UML Diagramas de clase

Su estructura

Ángel González M.

Introducción

Recuerda que los casos de uso describen el comportamiento del sistema como un conjunto de metas y objetivos.

Las clases serán las plantillas para generar objetos que vas a necesitar tener en tu sistema y sus relaciones entre ellos.

El diagrama de clases es estático, es decir, representa una vista estática de la aplicación mientra no está ejecutándose.

El diagrama de clases describen atributos y operaciones, y restricciones impuestas por el sistema.

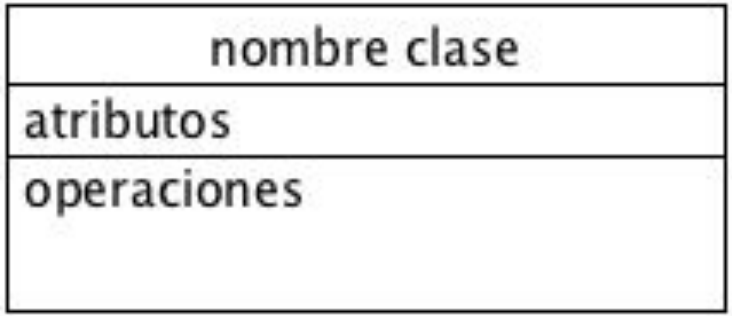
El diagrama de clases muestra un conjunto de colecciones de clases interfaces, asociaciones, colaboraciones y constraints(restricciones)

Clase (class)

Esta es la estructura básica de un diagrama de clases.

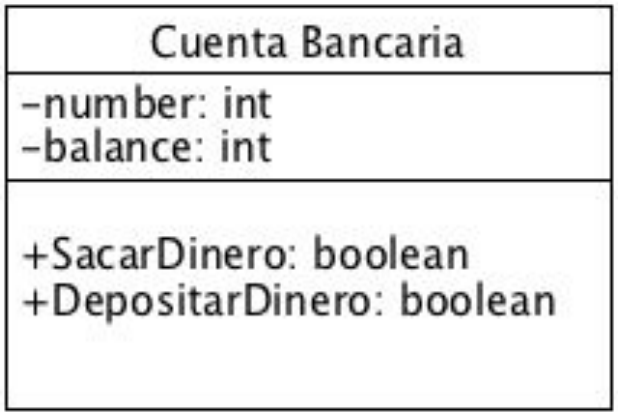
En su interior podemos incluir atributos y operaciones.

No es necesario que incluyan estos atributos y operaciones ya que son opcionales

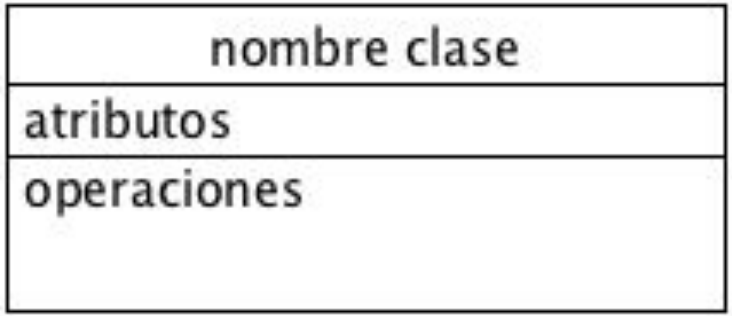


Ejemplo de clase

En nuestro ejemplo, la clase cuenta bancaria podría quedar así



Atributos en las clases



Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

Visibilidad: nos permite controlar si un elemento(atributo/objeto/propia clase) puede ser accesible desde el exterior. Esto se conoce con el nombre de encapsulación.

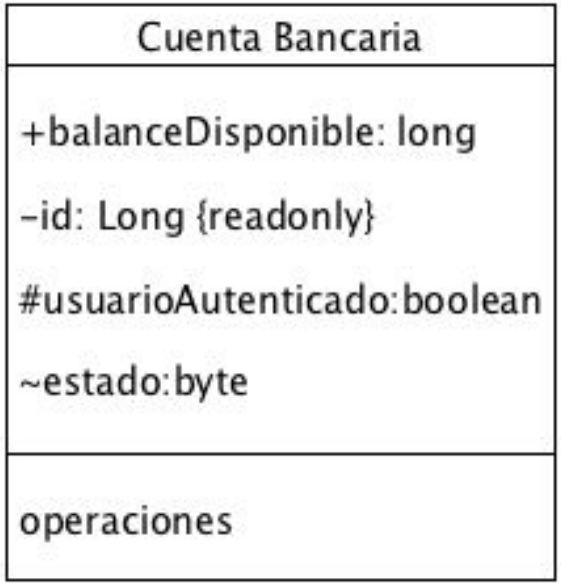
Tenemos varios tipos de visibilidad:

- public
- private
- protected
- package

Visibilidad pública

Se usa el símbolo más +

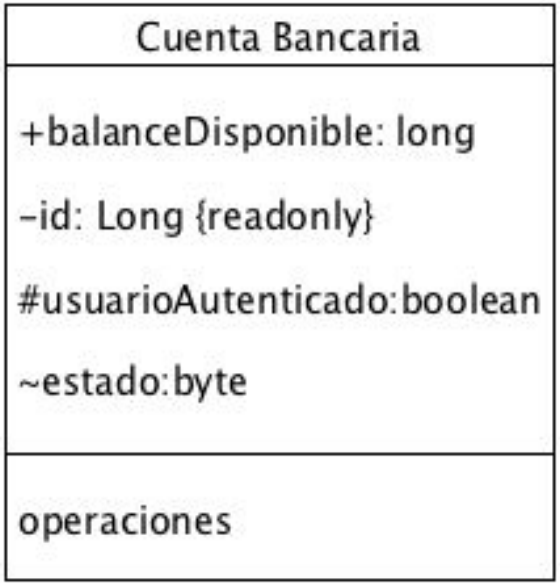
Si un atributo es público se puede acceder a él desde cualquier sitio



Visibilidad privada

Se usa el símbolo más -

Si un atributo es privado no se puede acceder a él desde ningún sitio, solamente desde la propia clase donde está definido

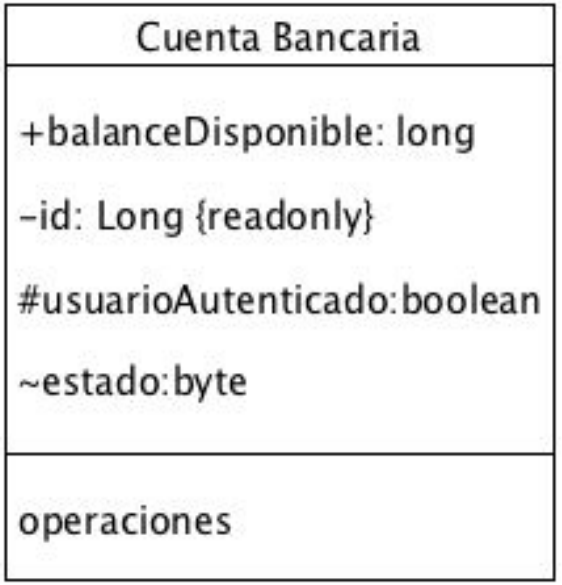


Visibilidad protected

Se usa el símbolo más #

Si un atributo es protegido, solo se podrá acceder a él:

- Desde las clases hijas que hereden de la clase en la que ese atributo está definido
- Desde la propia clase donde está definido

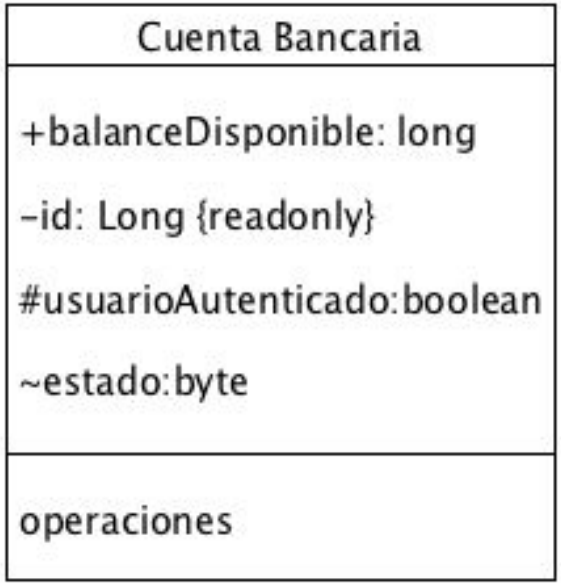


Visibilidad package

Se usa el símbolo más ~

Si un atributo es protegido, solo se podrá acceder a él:

- Las la clase está en el mismo paquete que la clase en la que ese atributo está definido
- Desde la propia clase donde está definido

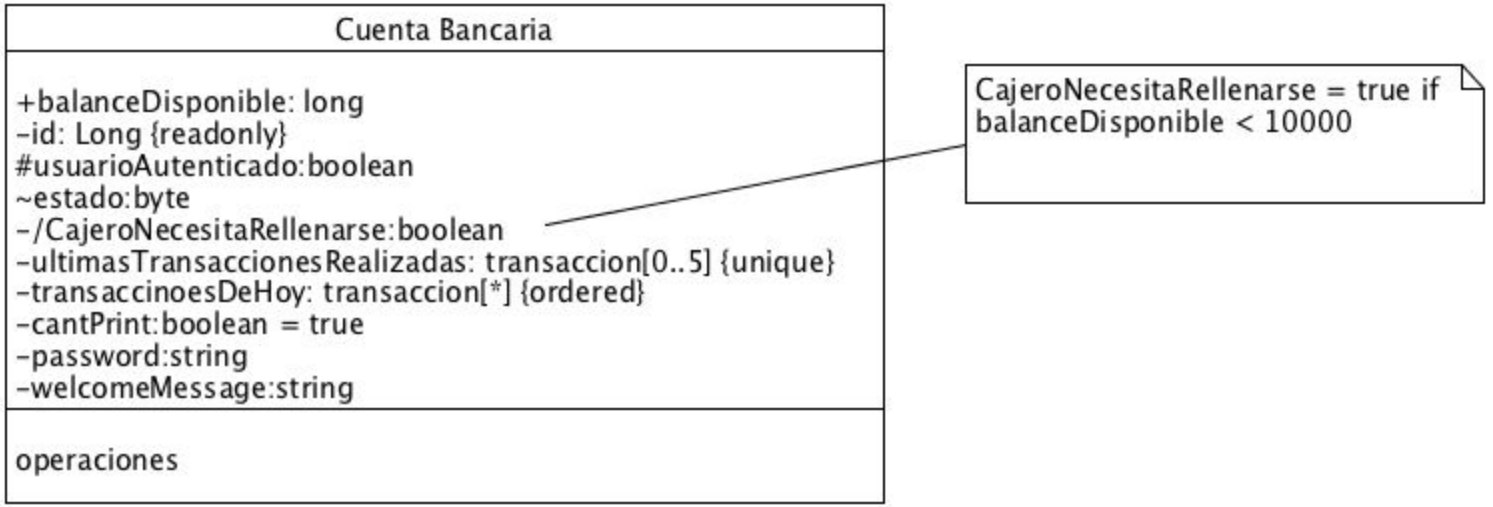


MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
No especificado	Sí	Sí	No	No
private	Sí	No	No	No

Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

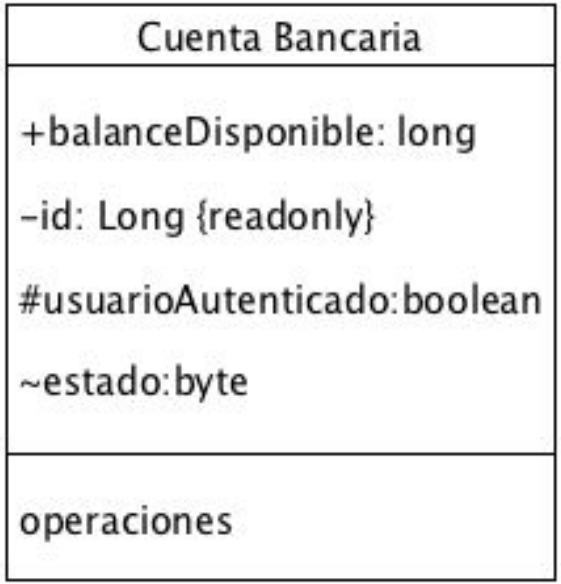
El backslash (/) indica si el atributo es un atributo derivado. Es decir, su valor se genera automáticamente en función de otro atributo existente.



Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

Este será el nombre del atributo

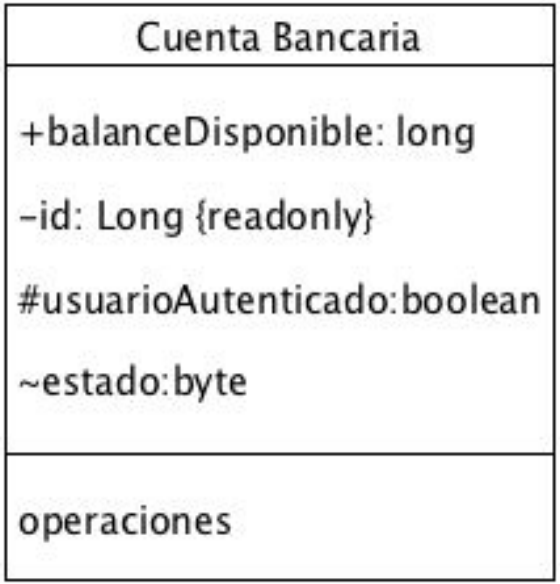


Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

Tipo nos determina el tipo de atributo

boolean, entero, flotante, ...

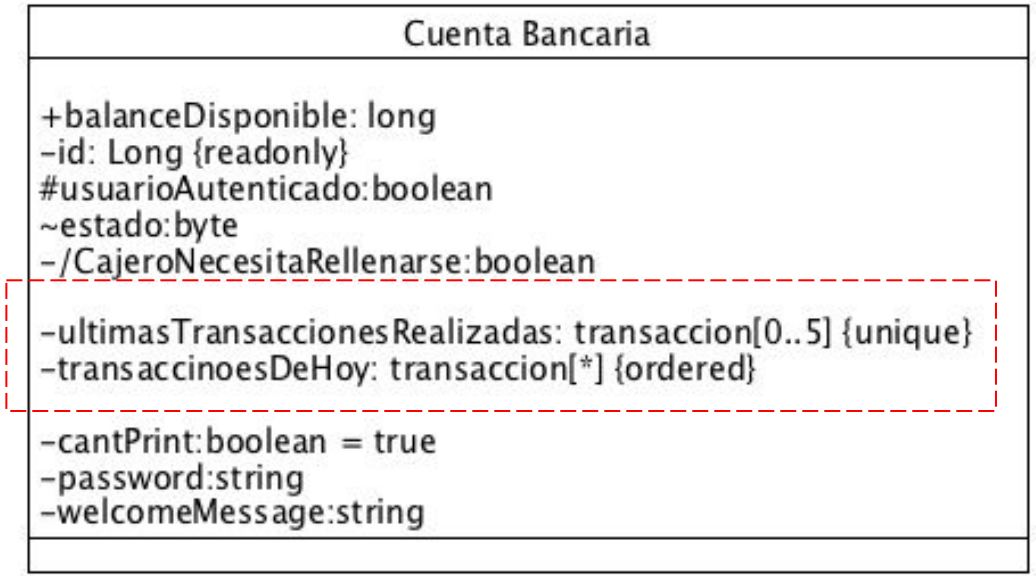


Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

La multiplicidad indica cuántas instancias del elemento vamos a tener

- Podemos especificar un rango [x..y]
- O un valor directamente [x]
- incluso indicar una cantidad ilimitada [*]

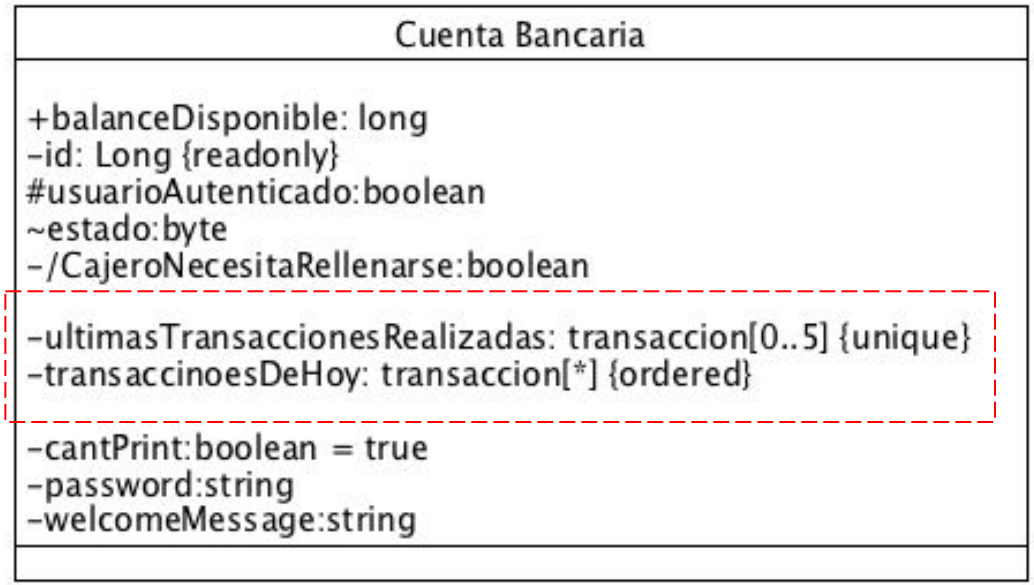


Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

En la multiplicidad también podemos indicar si el conjunto de objetos permite elementos repetidos o no {unique}

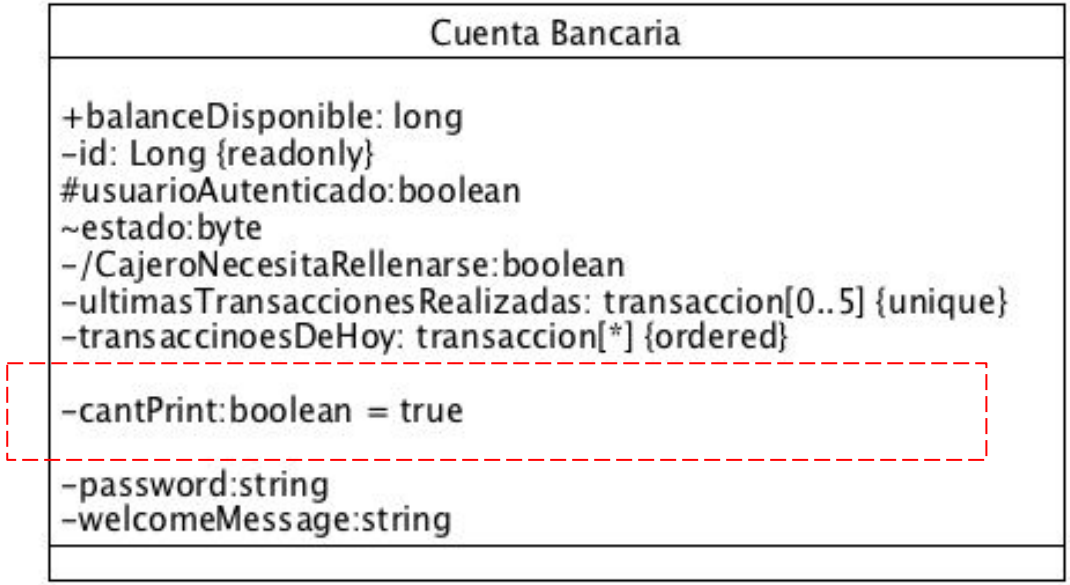
O si el conjunto de objetos está ordenado por algún criterio {ordered}



Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

Establece el valor por defecto que tendrá de forma predeterminada el atributo.

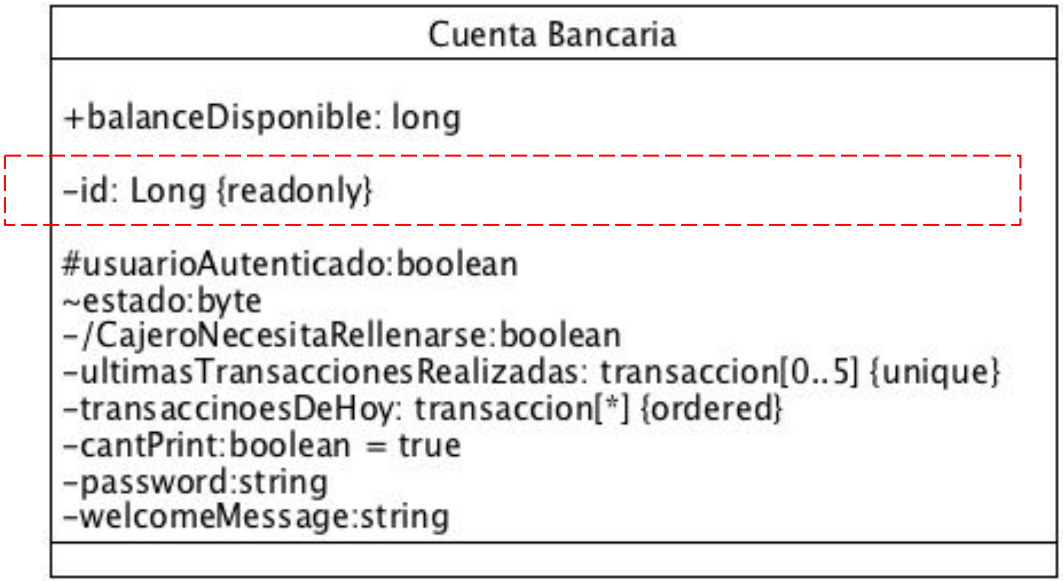


Atributos en el formato inline

Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

Son propiedades adicionales que podemos asociar al atributo y dotan de significado a este. La más usada es **readOnly**

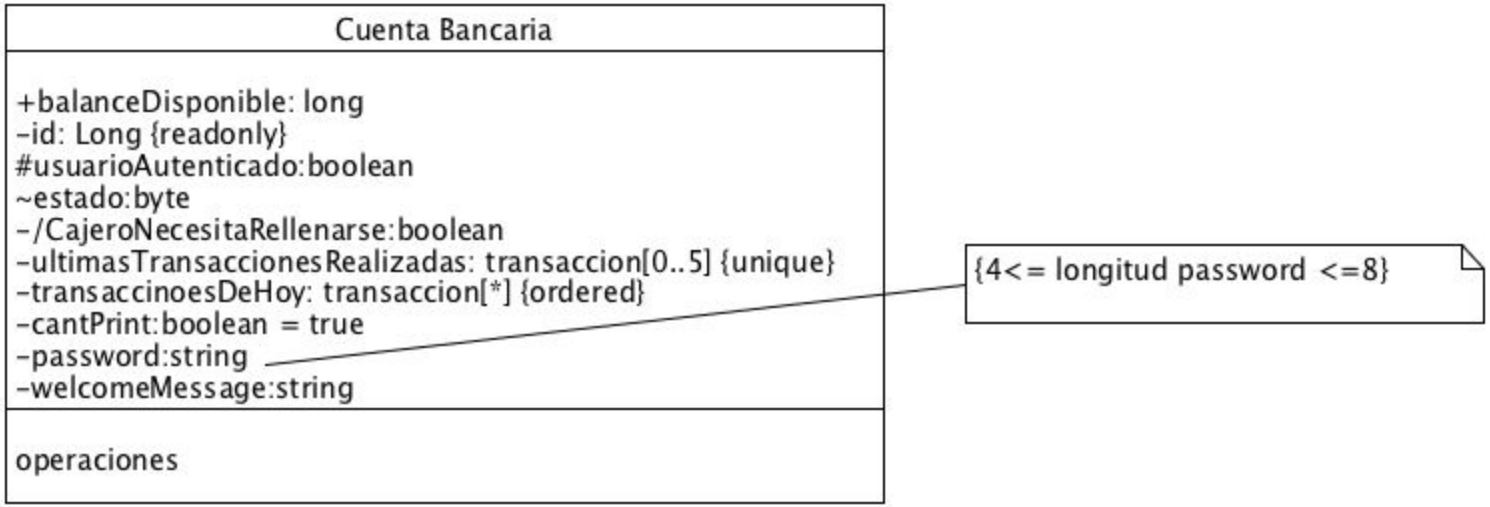
La multiplicidad no se coloca en las propiedades.



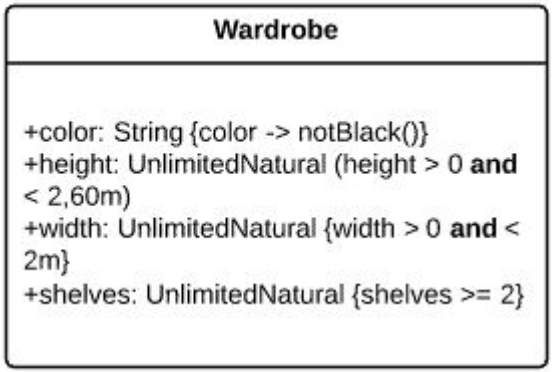
Atributos en el formato inline

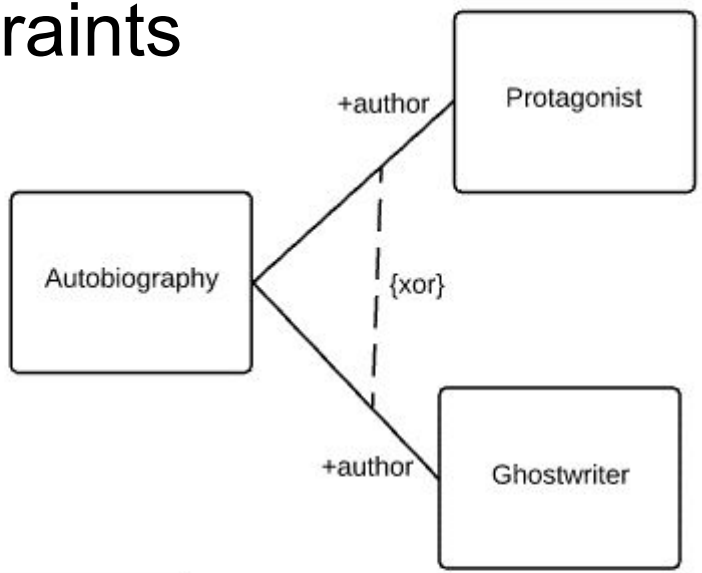
Visibilidad / nombre: tipo multiplicidad = default{propiedades, constraints}

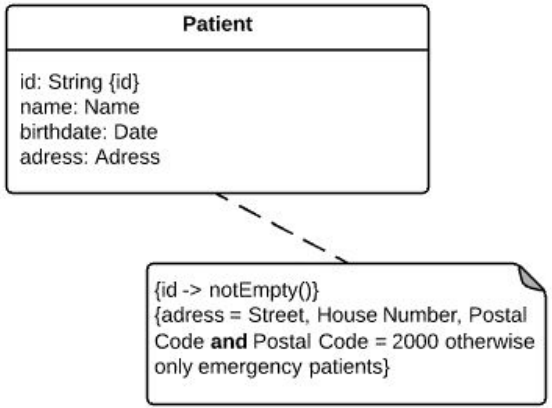
Las constraints van entre {} e implican algún tipo de restricción en el atributo.



Ejemplo uso de constraints







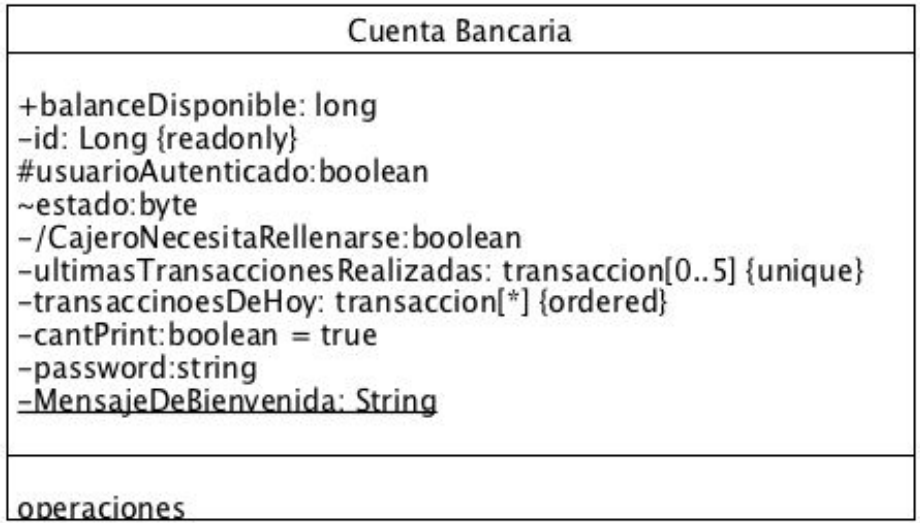
Ángel González M.

Atributos estáticos (static atribute)

Los atributos estáticos van **subrayados**.

Un atributo estático implica que su valor es compartido por todos los objetos de la clase.

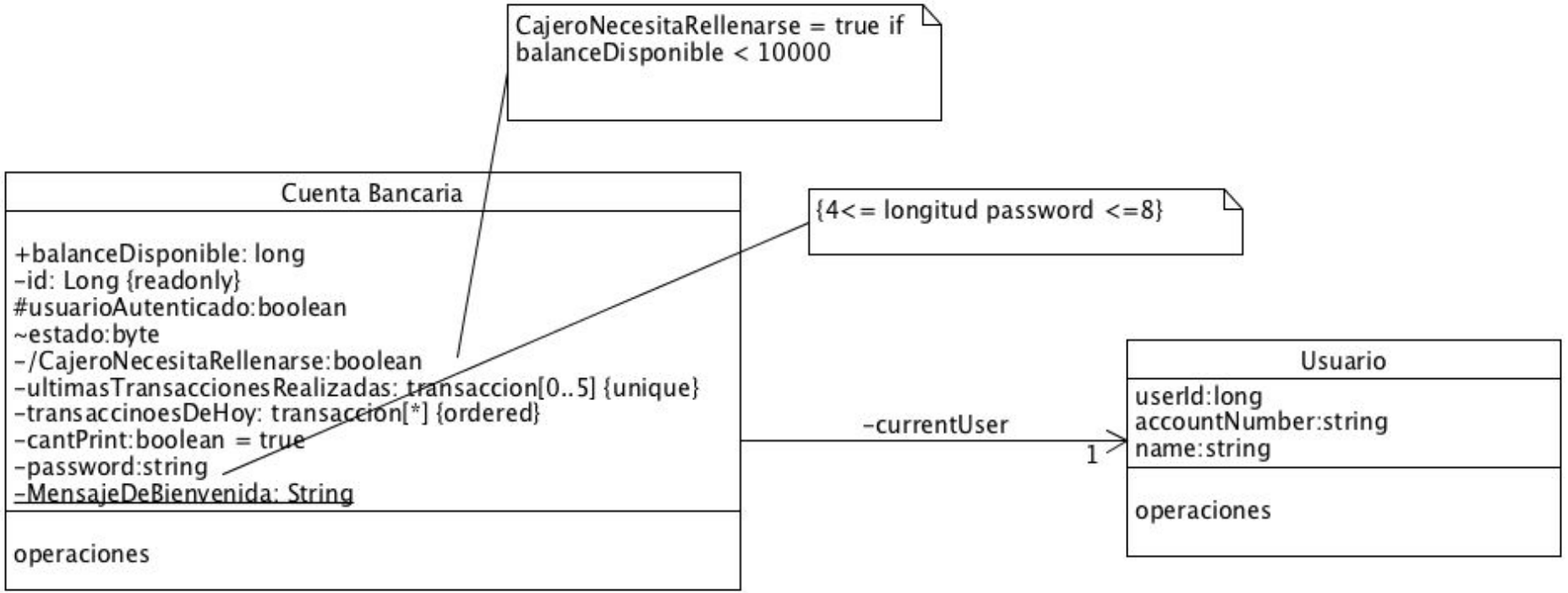
Es decir, si un atributo es estático los objetos no tendrán cada uno un valor en este atributo, sino que todos los objetos tendrán el mismo valor en este atributo.



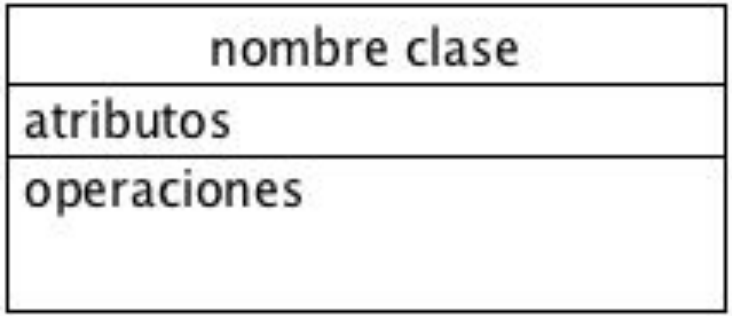
Notas de las relaciones

Soporta la misma sintaxis que las notaciones inline

Para que el diagrama sea más claro, se pueden colocar las propiedades y constraints en una nota.



Operaciones en las clases

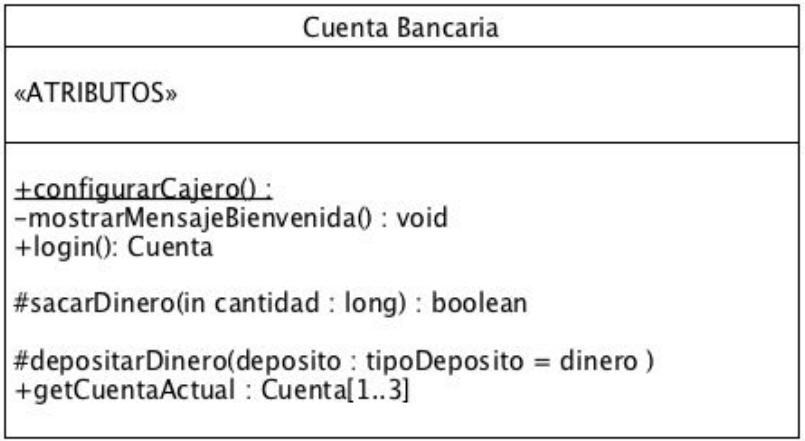


Operaciones

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

donde los parámetros serán:

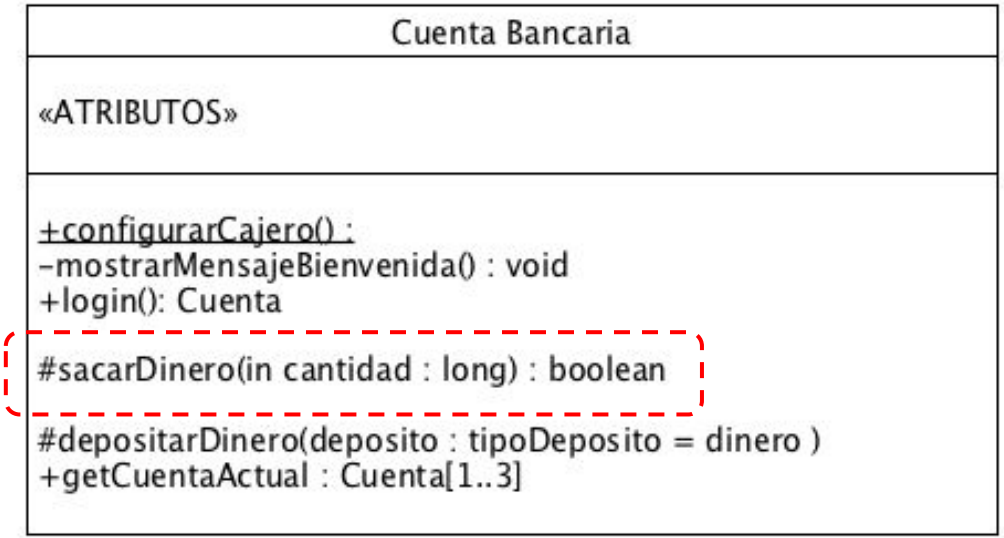
direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}



Visibilidad

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

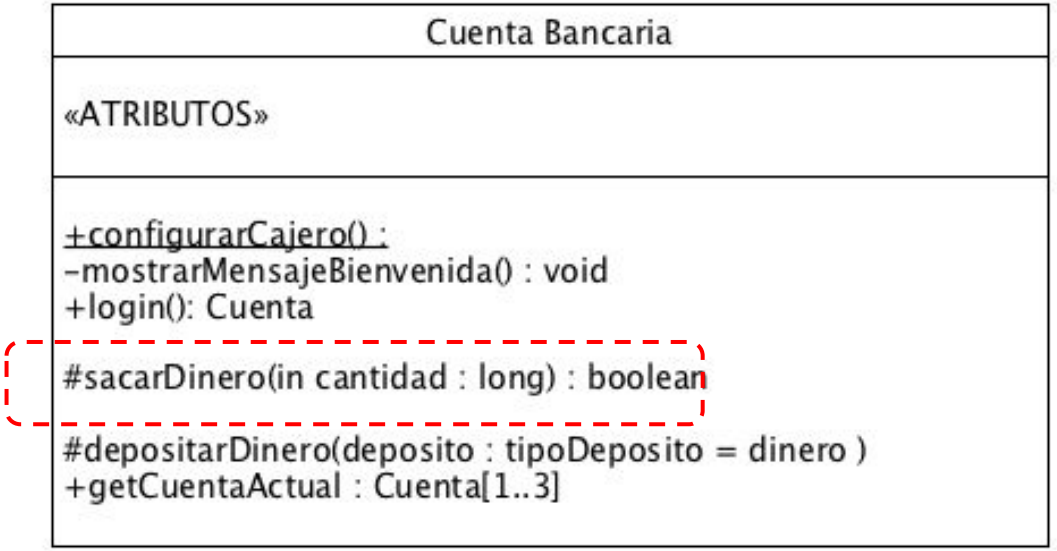
Funciona exactamente igual que en los atributos (public, private, protected y package)



Nombre

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

Será el nombre que recibe la operación



Parámetros

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

Aquí indicamos los parámetros que va a tener la clase.

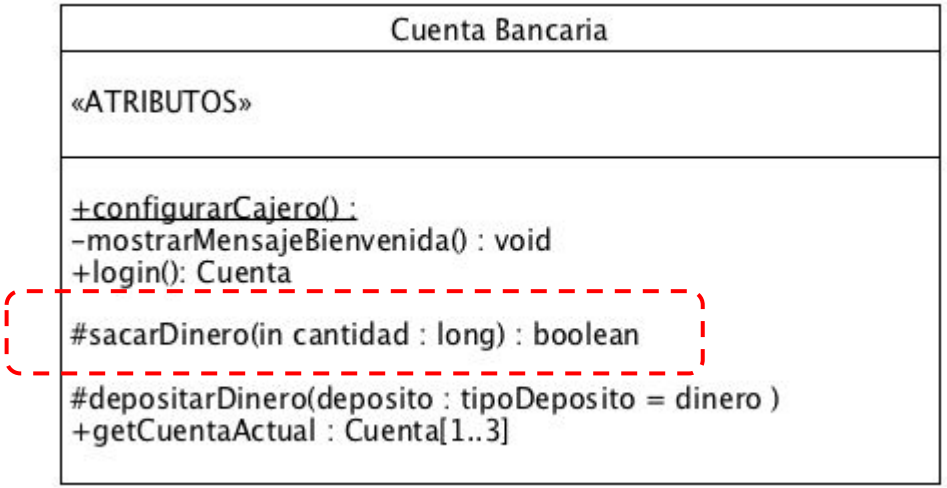
Dirección del parámetro

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Puede ser:

- in: un parámetro de entrada (no puede ser modificado)
- out: un parámetro de salida, sirve para com
- inout un parámetro de entrada y salida
- return: un valor de retorno del llamante

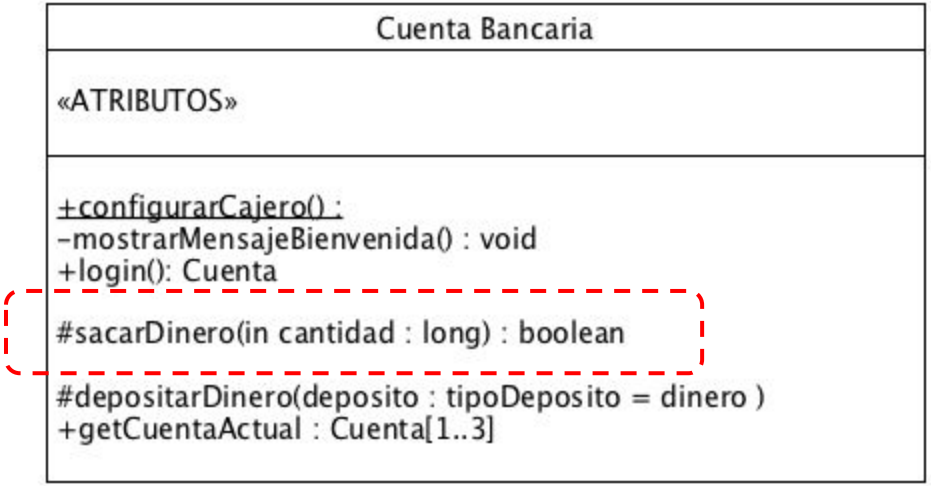


nombre del parámetro

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Indica el nombre del parámetro



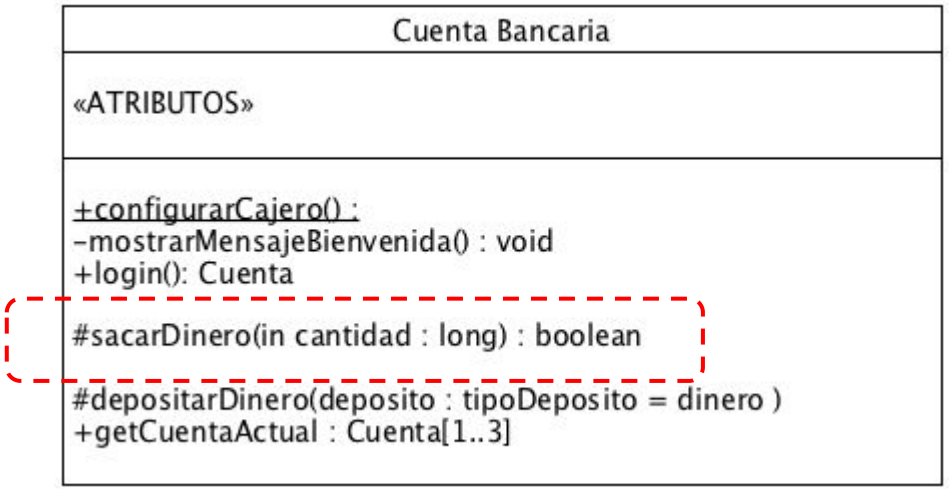
tipo del parámetro

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Nos indica el tipo del parámetro

- Tipo primitivo (int, long, float, string)
- Otra clase
- Una interfaz
- Una colección

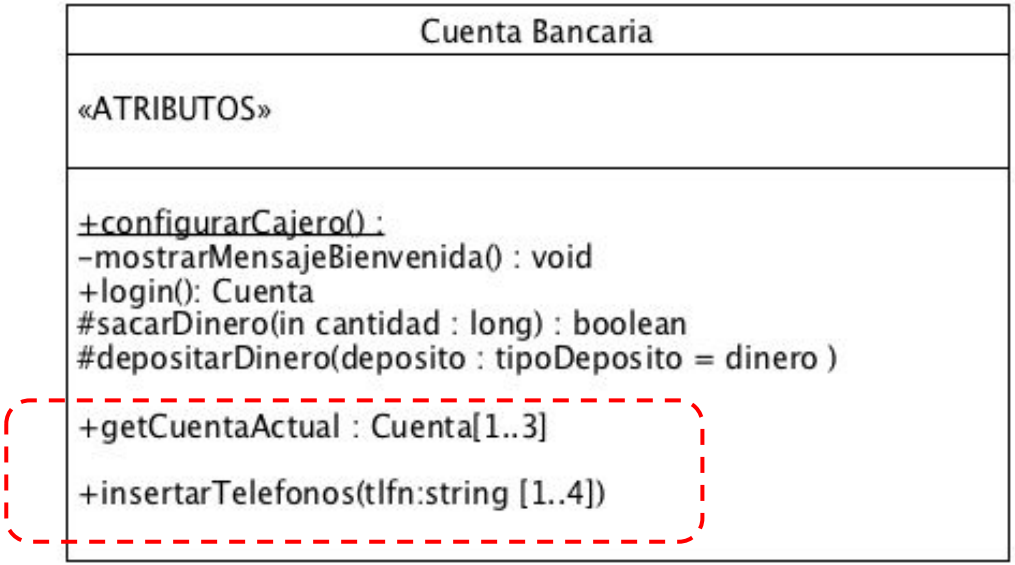


Multiplicidad

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Nos indica el numero de instancias del parámetro

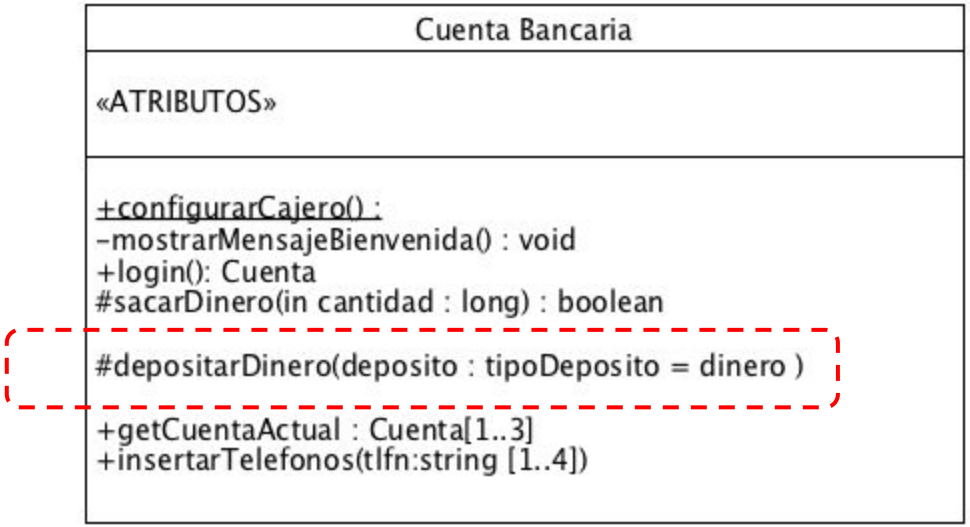


Valor por defecto

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Valor que toma por defecto un parámetro.

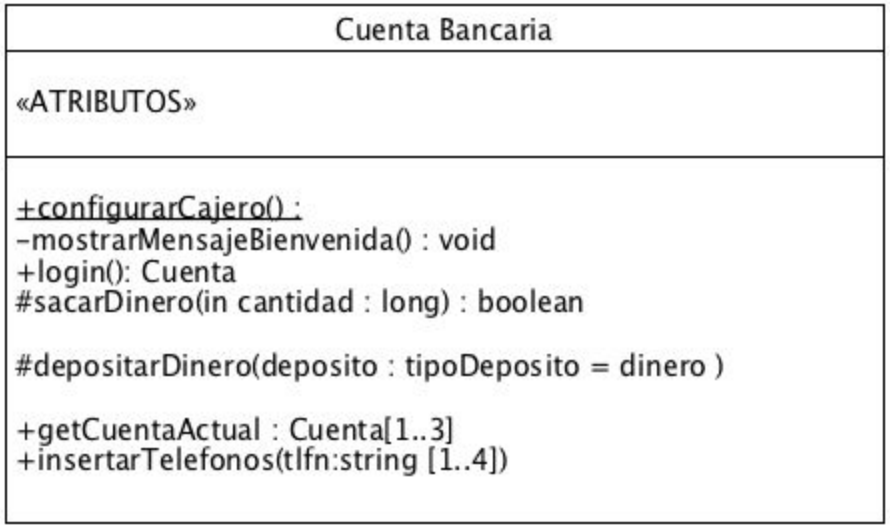


propiedades del parámetro

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

direccion nombre_parametro : tipo [multiplicidad] = valor_por_defecto {propiedades}

Propiedades adicionales como pueden ser readOnly, unique, ordered,...

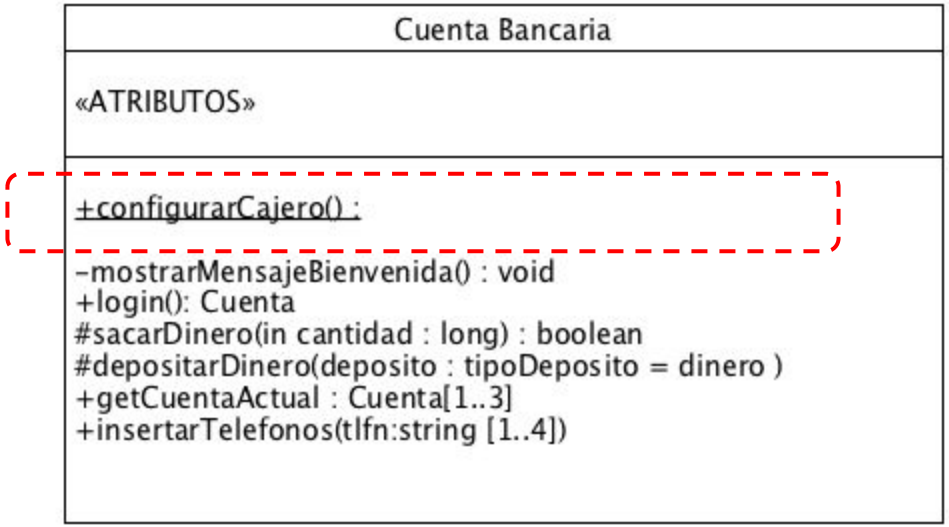


Operaciones estáticas (static operations)

Las operaciones estáticas van subrayados.

Una operación estática implica que su acción es compartida por todos los objetos de la clase.

Se podrá usar la operación sin necesidad de instanciar un objeto de la clase.

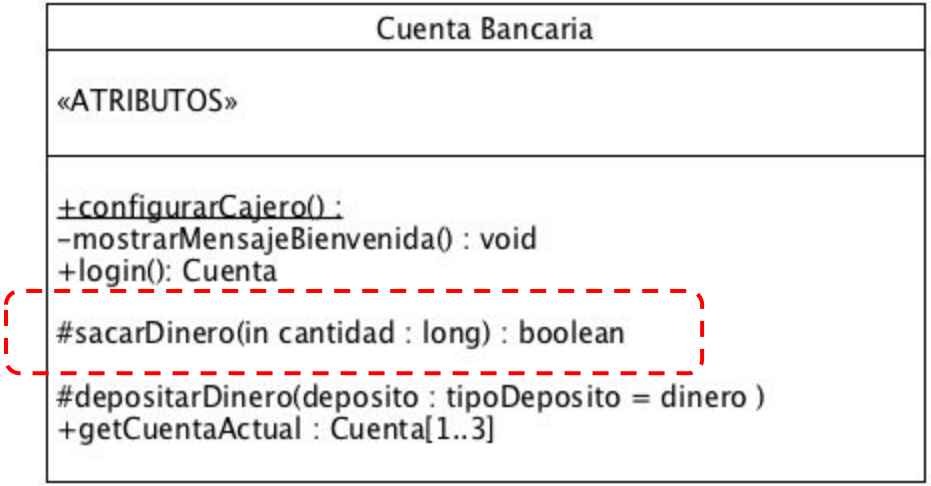


Tipo de retorno

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

El tipo de objeto que retorna la operación.

En caso de no devolver nada usaremos void



propiedades (precondiciones y postcondificiones)

visibilidad nombre (parámetros) : tipo_de_retorno { propiedades }

Podemos especificar precondition y postconditions.

En este ejemplo aplicamos una precondición a la operación de sacar dinero.

No se podrá sacar dinero si el saldo de la cuenta del usuario es menor a la cantidad pedida o si el total disponible en cajero es menor que la cantidad pedida