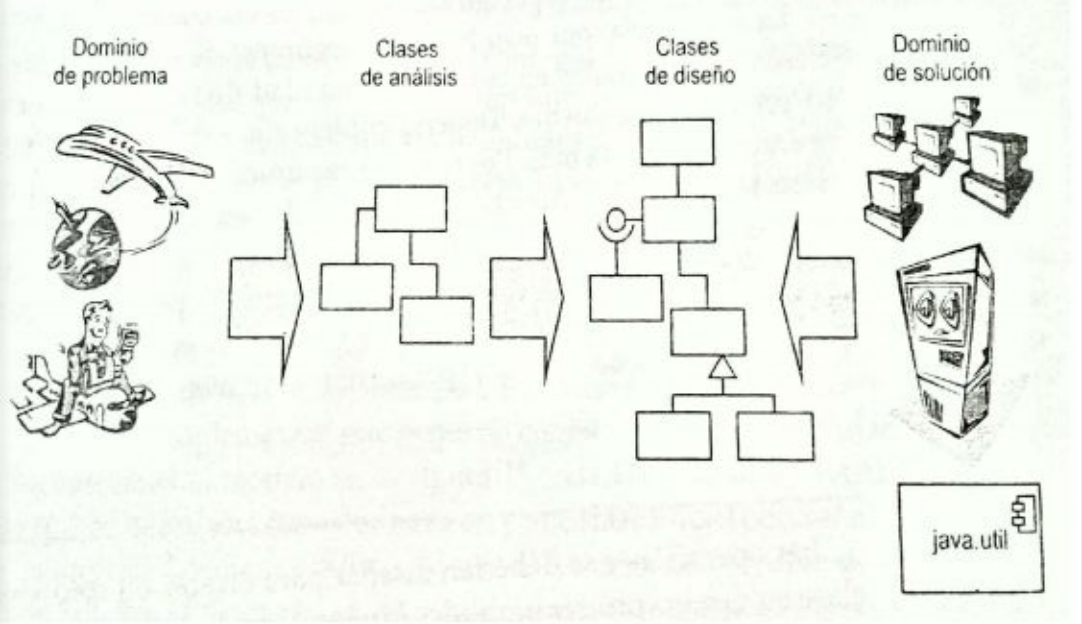
Una clase de diseño completa es una que está suficientemente detallada para servir como una buena base para crear código fuente.

En análisis el origen de las clases es el ámbito del problema. Este es el conjunto de requisitos que describe el problema que está tratando de solucionar.

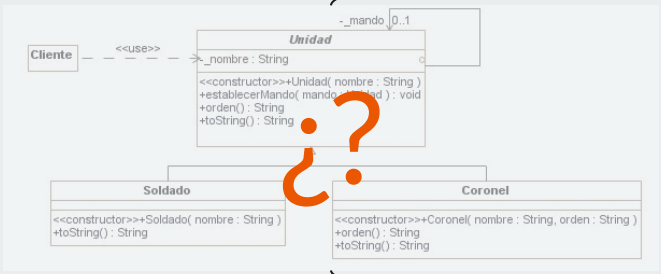
Las clases de diseño proceden de dos lados:

. El ámbito del problema haciendo una mejora de las clases de análisis.

. El ámbito de la solución (librerías de clases de utilidad y componentes reutilizables)



¿Con qué grado de detalle diseñamos las clases?



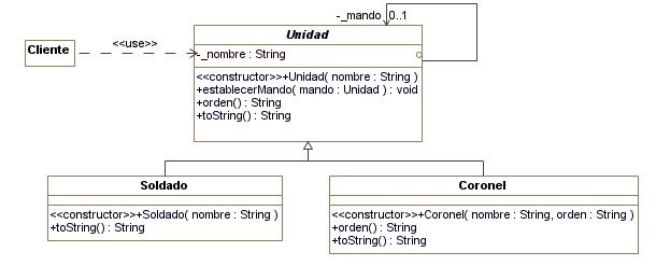
```
' Globales -----------
Var Variable0:Booleano
Var Variable1:Cadena
' Fin Globales -------
Proc Procedimiento ' <- Procedimiento sin retorno.
   Var Variable2:Entero ' Locales
   Var Variable3:Real

   Si Variable0 = Falso Entonces ' Condición "If"
      Contar Variable2 = 0 a 9 ' Bucle "For"
         Variable1 = Variable1 + "1"
      Seguir ' "End For"
   FinSi ' "End If"

   Variable3 = 5.13
FinProc
```

El método elegido de implementación determina el grado de totalidad que necesita en las especificaciones de la clase de diseño. Si el modelo de la clase de diseño se proporcionará a programadores que lo utilizaran como una guia para escribir código, entonces las clases de diseño solamente necesitan estar los suficientemente completas para permitirles realizar esa tarea de forma eficaz.

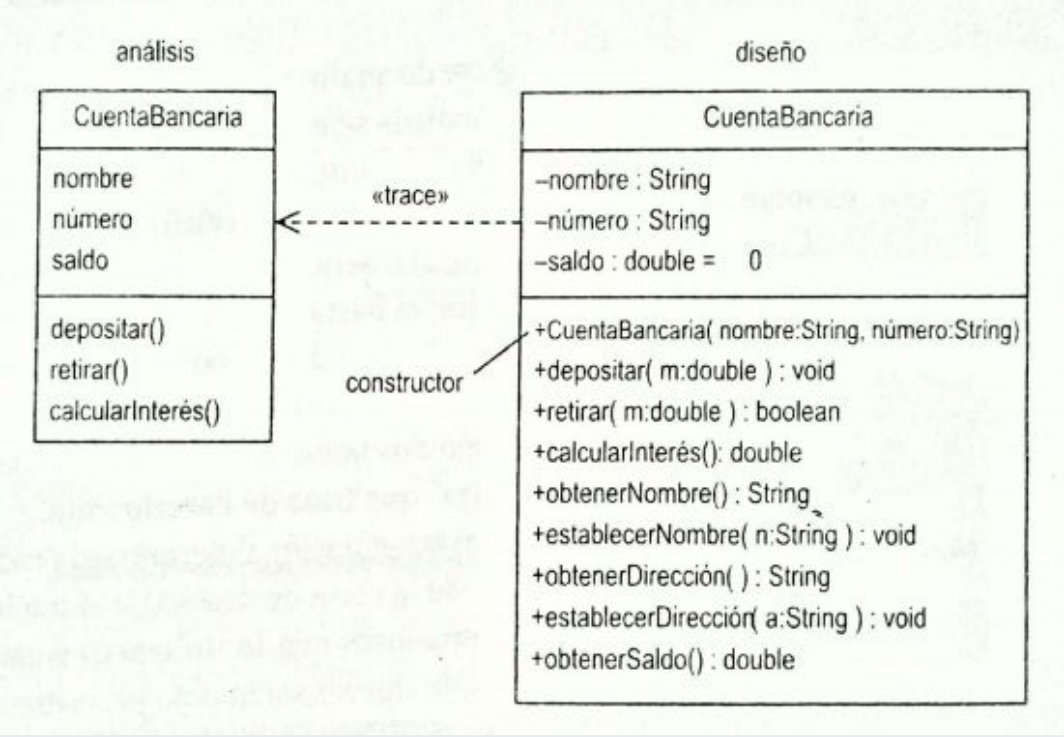
Si se trata de generar código desde las clases de diseño con una herramienta de modelado apropiada, las especificaciones de clase de diseño deben estar completas en todos los aspectos.



El análisis trata sobre modelar lo que el sistema debería hacer.

El diseño es sobre modelar cómo el comportamiento se puede implementar.

¿Cómo convertimos las clases de análisis en clases de diseño?



Partiendo de las clases de análisis hay que realizar algunos ajustes para convertirlas en clases de diseño:

Complete el conjunto de atributos y especifiquelos incluyendo nombre, tipo, visibilidad y valor predeterminado si aplicara.

Complete el conjunto de operaciones y especificarlas incluyendo nombre, lista de parámetros y tipo de retorno.

¿cómo se ve la clase?

¿es demasiado compleja?

¿falta algo?

Clases de diseño bien creadas

¿está muy acoplada a otras clases?

¿realiza lo que podría esperarse por su nombre?

Revise el siguiente ejemplo

Totalidad

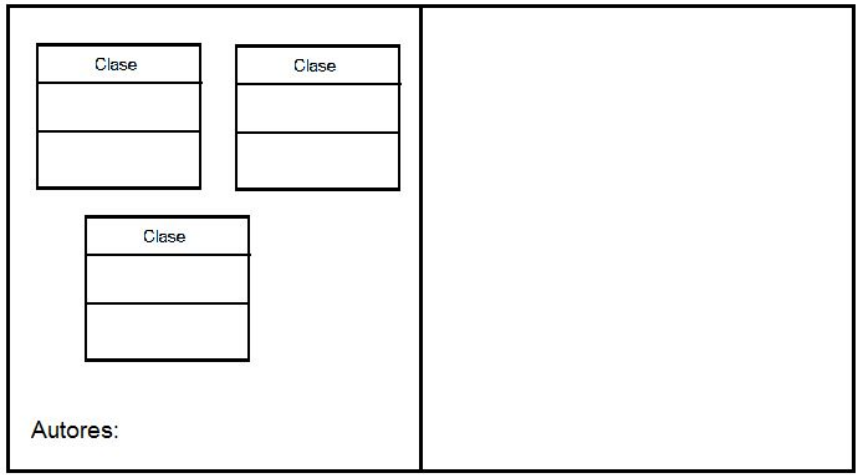
La característica de totalidad trata sobre proporcionar a los clientes de una clase lo que podrían esperar.

Los clientes asumen el conjunto de operaciones que debería poner disponible, según el nombre de la clase.

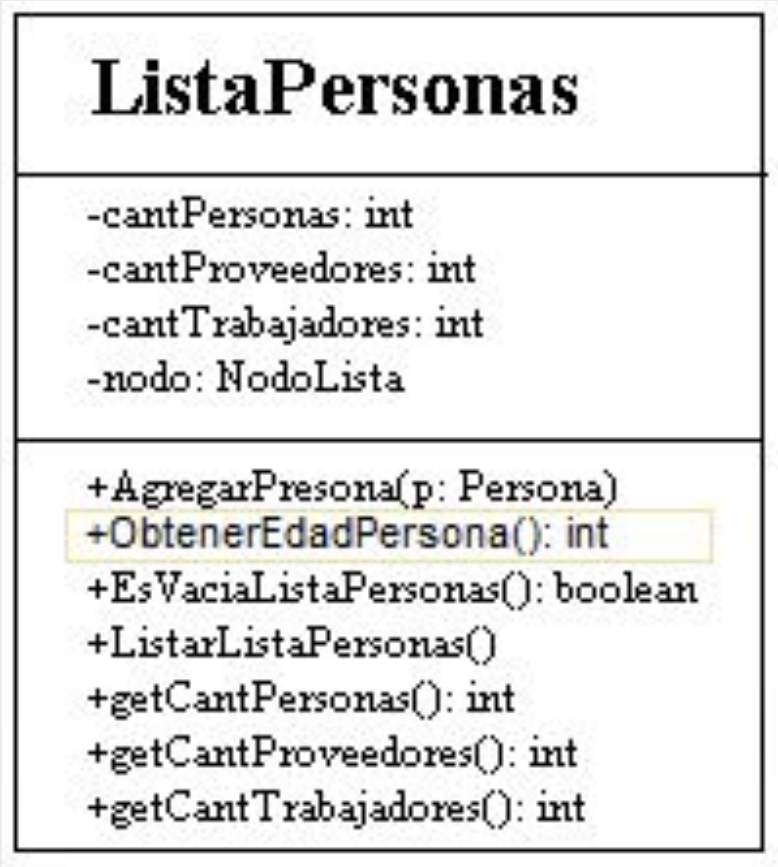
Asegurarse de que las clases satisfacen todas las expectativas razonables de cliente.

Totalidad

Escriba en una tarjeta una clase que no cumpla esta característica.



Revise el siguiente ejemplo



Suficiencia

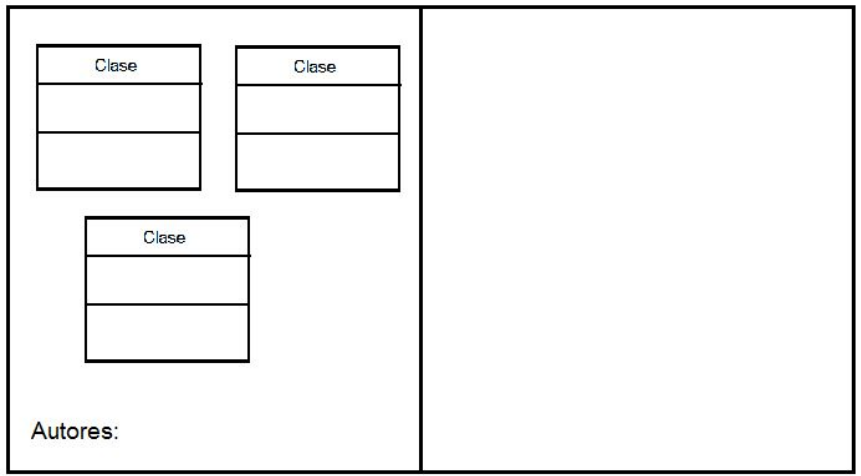
Todas las operaciones de la clase están centradas en realizar la finalidad de la clase.

La suficiencia trata sobre mantener la clase de diseño lo más sencilla y centrada posible.

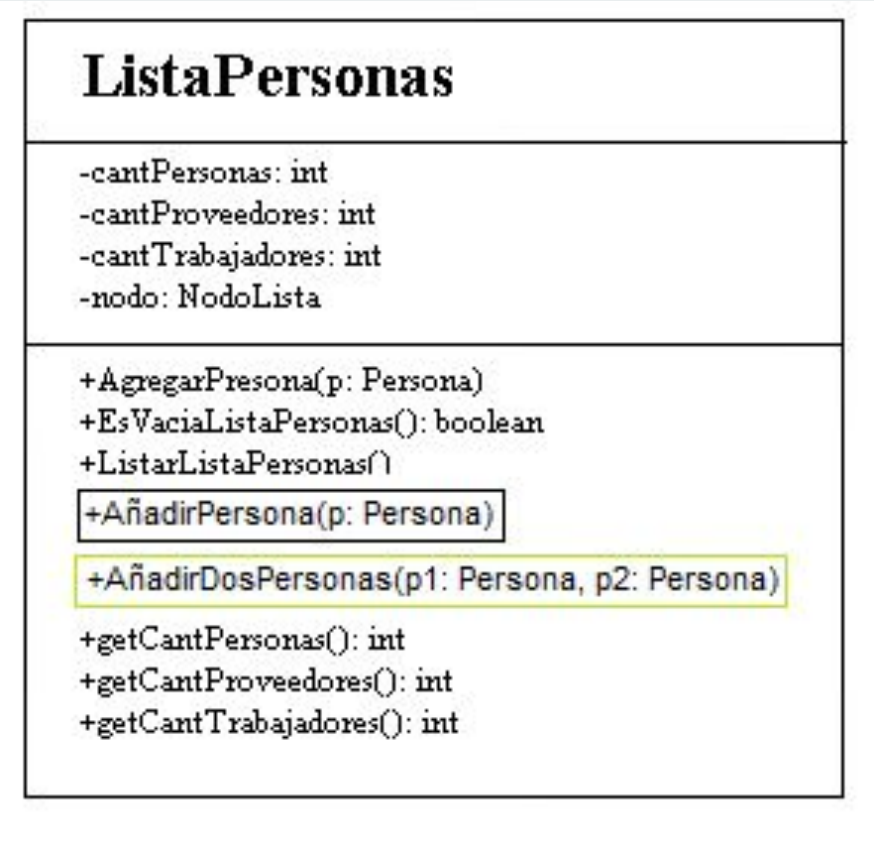
La regla de oro para la totalidad y suficiencia es que una clase debería hacer o que los usuarios de la clase esperan, ni más ni menos.

Suficiencia

Escriba en una tarjeta una clase que no cumpla esta característica.



Revise el siguiente ejemplo



Sencillez

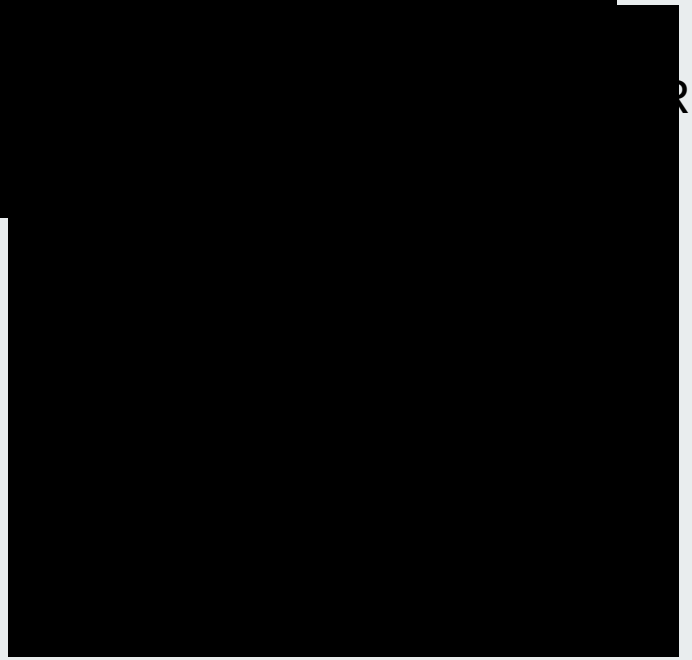
Las operaciones se deberían diseñar para obtener un solo servicio sencillo.

Una clase no debería ofrecer múltiples formas de realizar lo mismo.

Las clases deberían siempre poner disponible el conjunto de operaciones más sencillo y pequeño posible.

81,'$' :RUNIORZ GH GLVH³R

6HQFLOOH]



```
81,'$'    'LDJUDPDV GH DFWLYLGDG
```