

# Objetos y Clases

# ¿Qué son objetos?

## ■ UML Reference Manual

- ◆ Entidad discreta con unos límites bien definidos que encapsula estado y comportamiento; una instancia de una clase

## ■ Más sencillo

- ◆ Conjunto de datos y funciones relacionados relacionados entre sí, empaquetados en una unidad que puede ser reutilizable.
  - Citroen C3 azul con matricula 1909 DDF
    - Datos → color, matricula, gasolina,
    - Funciones → arrancar, girar, acelerar, ...

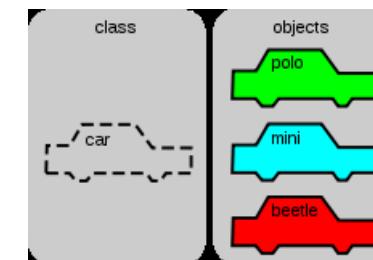
# ¿Qué son objetos?

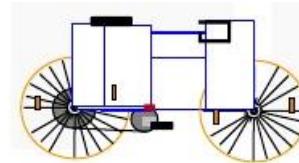
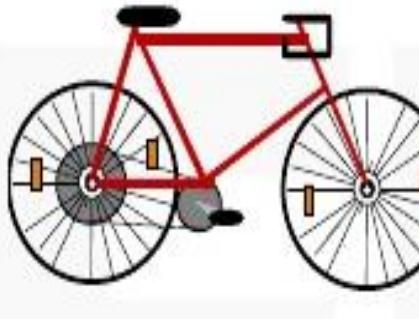
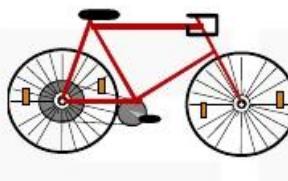
## ■ Instancias de una **clase**

- ◆ Define un conjunto de características (atributos y operaciones) que son comunes a todos las instancias

## ■ Un objeto tiene

- ◆ Atributos : datos
- ◆ Operaciones (funciones): comportamiento





**Objetos bicicleta**

*instancias*



## CLASE BICICLETA

### Atributos

Tamaño del cuadro  
Tamaño de la llanta  
Material

### Operaciones

Frenar  
Mover  
Cambiar

# ¿Qué son objetos?

- Propiedades comunes a todos los objetos
  - ◆ Identidad
    - Cada objeto tiene su propia identidad (existencia única)
    - Identificador único
      - Coche → número de bastidor
  - ◆ Estado
    - Valor de los atributos de un objeto y de sus relaciones con otros objetos en un momento determinado
  - ◆ Comportamiento
    - Conjunto de operaciones que un objeto puede desarrollar
      - Pueden cambiar el estado de un objeto
      - Método → implementación de un comportamiento

# Encapsulación

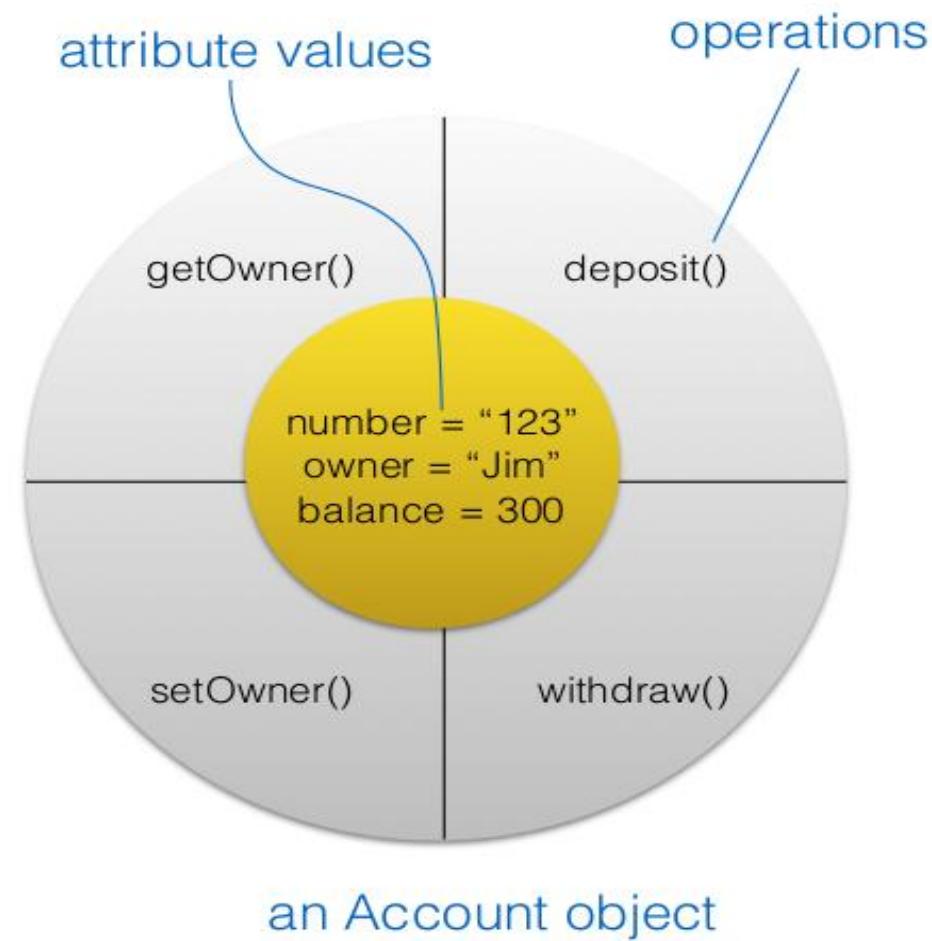
## ■ ¿Qué significa?

- ◆ Los datos (valores de los atributos) quedan escondidos dentro del objeto
  - ◆ Sólo se puede acceder a ellos a través de las operaciones

## ■ Ventajas

- ◆ Software más robusto y ampliable
  - ◆ No es necesario preocuparse por la estructura de los datos ocultos, sólo de lo que él objeto puede hacer.

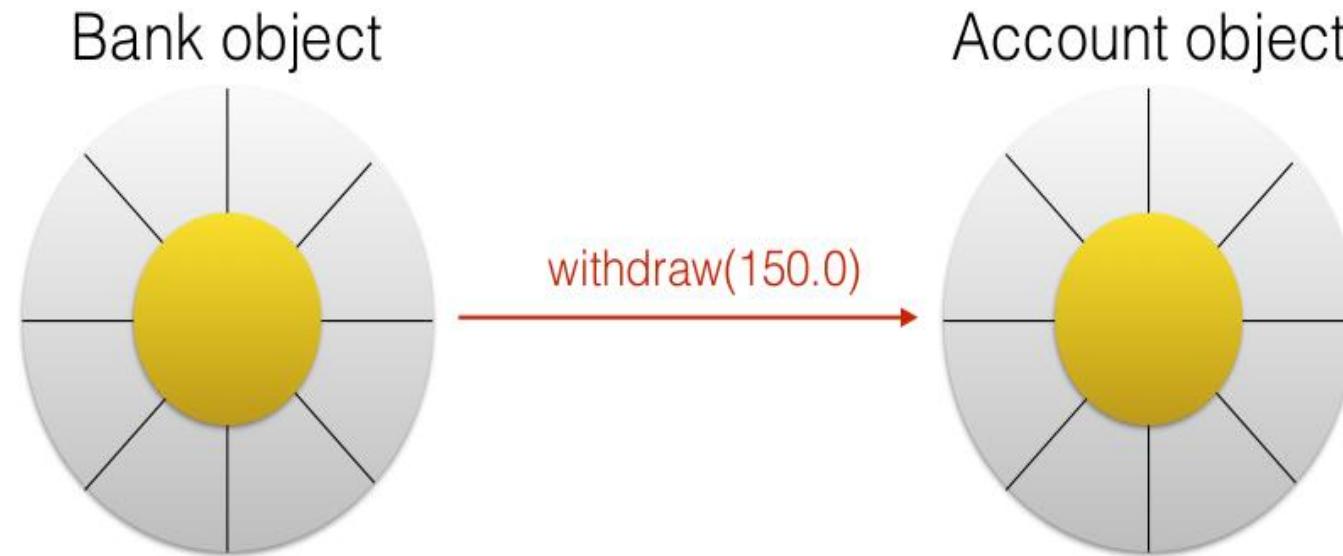
# Encapsulación



# Mensajería

- ¿Cómo agrupamos objetos para crear sistemas software?
  - ◆ Los objetos colaboran unos con otros
    - Realizar las funciones del sistema
  - ◆ Establecen vínculos
    - Envían mensajes a otros objetos
  - ◆ Al recibir un mensaje
    - Invoca una operación
- Estructura en tiempo de ejecución SOO
  - ◆ Conjunto de objetos que se crean, permanecen y se destruyen
    - Envían mensajes para invocar servicios

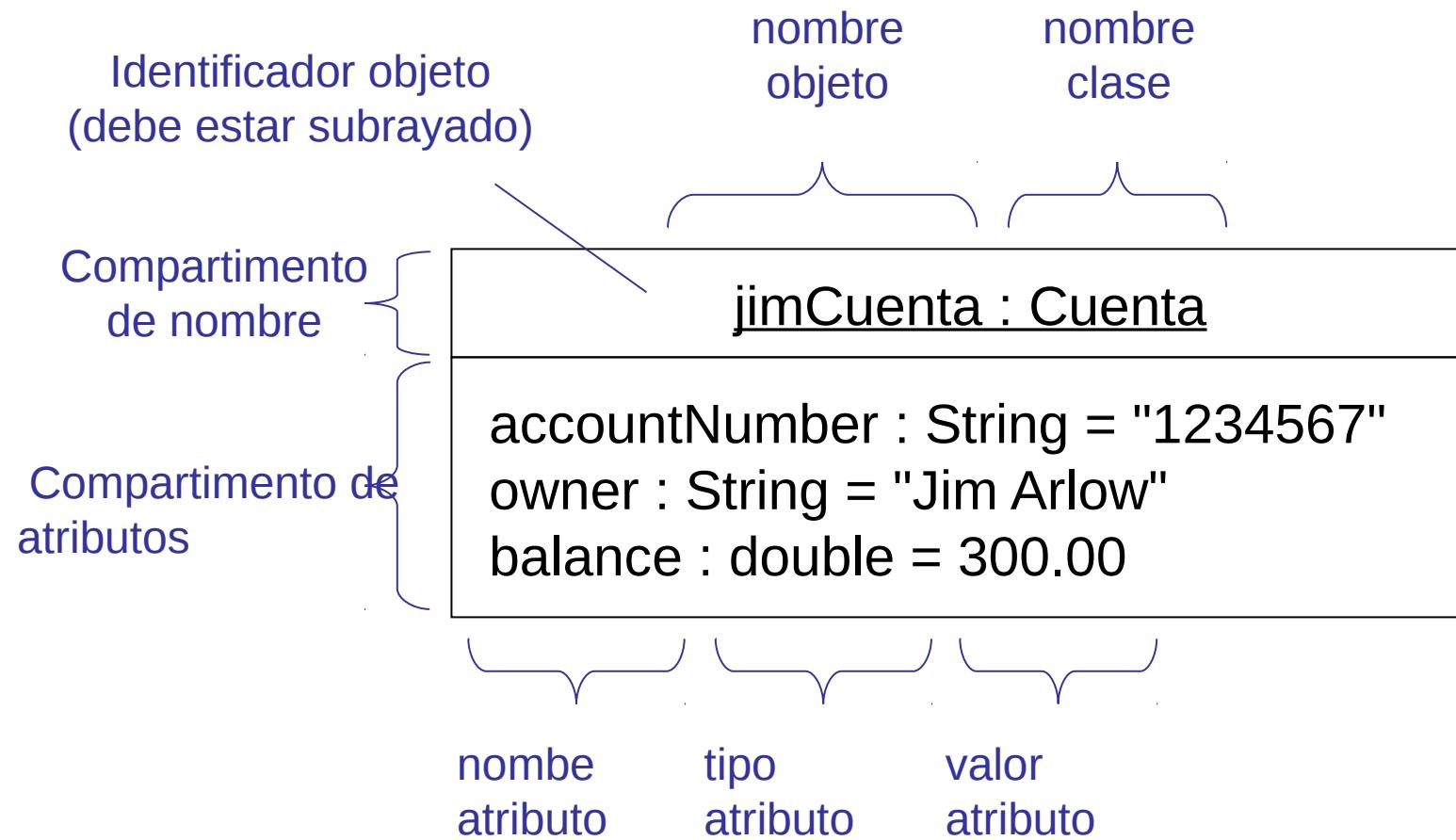
# Mensajería



the Bank object sends the message  
“withdraw 150.00” to an Account object

the Account object responds by invoking  
its `withdraw()` operation with the parameter  
150. This operation decrements the  
account balance by 150.00

# Sintaxis de objetos en UML



# Sintaxis de objetos en UML

## ■ Variantes identificador objeto

- ◆ El nombre del objeto:nombre de la clase
  - jimCuenta:Cuenta
  - Existe un objeto denominado jimCuenta que es una instancia de la clase Cuenta
- ◆ Solamente el nombre de la clase
  - :Cuenta
  - Objeto anónimo
  - Se está examinando una instancia genérica
- ◆ Solamente el nombre del objeto
  - jimCuenta
  - No identificamos la clase. No se han descubierto todas

# Sintaxis de objetos en UML

## ■ Operaciones

- ◆ Todos los objetos de una clase tienen el mismo conjunto de operaciones y atributos
- ◆ Se muestran en el diagrama de clases, no aquí

## ■ Atributos

- ◆ Se pueden mostrar opcionalmente
  - nombre:tipo=valor
- ◆ Los tipos se suelen omitir para simplificar

## ■ Nomenclatura

- ◆ Objetos y atributos → lowerCamelCase
- ◆ Clases → UpperCamelCase

# ¿Qué son clases?

## ■ UML Reference Manual

- ◆ Descriptor para un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamientos

## ■ Más sencillo

- ◆ Descriptor para un conjunto de objetos que tienen las mismas características



# ¿Qué son clases?

## ■ Clase

- ◆ Permite modelar conjuntos de objetos que tienen el mismo conjunto características
  - La clase determina la estructura (conjunto de características) de todos los objetos de la clase
  - Los objetos de una clase tienen el mismo conjunto de operaciones y atributos, pero diferentes valores de atributos
  - Los objetos responderán a los mensajes al invocar las operaciones
  - Dependiendo de su estado, responden de diferentes maneras

# ¿Qué son clases?

- Clasificador-instanciación
  - ◆ Clase-objeto
- Objetos
  - ◆ Valores específicos para los atributos definidos
  - ◆ Responderán a los mensajes al invocar las operaciones
    - Dependiendo de su estado, responden de diferentes maneras

# ¿Cuántas clases?

- La clasificación es la forma más importante de ordenar la información sobre el mundo



# Clases y objetos

- ¿Cómo se relacionan clases y objetos?
  - ◆ Relación <<instantiate>>
  - ◆ Relación de dependencia (----->) estereotipada
    - Significado especial
- Relación
  - ◆ Conexión entre elementos de un modelo
- Relación dependencia
  - ◆ Relación entre dos elementos en los que el cambio en un elemento (*proveedor*) puede afectar o proporcionar información necesaria para el otro elemento (*cliente*)

# Clases y objetos

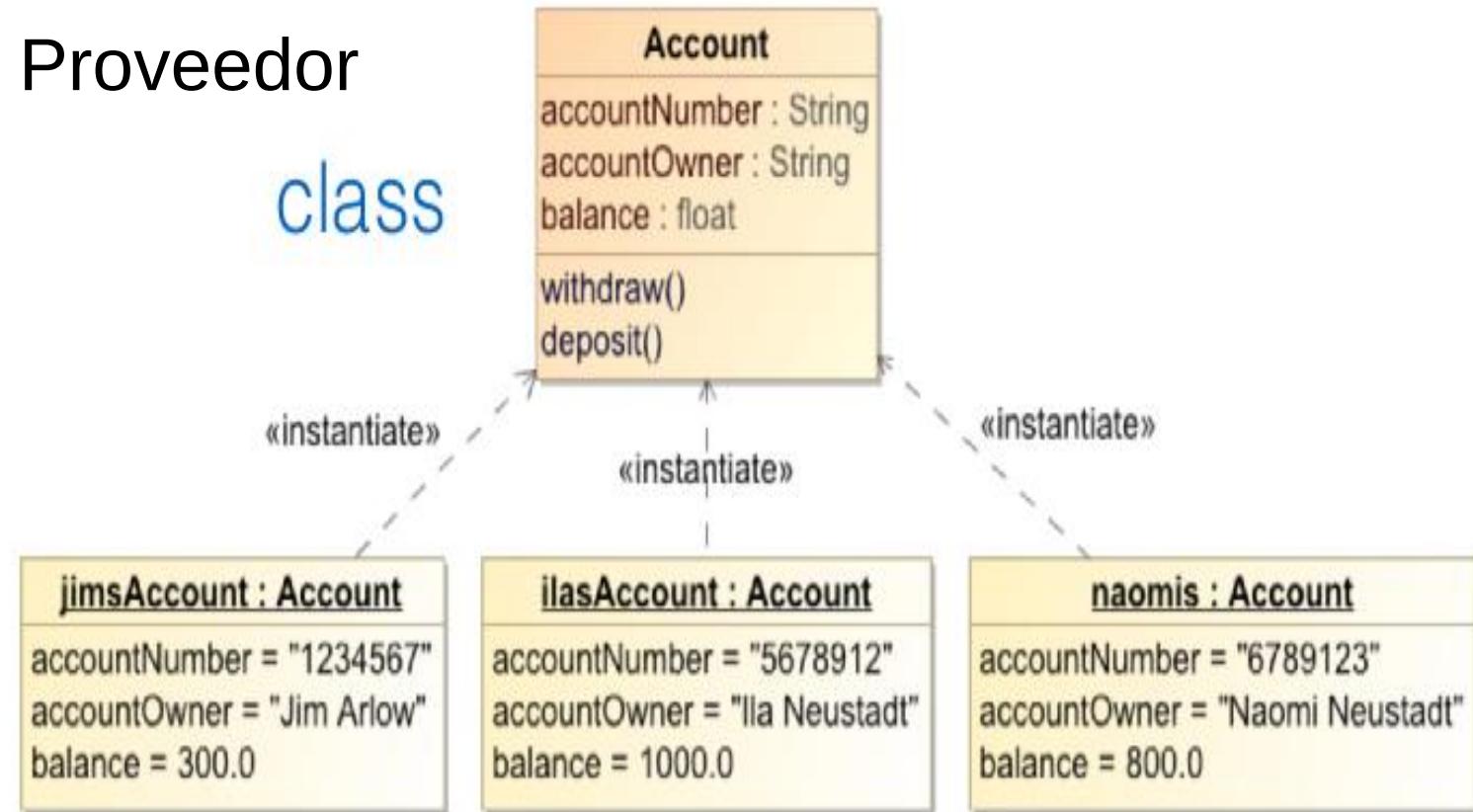
## ■ Estereotipada

- ◆ << ... >>
- ◆ Forma de personalizar elementos de modelado
- ◆ Crear variaciones con nueva semántica
- ◆ << instantiate >>
  - Convierte la relación de dependencia ordinaria en una relación de instanciación entre clase y objetos

# Clases y objetos

objects

Proveedor  
class



Cliente

# Clases y objetos

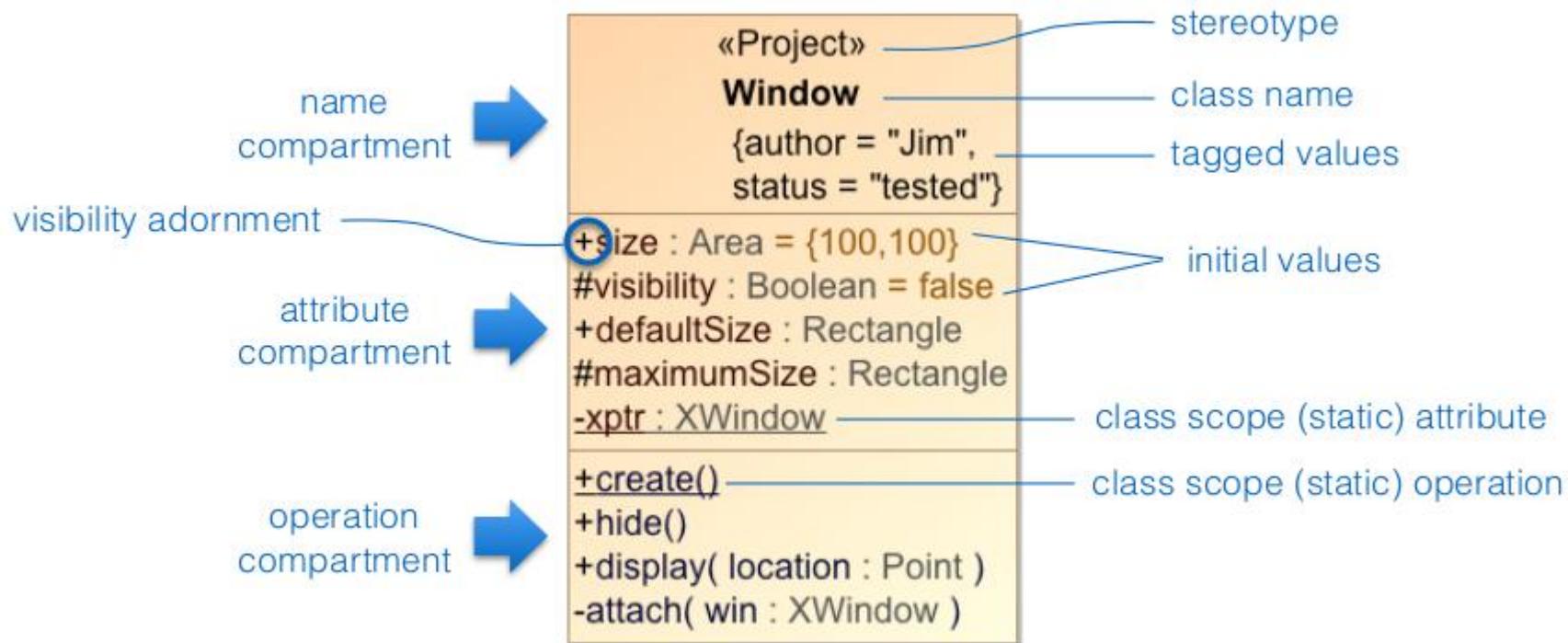
## ■ Instanciar

- ◆ Crear una instancia específica de algo a partir de una plantilla
- ◆ Los objetos son instancias de una clase

## ■ ¿Cómo crear instancias?

- ◆ Operaciones especiales: Constructores
  - Pertenecen a la clase, no a los objetos
- ◆ Se le asigna memoria al nuevo objeto
- ◆ Se le asigna una identidad única
- ◆ Se establecen valores iniciales para los atributos
- ◆ Se establecen vínculo con otros objetos

# Sintaxis de clases en UML



## ■ Nomenclatura

- ◆ UpperCamelCase
- ◆ Usar nombres descriptivos
- ◆ Evitar abreviaciones

# Sintaxis de clases en UML

- ¿Hay que mostrarlo todo?
  - ◆ Compartimento del nombre → obligatorio
  - ◆ Resto → opcionales
- ¿Qué mostrar?
  - ◆ Dependerá de la finalidad del diagrama
    - Solo relaciones → compartimento con el nombre
    - Comportamiento → compartimento operaciones
    - Datos → compartimento atributos
  - ◆ Modelo análisis
    - Nombre de la clase. Estereotipos si trascendencia
    - Atributos y operaciones clave

# Sintaxis de clases en UML

## ■ Compartimento de atributos

```
<visibilidad> <nombre>:<tipo>[multiplicidad]=[valorInicial]
```

- ◆ Todo es opcional salvo el nombre
- ◆ <valorInicial> es el valor que se le asigna al atributo cuando se crea un objeto (se instancia)
- ◆ Nomenclatura:
  - lowerCamelCase
  - Nombres descriptivos
  - Evitar abreviaciones
- ◆ Pueden llevar delante un estereotipo y detrás un conjunto de valores etiquetados

# Sintaxis de clases en UML (Compartimento atributos)

## ■ Visibilidad

- ◆ Adorno que se aplica a atributos y operaciones
- ◆ No se utiliza en el modelo de análisis

| Adorno | Nombre    | Semántica  |
|--------|-----------|--|
| +      | pública   | Cualquier elemento que pueda acceder a la clase puede acceder a cualquiera de sus características publicas |
| -      | privada   | Solamente las operaciones dentro de la clase pueden acceder a características privadas                     |
| #      | protegida | Sólo las operaciones dentro de la clase o dentro de sus hijos pueden acceder a carac. protegidas           |
| ~      | paquete   | Cualquier elemento que esté en el mismo paquete o paquete anidado puede acceder                            |

# Sintaxis de clases en UML (Compartimento atributos)

## ■ Tipo

- ◆ Otra clase o tipo primitivo
- ◆ 4 tipos primitivos en UML
  - Integer (Entero)
    - [Número entero](#)
  - NaturalIlimitado
    - [Número entero  \$\geq 0\$ .](#)
    - Infinito se muestra como \*
  - Boolean (Booleano)
    - Verdadero o falso
  - String (Cadena)
    - Secuencia de caracteres. Entre comillas “ ”

# Sintaxis de clases en UML (Compartimento atributos)

- Tipo
  - ◆ Se pueden utilizar los tipos de un lenguaje concreto
    - CUIDADO!!! Queda ligado a ese lenguaje
    - No es recomendable
  - ◆ Añadir tipos primitivos
    - Crear una clase con el mismo nombre que el tipo primitivo que se quiere crear
    - Estereotipo <>primitive><
    - La clase no tiene atributos ni operaciones

# Sintaxis de clases en UML (Compartimento atributos)

## ■ Multiplicidad

- ◆ Nos permite modelar colecciones de elementos
- ◆ Restricciones sobre el número de elementos que participan en una relación



- ◆ **[0 .. 1]** indica que el atributo puede tomar el valor **null**

# Sintaxis de clases en UML

## ■ Comportamiento de operaciones

### ◆ Operación

- Función asociada a una clase en particular
- Signature
  - nombre, lista de parámetros, tipo
  - Identidad de la operación
  - Sirve para determinar si hay que invocar una operación

```
<visibilidad><nombre>(<propósito><nombreP>:<tipoP>=<default> ) :<tipoRetorno>
```



Lista de parámetros

# Sintaxis de clases en UML (Compartimiento de operaciones)

## ■ Nomenclatura

- ◆ Operaciones

- LowerCamelCase
- Verbos o frases que indiquen acción
- Evitar símbolos especiales y abreviaturas

- ◆ Parámetros

- LowerCamelCase
- Nombres o sintagmas nominales

## ■ Tipo de retorno

- ◆ Las operaciones pueden devolver varios objetos
- ◆ Más de un tipo de retorno

# Sintaxis de clases en UML (Compartimiento de operaciones)

## ■ Propósito del parámetro

- ◆ **in** (default)
  - El parámetro es una entrada a la operación
  - No puede ser modificado por ella
- ◆ **out**
  - El parámetro sirve como repositorio para la salida
- ◆ **inout**
  - El parámetro es una entrada a la operación
  - Puede ser modificado por la operación
- ◆ **return**
  - El parámetro es uno de los valores que devuelve la operación

# Sintaxis de clases en UML (Compartimiento de operaciones)

- Propósito del parámetro
  - ◆ Característica usada en un diseño muy detallado
- Ejemplos
  - ◆ `maximo(in a:Integer, in b:Integer): Integer`
  - ◆ `maxMin(in a:Integer, in b:Integer): Integer, Integer`
  - ◆ `maxMin(in a:Integer, in b:Integer, return max:Integer, return min:Integer)`

# Sintaxis de clases en UML (Compartimiento de operaciones)

- Valores por defecto de los parámetros
  - ◆ Se le asigna un valor por defecto al parámetro
  - ◆ Al invocar a la función, si no se le asigna ningún valor al parámetro, se utiliza el valor por defecto
  - ◆ Característica raramente usada en el análisis

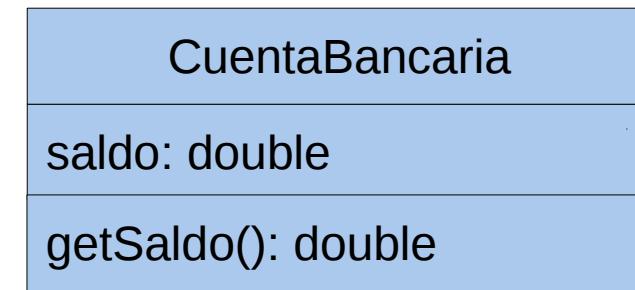
Canvas

```
dibujarCirculo(origen: Point = Point(0,0), radio: Integer)
dibujarCuadrado (origen: Point = Point(0,0), radio:Integer)
```

# Sintaxis de clases en UML (Compartimiento de operaciones)

## ■ Operaciones de consulta

- ◆ Operaciones que no modifican el estado del objeto sobre el que son llamadas
- ◆ Sus nombres suelen empezar por *get/obtener*



Devuelve el  
valor de saldo

# Ámbito

## ■ Definición

- ◆ Lugar de aplicación de los atributos y operaciones

## ■ Tipos

- ◆ Ámbito de instancia

- Atributos y operaciones que pertenecen y operan sobre un determinado objeto

- Pueden tener diferente valor/comportamiento cada uno

- ◆ Ámbito de clase

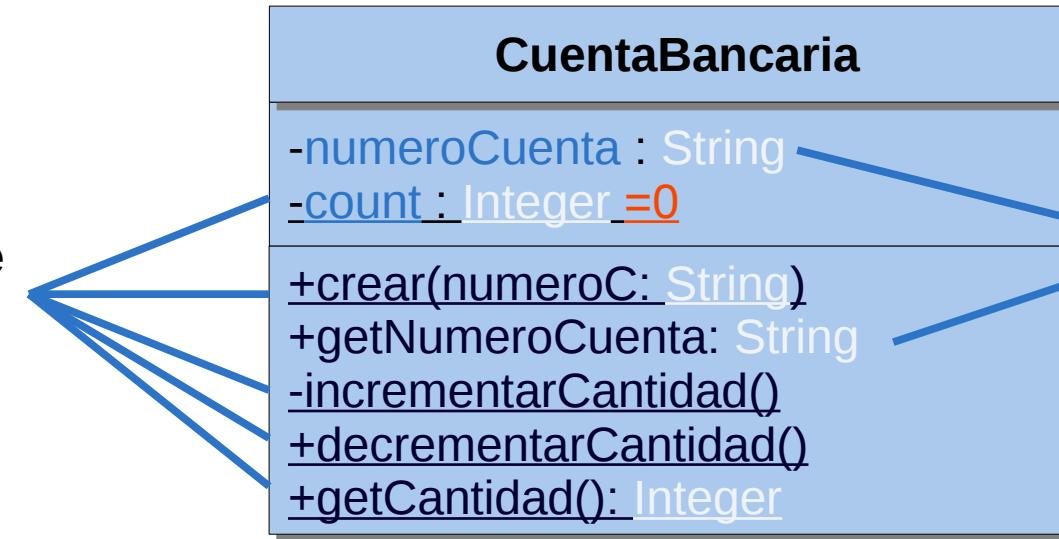
- Atributos y operaciones que pertenecen y operan sobre todos los objetos de una clase

- Mismo valor/comportamiento para todos

# Ámbito

ámbito clase  
(subrayado)

ámbito estancia



# Ámbito

|             | Ámbito de instancia  | Ámbito de clase  |
|-------------|--|--|
| Atributos   | <p>Por defecto, los atributos tienen ámbito de instancia</p> <p>Cada objeto de la clase tiene su propia copia de los atributos con ámbito de instancia</p> <p>Cada objeto tiene valores diferentes de los atributos</p>      | <p>Se pueden definir atributos con ámbito de clase</p> <p>Todos los objetos de la clase comparte la misma copia de los atributos con ámbito de clase</p> <p>Todos los objetos tienen el mismo valor para los atributos</p> |
| Operaciones | <p>Por defecto, las operaciones tienen ámbito de instancia</p> <p>Cada invocación de una operación se aplica un objeto específico</p> <p>No se puede invocar una operación a menos que se tenga una instancia disponible</p> | <p>Se pueden definir operaciones con ámbito de clase</p> <p>La invocación de una operación se hace sobre la clase misma</p> <p>Se puede invocar una operación incluso si no hay una instancia disponible</p>               |

# Ámbito

- El ámbito determina el acceso

- ◆ Las operaciones con ámbito de instancias

- Otros atributos y operaciones con ámbito de instancia
  - Todas las operaciones y atributos con ámbito de clase

## ◆ Las operaciones con ámbito de clase

- ✓ Operaciones y atributos con ámbito de clase
  - ✗ Operaciones y atributos con ámbito de instancia
    - No hay instancias creadas
    - Si las hubiera, no sabría a cual.

# Construcción de objetos

## ■ ¿Cómo crear instancias de una clase?

### ◆ Constructores

- Operaciones con ámbito de clase
- Una clase puede definir varios
  - Mismo nombre
  - Se diferencian en los parámetros

### ◆ Constructor sin parámetros

- Constructor por defecto

### ◆ Constructor con parámetros

- Inicializan los atributos de los objetos

CuentaBancaria  
+crear(numeroC: String)

Sintaxis genérica

CuentaBancaria  
+CuentaBancaria(numeroC: String)

Sintaxis Java /C++

# Ejemplo

## MiembroClub

```
-numeroMiembro : String
-nombreMiembro: String
-numeroDeMiembros : Integer =0

+crear(numero: String, nombre: String)
+getNumeroMiembro(): String
+getNombreMiembro(): String
-incrementarNumeroDeMiembros()
+decrementarNumeroDeMiembros()
+getNumeroDeMiembros(): Integer
```

# Ejemplo

- Cada objeto de la clase `MiembroClub` tiene su propia copia del atributo `numeroMiembro`
- El atributo `numeroDeMiembros` existe sólo una vez y es compartido por todas las instancias de la clase `MiembroClub`
- Cuando se crea el atributo `numeroDeMiembros` se inicializa a cero
  - Supongamos que `crear` invoca a `incrementarNumeroDeMiembros` ¿Cuánto valdrá `numeroDeMiembros` después de crear tres objetos?
- ¿Cómo aseguramos que `numeroDeMiembros` es correcto después de que abandone un miembro?