Ingeniería de Software

11 - Diseño UML

IIC2143-3

Josefa España

jpespana@uc.cl

UML

- Unified Modeling Language (UML).

¿Qué se usa en el presente de este modelo?

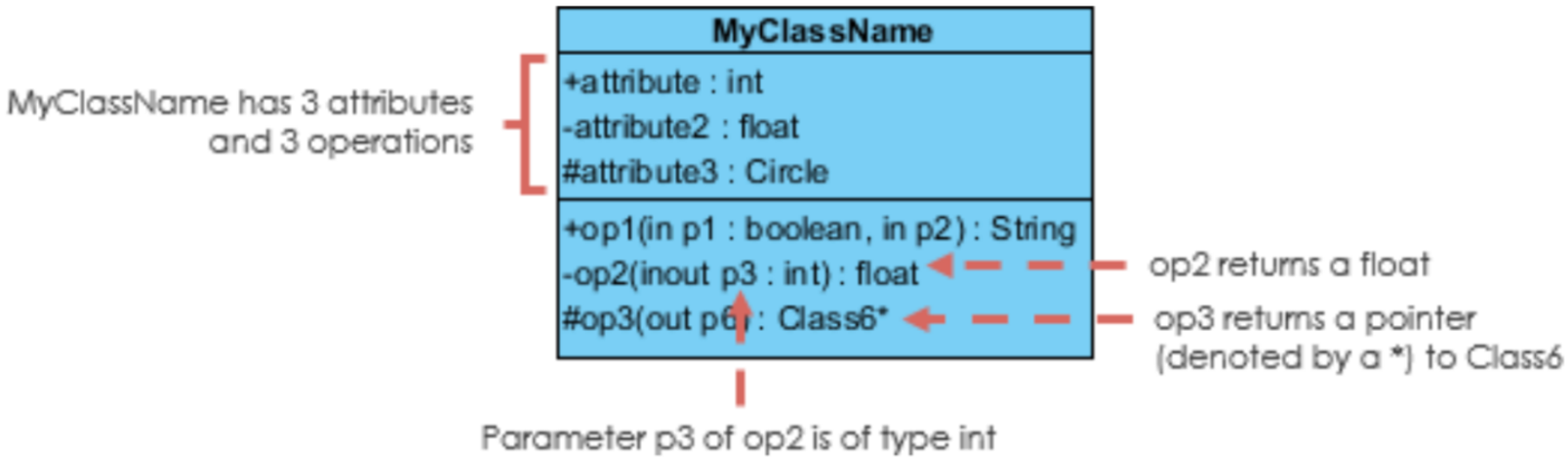
- Diagrama de Clases
- Diagrama de Secuencia

Diagramas de Clase

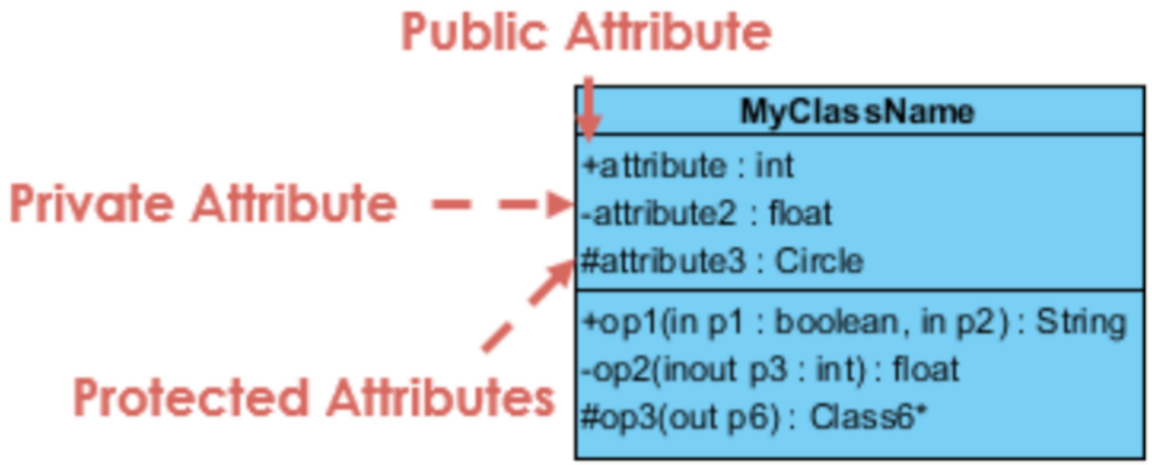
Clases


Name
Shape
Attribute(s)
-length
+getLength()
Operation(s)
+setLength()
Shape
-length : int
+getLength() : int
+setLength(n : int) : void
Class without signature
Class with signature

Clases

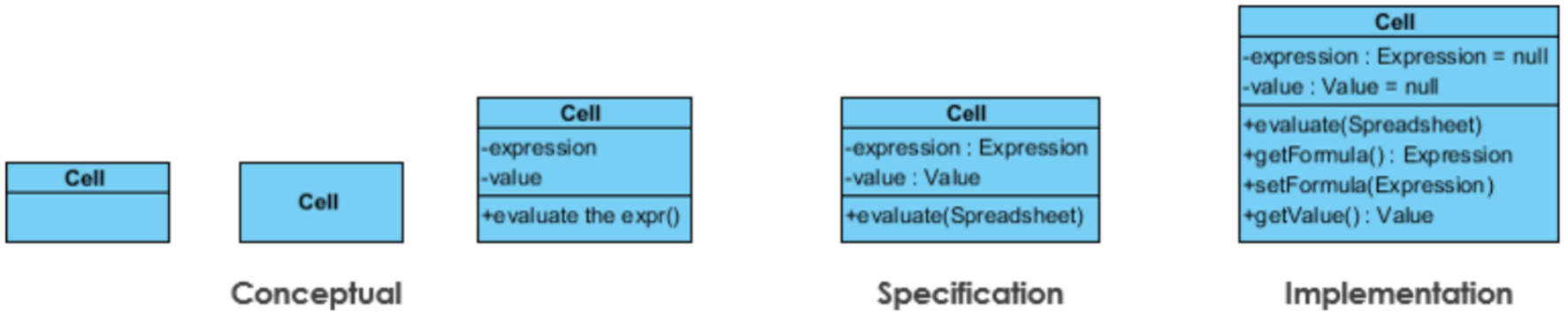

MyClassName
+attribute : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : boolean, in p2) : String
-op2(inout p3 : int) : float
#op3(out p6) : Class6*
MyClassName has 3 attributes and 3 operations
op2 returns a float
op3 returns a pointer (denoted by a *) to Class6
Parameter p3 of op2 is of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

Clases



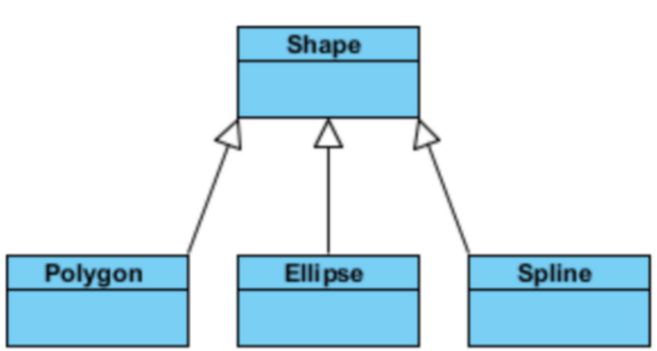
- **Público**: puede ser accedido desde cualquier clase del sistema.
- **Protegido**: solo puede ser accedido desde la misma clase y clases hijas.
- **Privado**: solo puede ser accedido desde la misma clase.

Clases

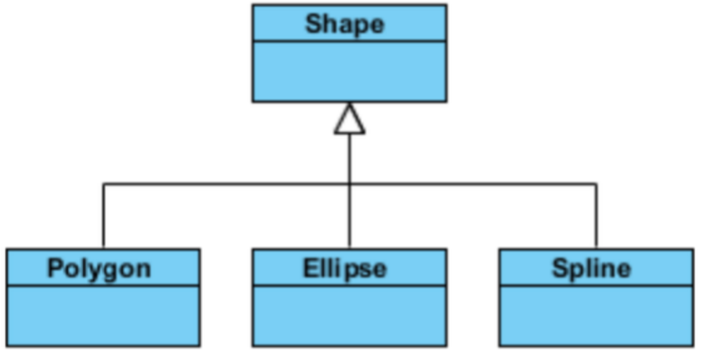


Para cada clase se pueden agregar diferentes niveles de detalle. Para este curso, se pide el mayor detalle posible.

Herencia



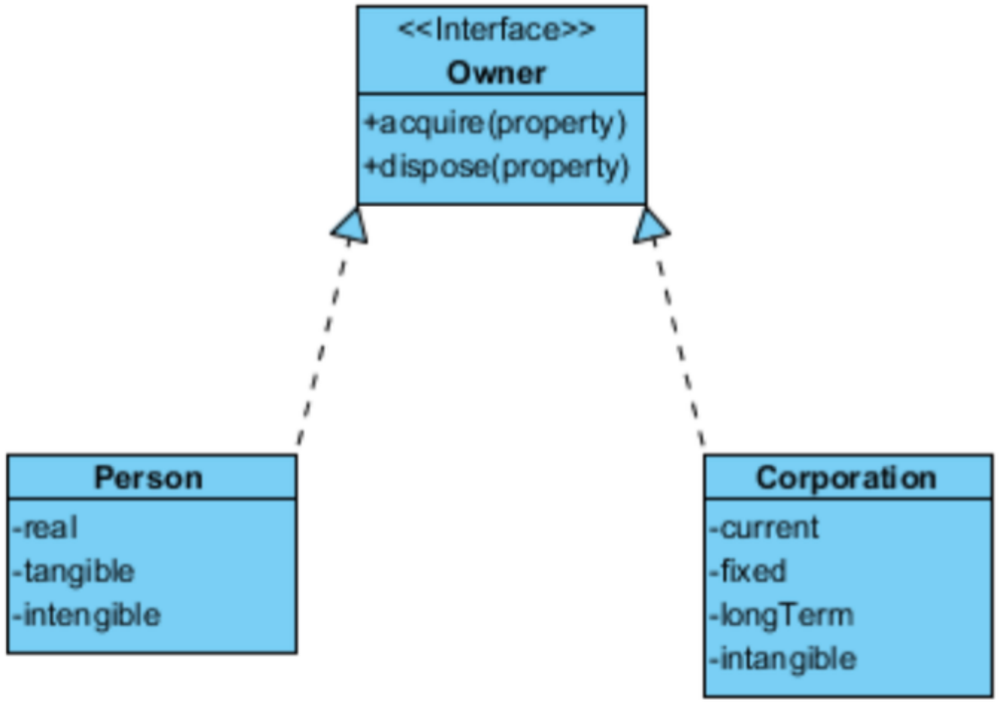
Style 1: Separate target



Style 2: Shared target

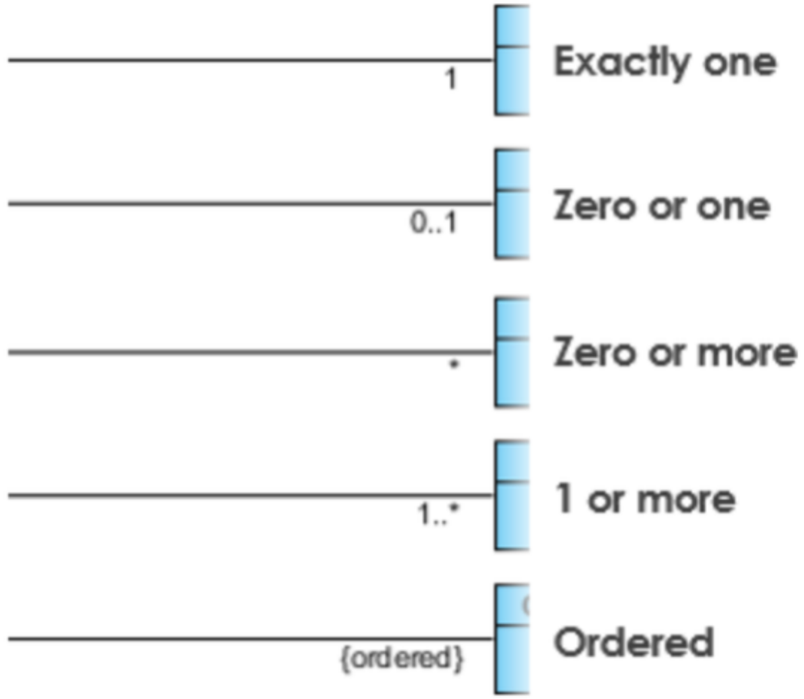
Se puede utilizar cualquiera de los dos estilos de forma indiferente.

Realización

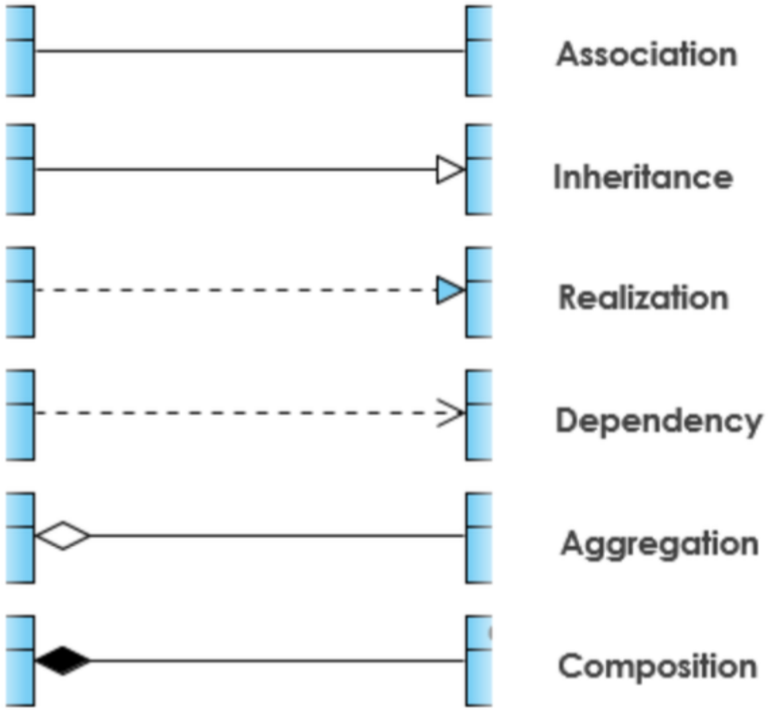


En Ruby, el equivalente a una interfaz es un módulo.

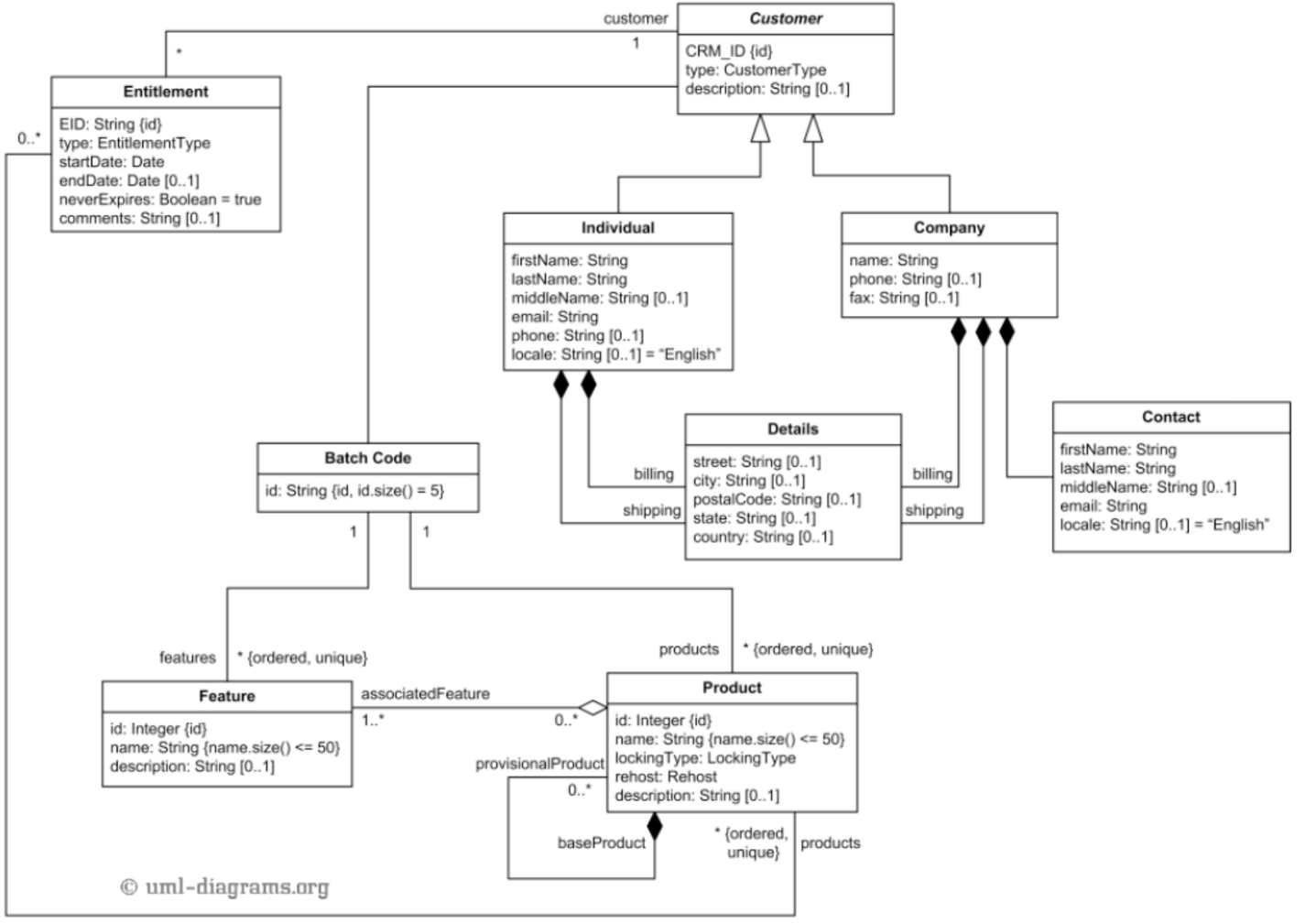
Cardinalidad



Relaciones entre clases



- **Association**: Un objeto de la clase tiene una referencia a un objeto de otra clase por mucho tiempo. (ej: atributos).
- **Inheritance**: Una clase hereda de otra.
- **Realization**: Una clase implementa una interfaz o módulo.
- **Dependency**: un objeto de la clase tiene una referencia a otro objeto de otra clase de forma temporal (ej: variables temporales o argumentos)
- **Aggregation**: Un objeto de la clase A (rombo vacío) está compuesto por objetos de la clase B. Los objetos de B no son creados dentro de A.
- **Composition**: Un objeto de la clase A (rombo lleno) está compuesto por objetos de la clase B. Los objetos de B son creados dentro de B, creando dependencia.


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]
Batch Code
id: String {id, id.size() = 5}
billing
shipping
Details
street: String [0..1]
city: String [0..1]
postalCode: String [0..1]
state: String [0..1]
country: String [0..1]
billing
shipping
Contact
firstName: String
lastName: String
middleName: String [0..1]
email: String
locale: String [0..1] = "English"
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

Diagramas de Secuencia

Diagrama de Secuencia

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

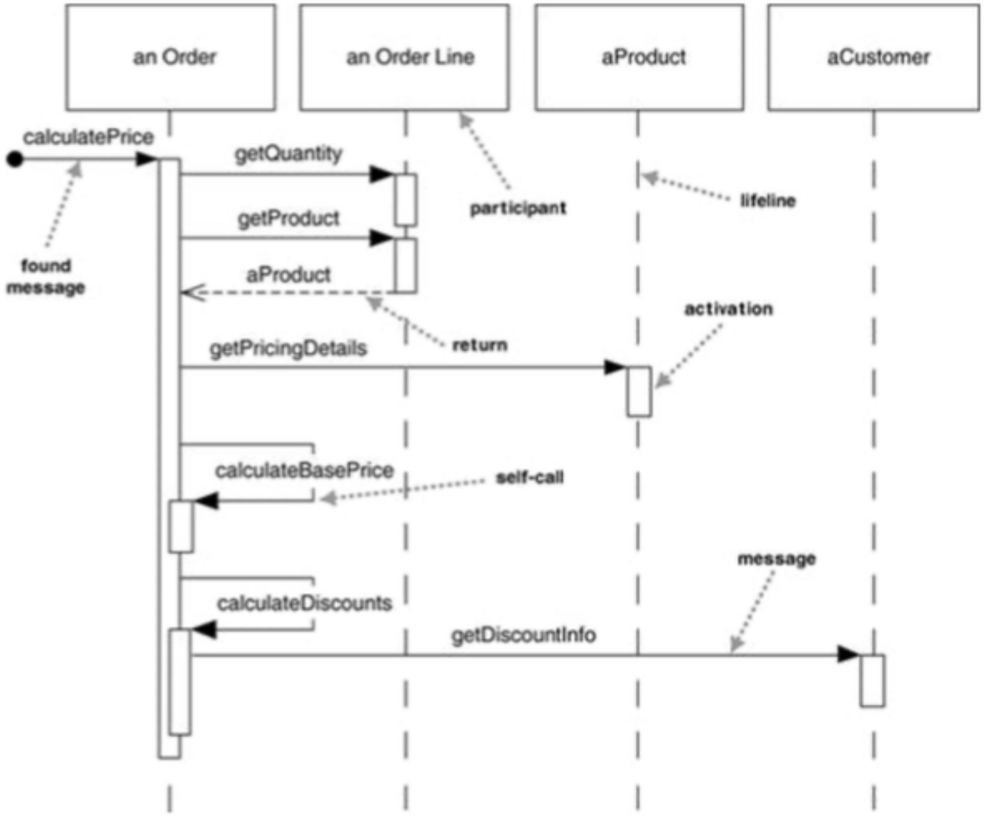
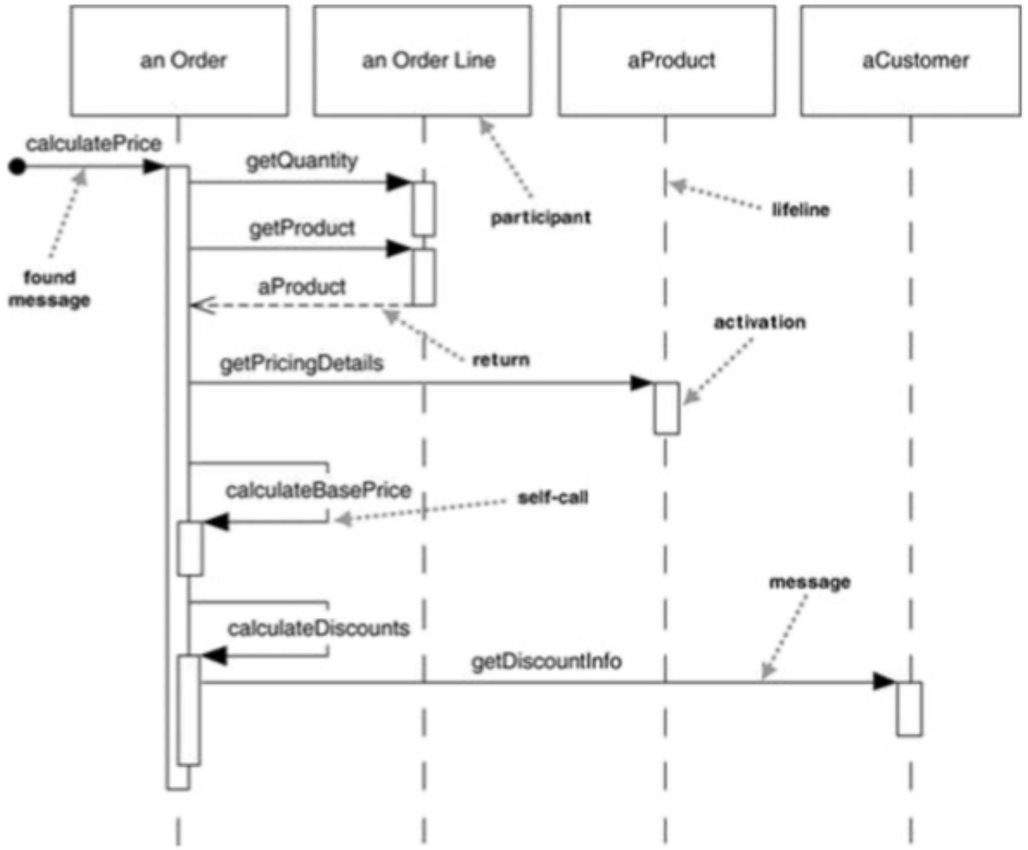
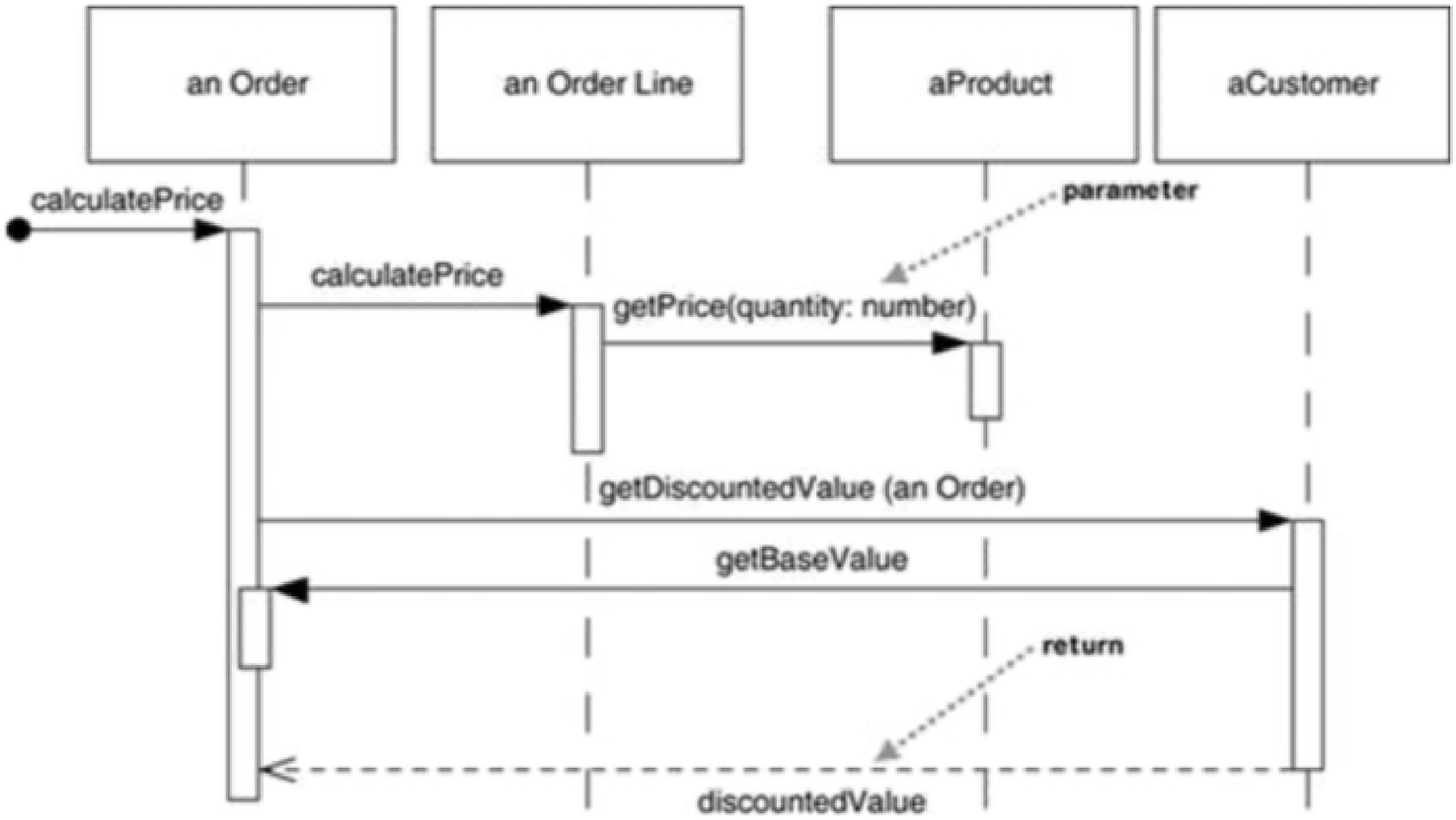


Diagrama de Secuencia

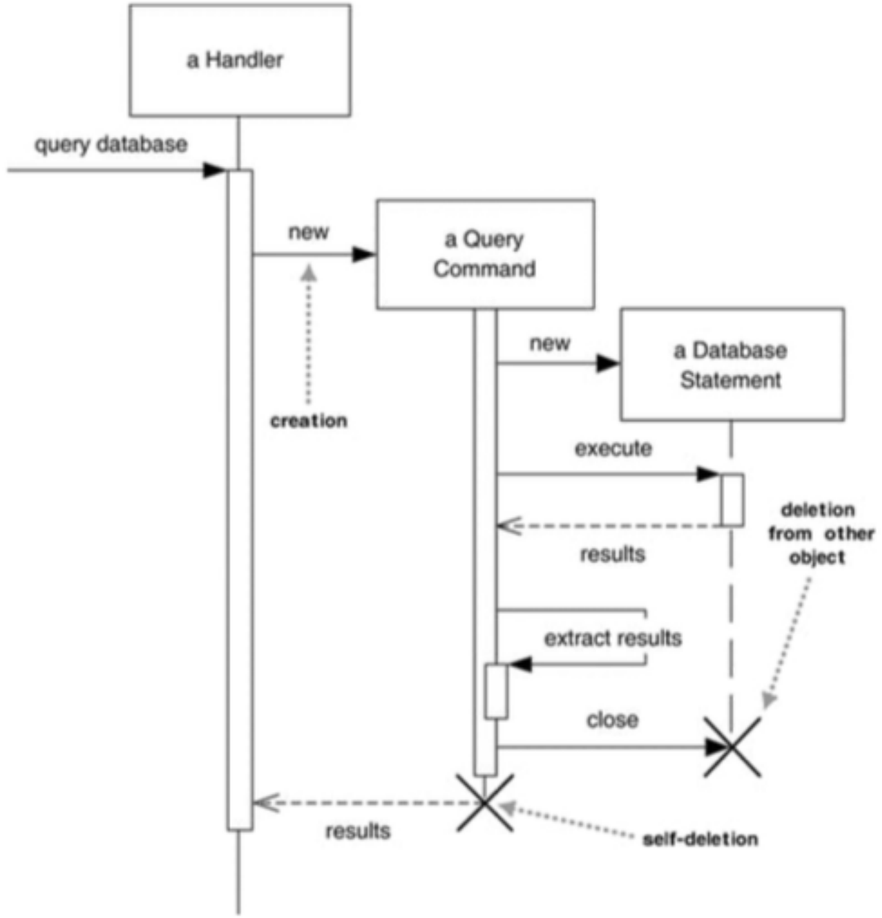
```
class Order
     def calculatePrice()

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

     end
end
```



Argumentos y valores retornados

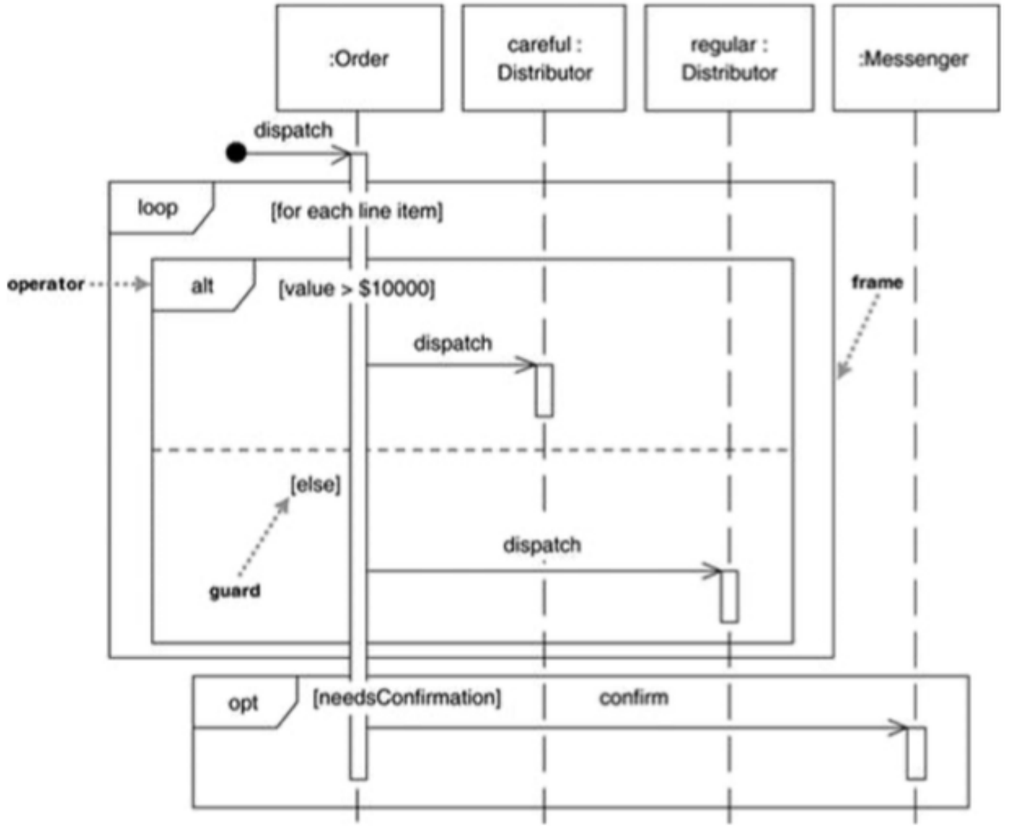


Creación y eliminación de objetos



Iteraciones y condicionales

- Es posible utilizar las dos notaciones “an Order” o “order1: Order”.
- **Alt** es un “if-then-else”, y 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”.



¿Consultas?

Ingeniería de Software

11 - Diseño UML

IIC2143-3

Josefa España

jpespana@uc.cl