UML

Juan Pablo Sandoval

Lo que quedo de UML

- *Diagrama de Clases*
- *Diagrama de Secuencia*

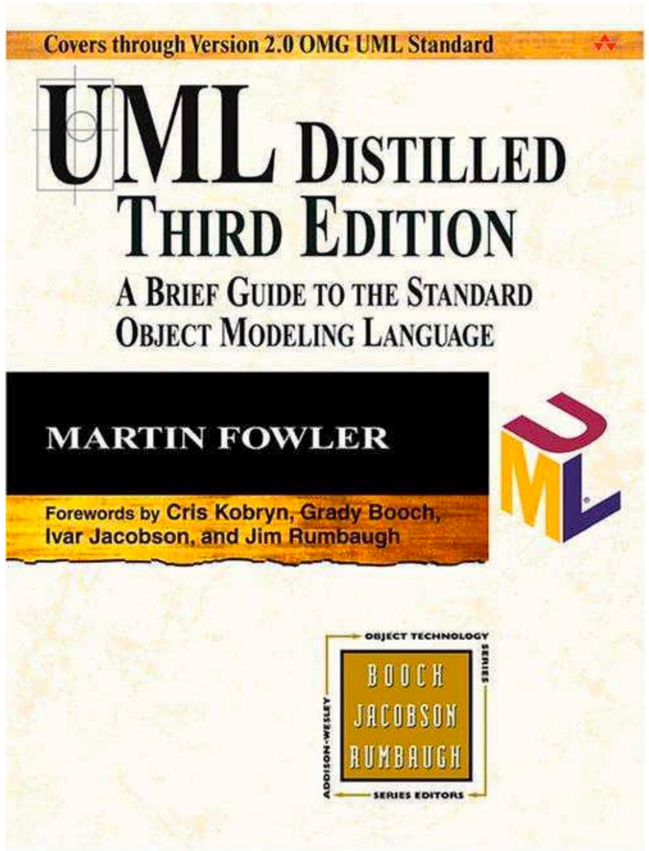
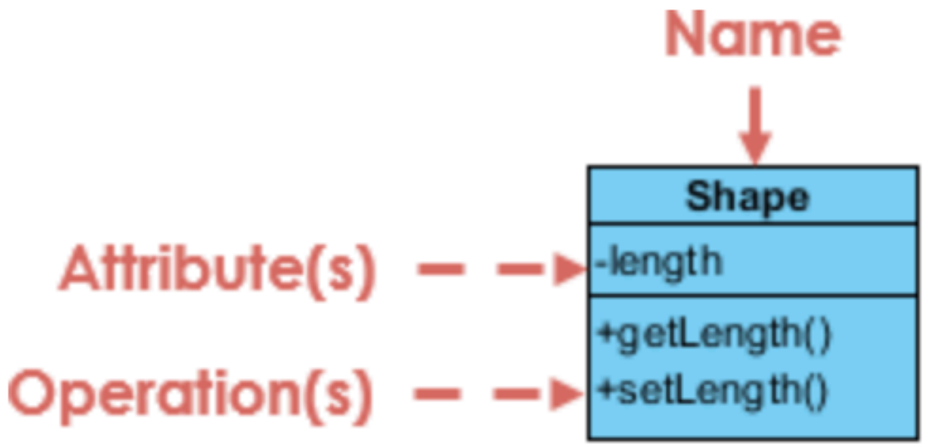


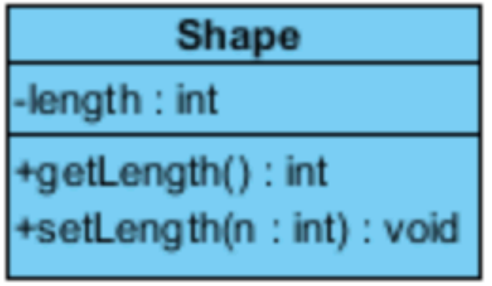
Diagrama de Clase

Juan Pablo Sandoval

Clase

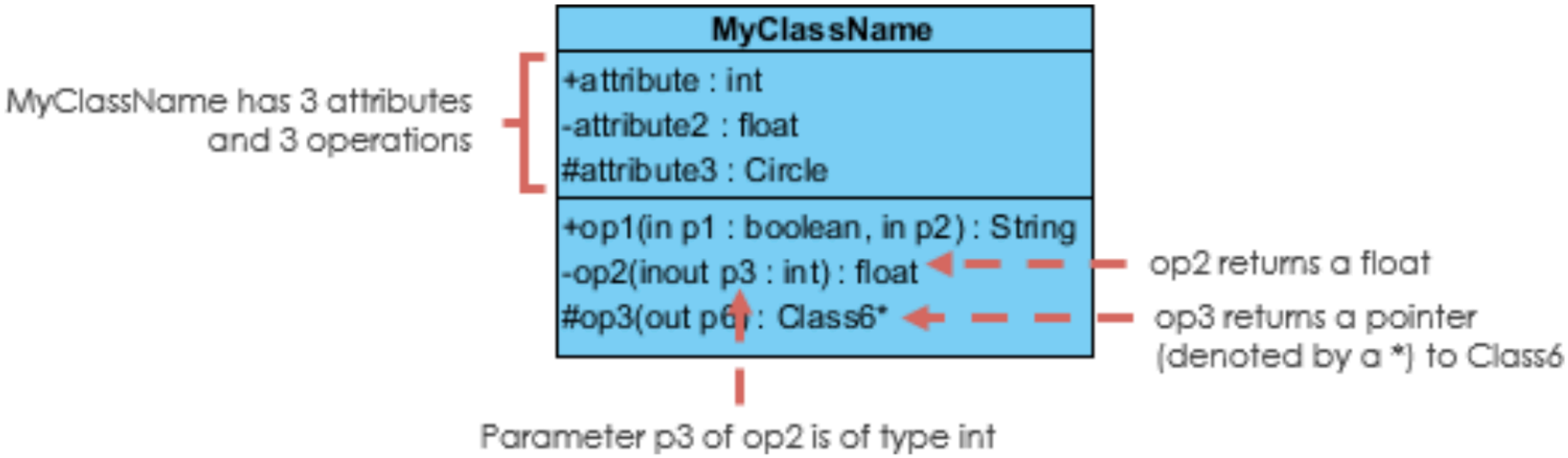


Class without signature

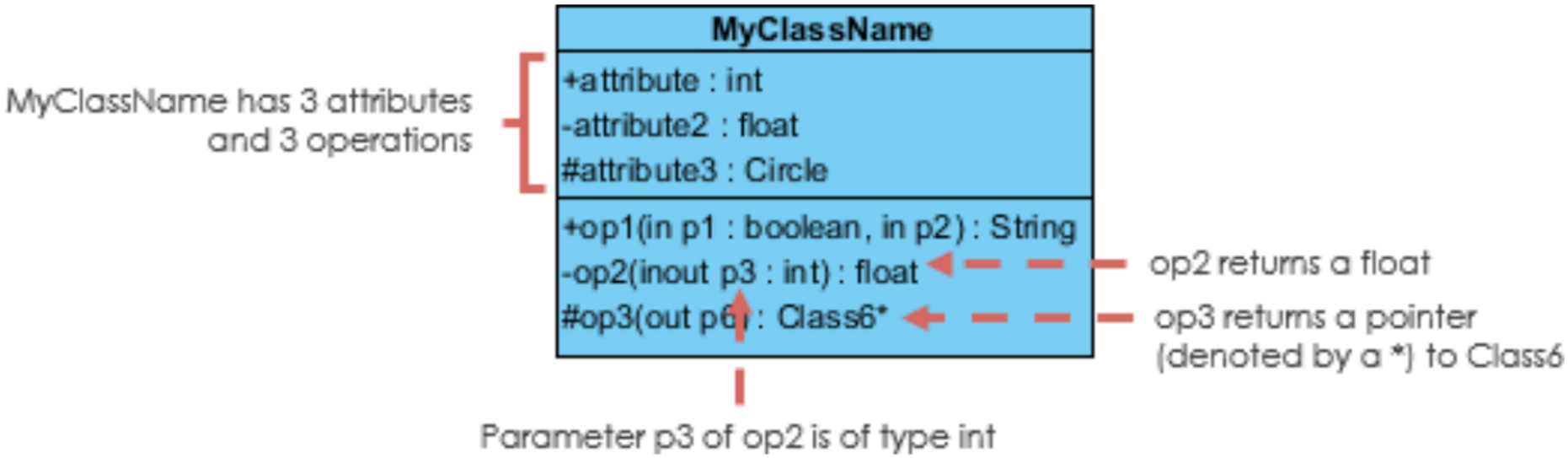


Class with signature

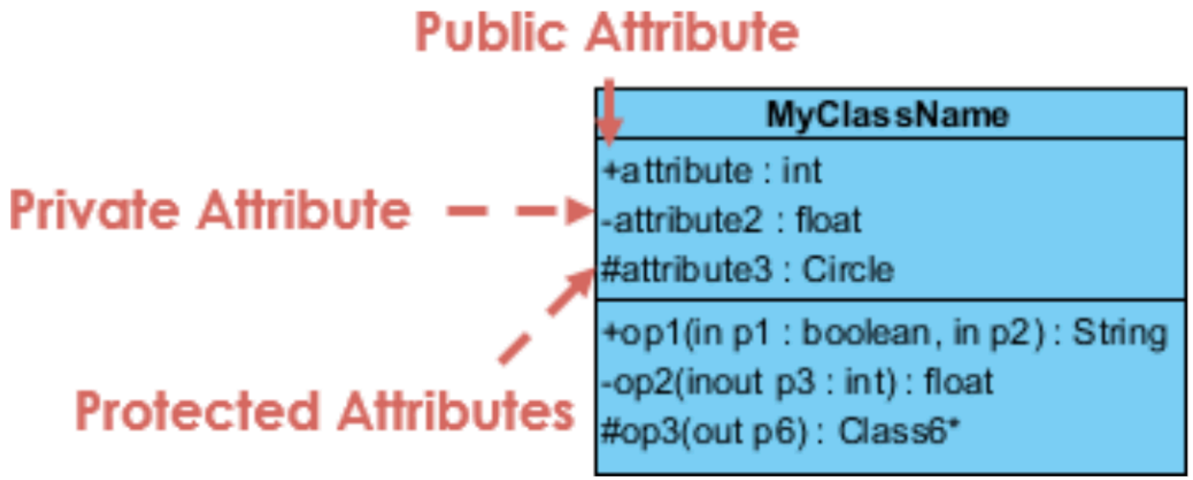
Clase


MyClassName
+attribute : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : boolean, in p2) : String
-op2(inout p3 : int) : float
#op3(out p6) : Class6*
MyClassName has 3 attributes and 3 operations
op2 returns a float
op3 returns a pointer (denoted by a *) to Class6
Parameter p3 of op2 is of type int

Clase

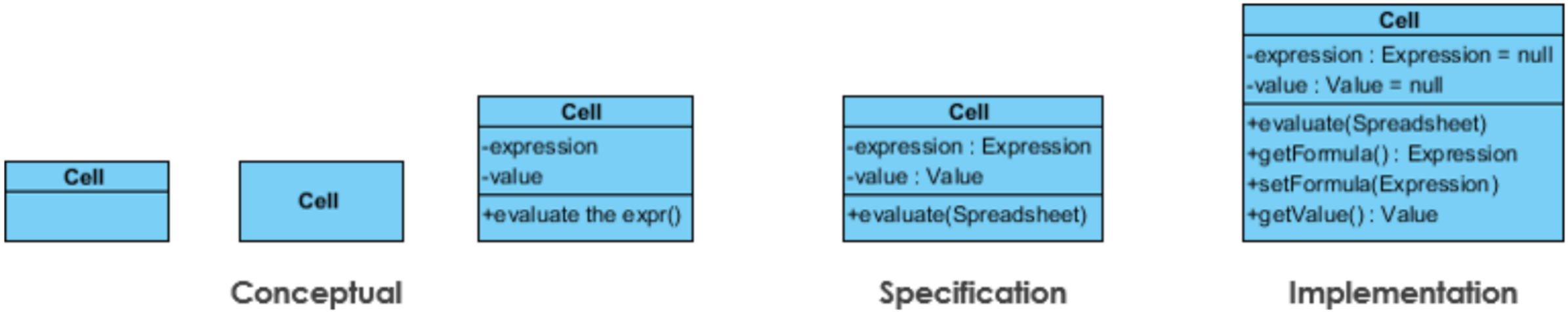

MyClassName
+attribute : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : boolean, in p2) : String
-op2(inout p3 : int) : float
#op3(out p6) : Class6*
MyClassName has 3 attributes and 3 operations
op2 returns a float
op3 returns a pointer (denoted by a *) to Class6
Parameter p3 of op2 is of type int

Clase



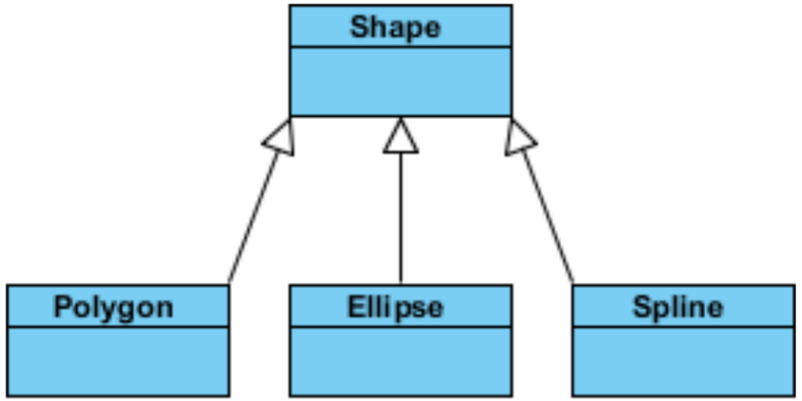
- ***Publico**: puede ser accedido desde cualquier clase del sistema.*
- ***Protegido**: solo puede ser accedido desde la misma clase y las clases hijas.*
- ***Privado**: solo puede ser accedido desde la misma clase.*

Clase

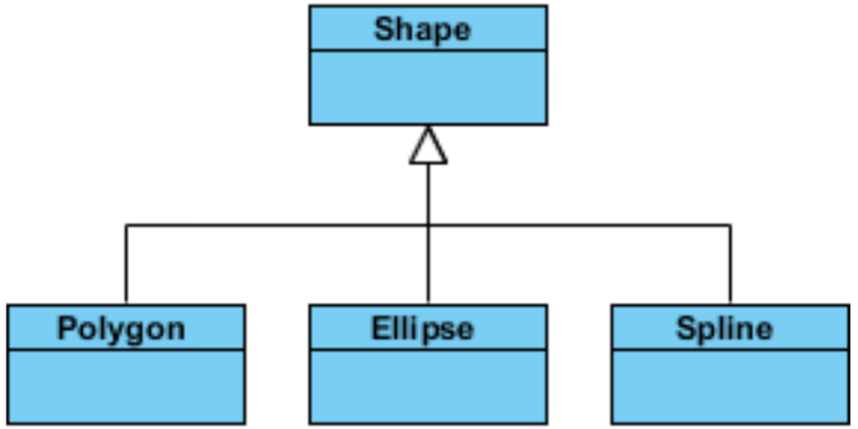


Para cada clase de puede poner diferente nivel de detalle, en la interrogación deben escribir con el mayor detalle posible.

Herencia



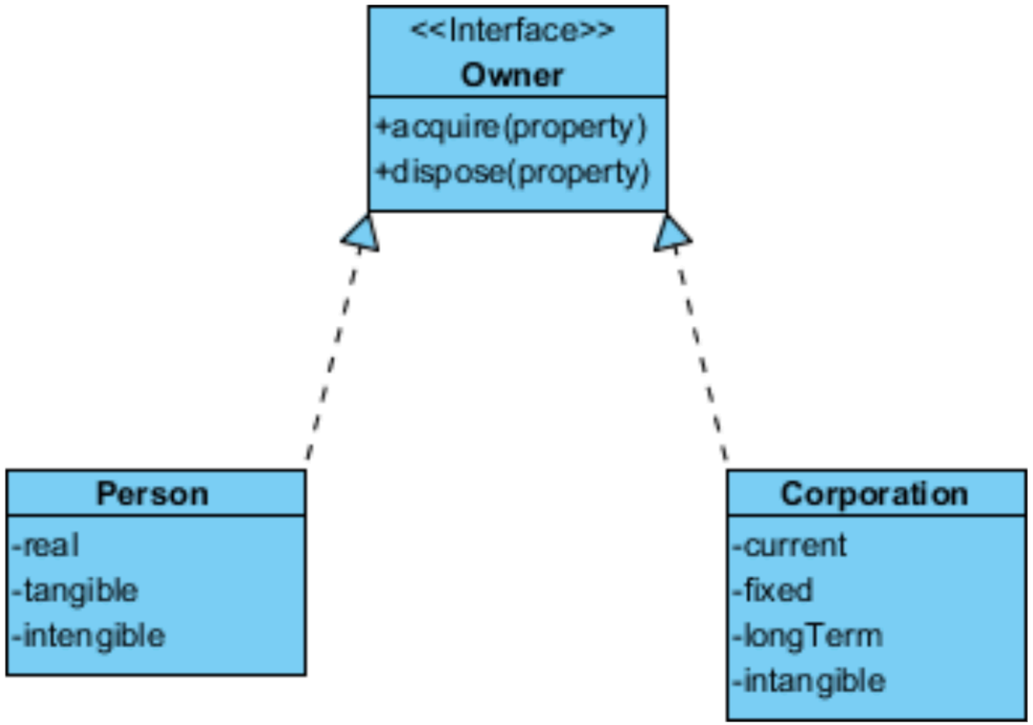
Style 1: Separate target



Style 2: Shared target

Se puede utilizar cualquier estilo de forma indiferente.

Realización



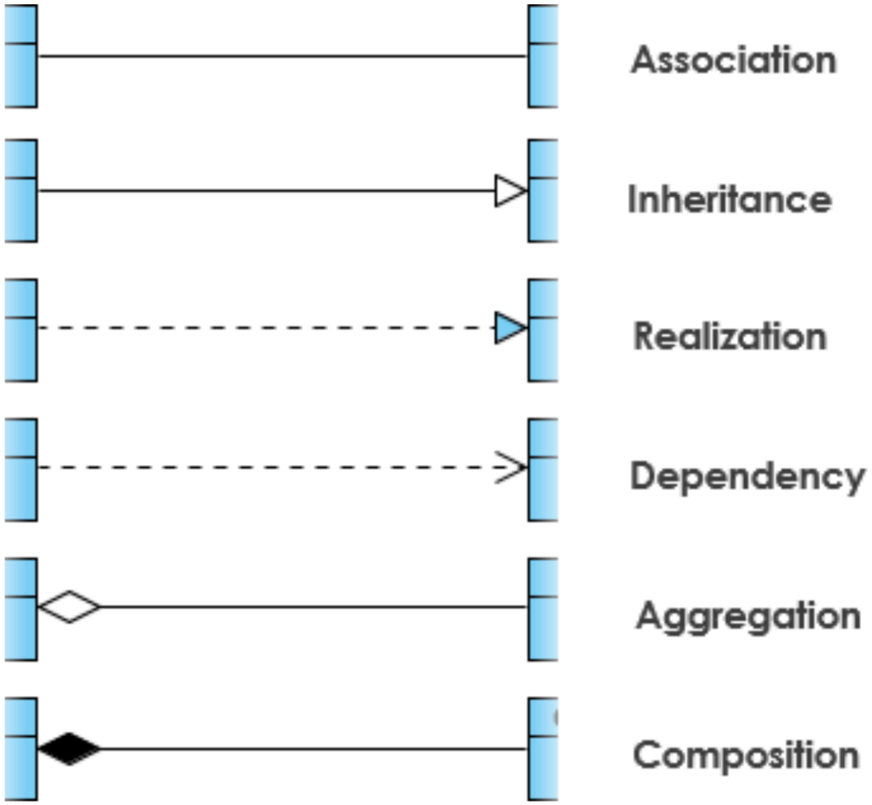
En ruby el equivalente a una interfaz es un “módulo”

Cardinalidad



Tipos de relación entre clase

- ***Association:*** *un objeto de la clase tiene una referencia a un objeto de otra clase por mucho tiempo. Por ejemplo, cómo atributo.*
- ***Inheritance:*** *una clase hereda de otra.*
- ***Realization:*** *una clase implementa una interfaz o modulo (en ruby).*
- ***Dependency:*** *un objeto de la clase tiene una referencia a un otro objeto de otra clase de forma temporal. Por ejemplo, como variable temporal o cómo argumento.*
- ***Aggregation:*** *rombo vacío, un objeto de una clase **A** esta compuesto de objetos de otra clase **B**. Los objetos de **B** no son creados dentro de la clase **A**.*
- ***Composition:*** *rombo lleno, un objeto de la clase **A** esta compuesto de objetos de otra clase **B**. Los objetos de la clase **B** son creados dentro de **A**, causando una dependencia más fuerte.*



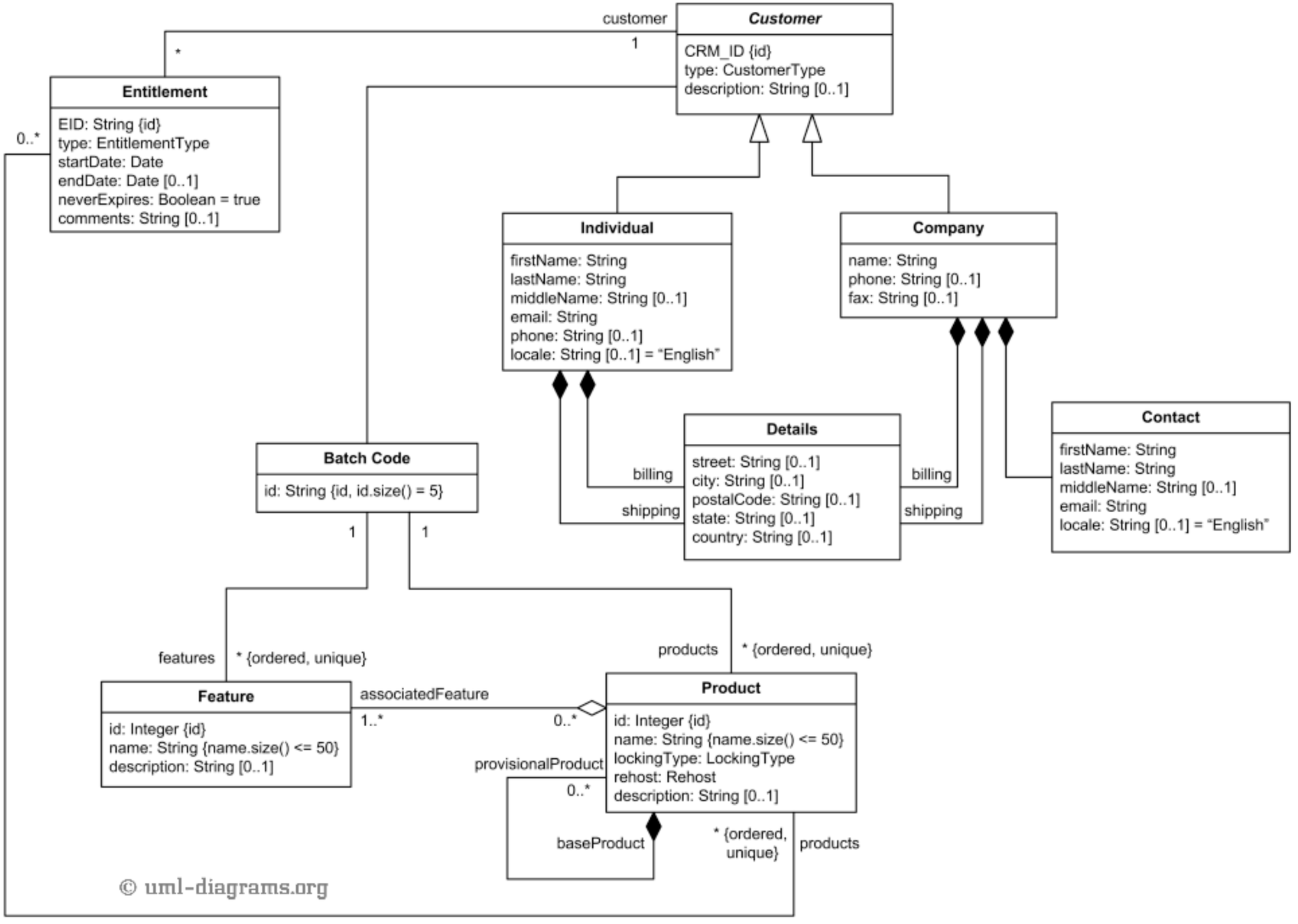

customer
1
*
Customer
CRM_ID {id}
type: CustomerType
description: String [0..1]
Entitlement
EID: String {id}
type: EntitlementType
startDate: Date
endDate: Date [0..1]
neverExpires: Boolean = true
comments: String [0..1]
0..*
Individual
firstName: String
lastName: String
middleName: String [0..1]
email: String
phone: String [0..1]
locale: String [0..1] = "English"
Company
name: String
phone: String [0..1]
fax: String [0..1]
Contact
firstName: String
lastName: String
middleName: String [0..1]
email: String
locale: String [0..1] = "English"
Details
street: String [0..1]
city: String [0..1]
postalCode: String [0..1]
state: String [0..1]
country: String [0..1]
billing
shipping
billing
shipping
Batch Code
id: String {id, id.size() = 5}
1
1
features
* {ordered, unique}
products
* {ordered, unique}
Feature
id: Integer {id}
name: String {name.size() <= 50}
description: String [0..1]
associatedFeature
1..*
0..*
Product
id: Integer {id}
name: String {name.size() <= 50}
lockingType: LockingType
rehost: Rehost
description: String [0..1]
provisionalProduct
0..*
baseProduct
* {ordered, unique}
products
© uml-diagrams.org

Diagrama de Secuencia

Juan Pablo Sandoval

Diagrama de Secuencia

- *Cada cajita es la instancia de una clase (un objeto).*
- *La línea punteada es la línea de vida de un objeto.*
- *Se lee de arriba hacia abajo.*

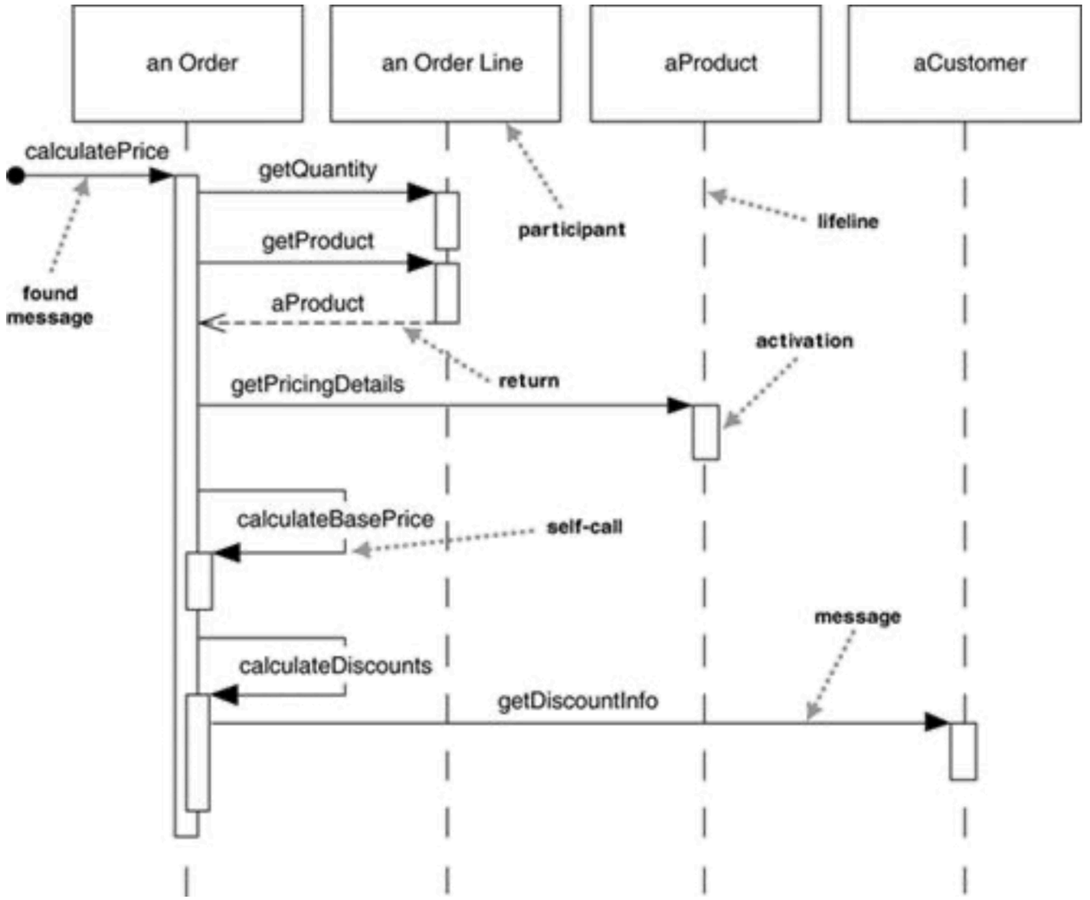
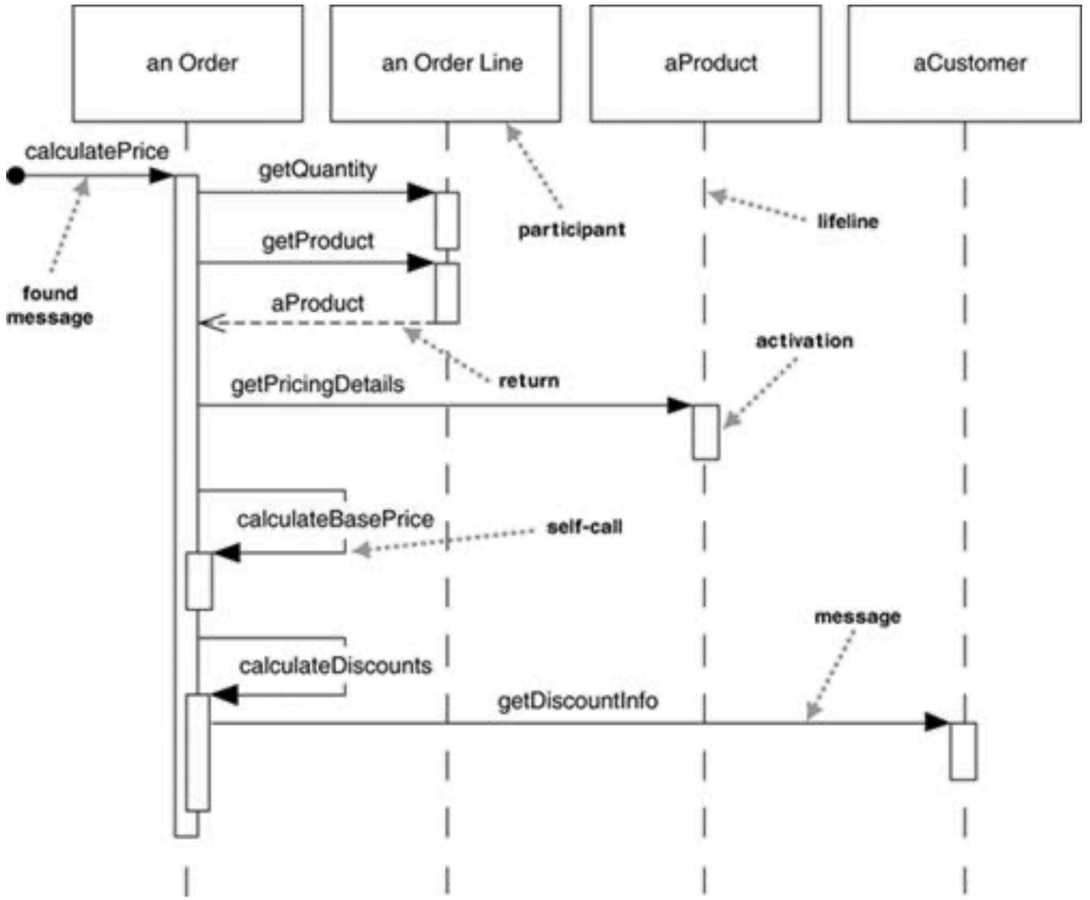


Diagrama de Secuencia

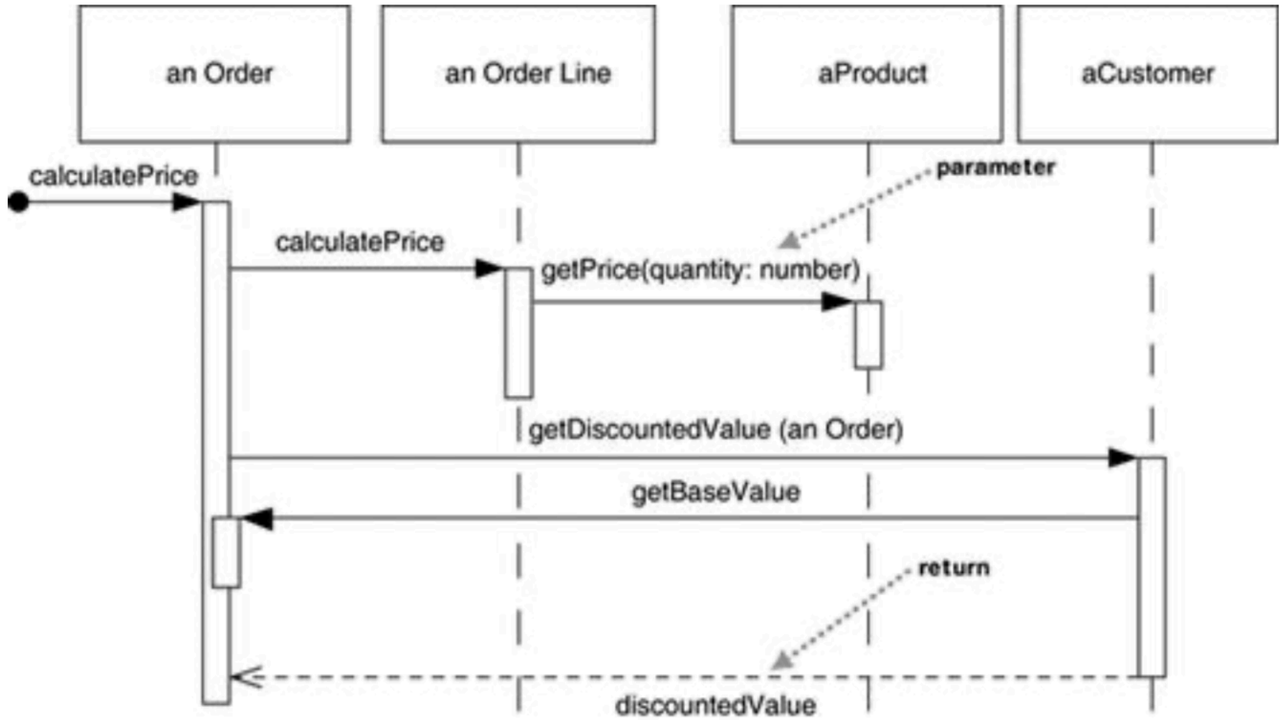
```
class Order
      def calculatePrice()

          a = orderLine.getQuantity()
          b = orderLine.getProduct()
          product.getPricingDetails()
          self calculateBasePrice()
          self calculateDiscount()
          customer.getDiscountInfo()

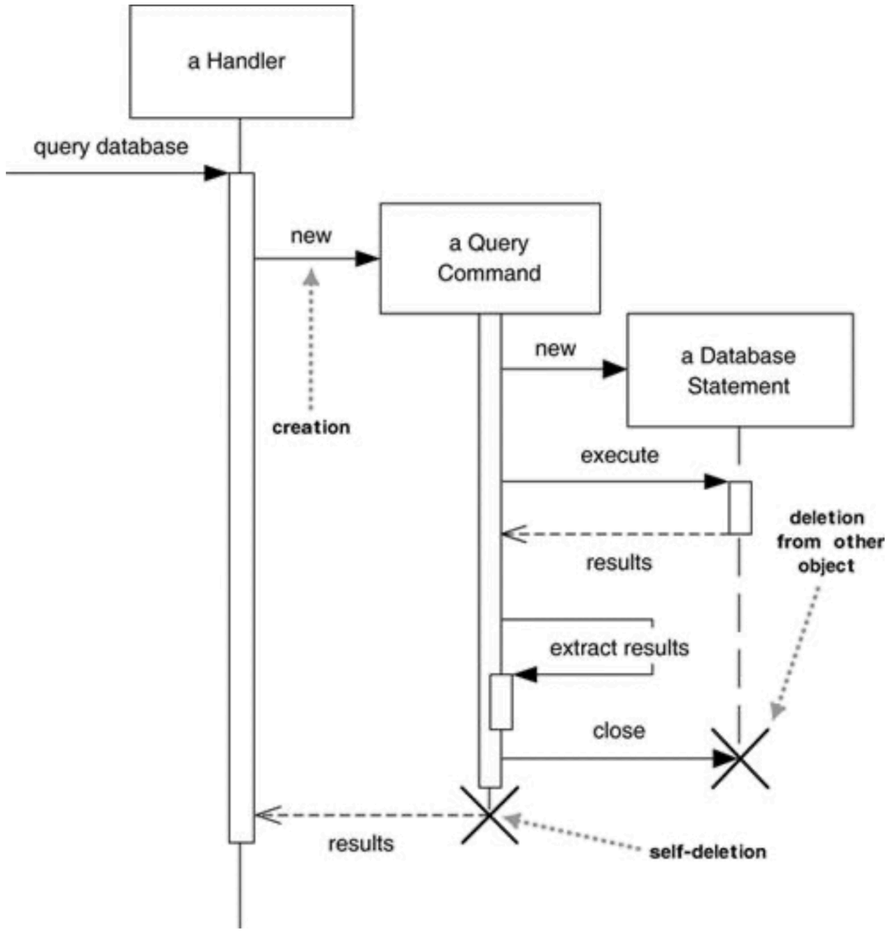
      end
end
```



Argumentos y valores retornado

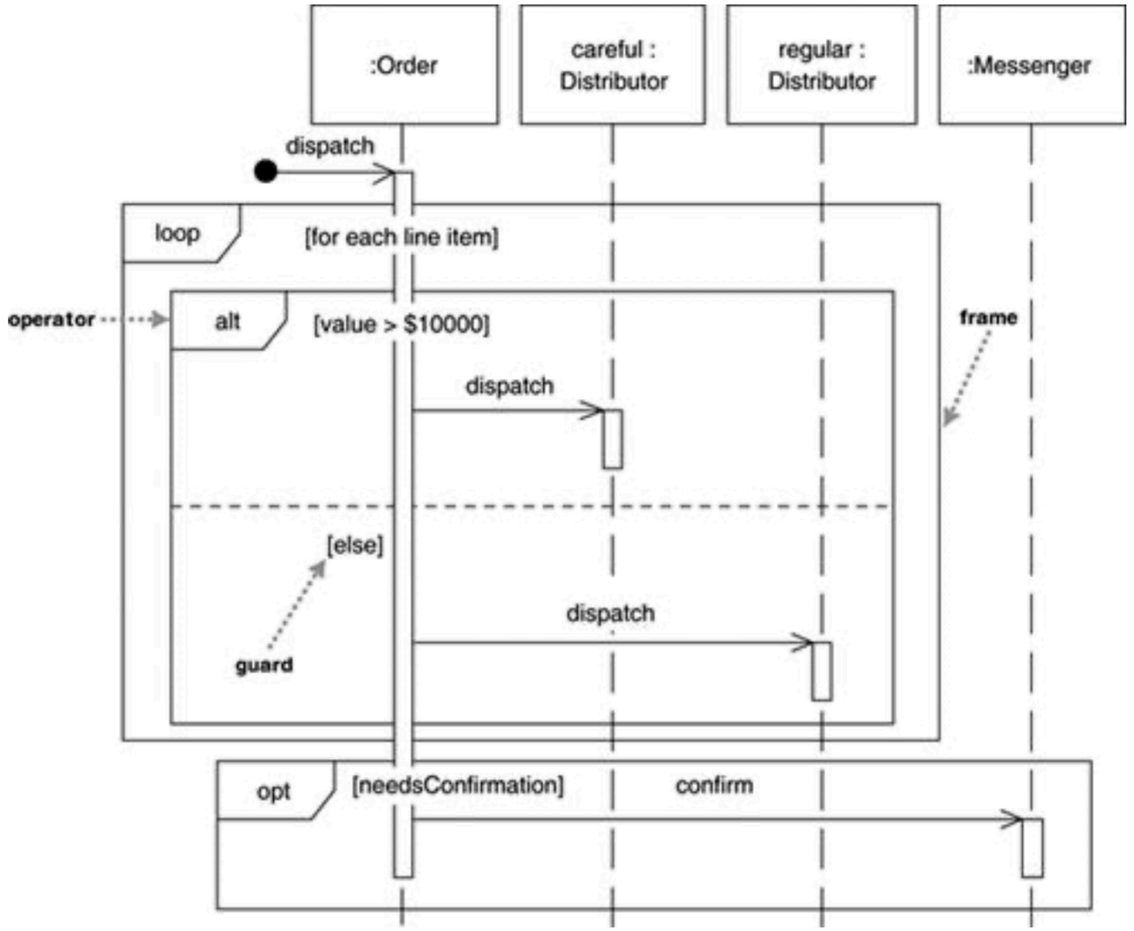


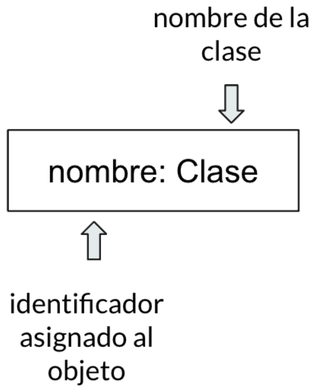
Creación y Eliminación de Objetos



Iteraciones y condicionales

- *Es possible utilizar las dos notaciones “an Order” o “ order1: Order”.*
- *Alt aquí es un if-then-else entre corchetes va la condición*
- *Loop hace referencia a un for loop*
- *El **frame** (marco) denota los mensajes que se llaman dentro del if o del for.*





Ejercicios

- *Dado el siguiente código dibujar un diagrama de clases y un diagrama de secuencia.*
- *Grupos de dos personas: uno hace el diagrama de secuencia y otro el diagrama de clase.*
- *Los que acabe ambos tendrá una décima extra en la I2.*