**Resumen**

**Representación de los requerimientos funcionales:**

Código R001: El sistema debe gestionar las ventas.

* Armar el pedido.
* Realizar el pago.

Código R002: El sistema debe manejar un stock de libros.

* Dar de alta un libro.
* Modificar cantidad de libros.
* Eliminar libros.

Código R003: El sistema debe llevar un control de stock.

Código R004: El sistema debe permitir el registro de usuarios.

Código R005: El sistema debe loguear usuarios.

Código R006: El sistema debe permitir registrar los datos necesarios para el envió del pedido.

Código R007: El sistema debe listar los libros por rubro.

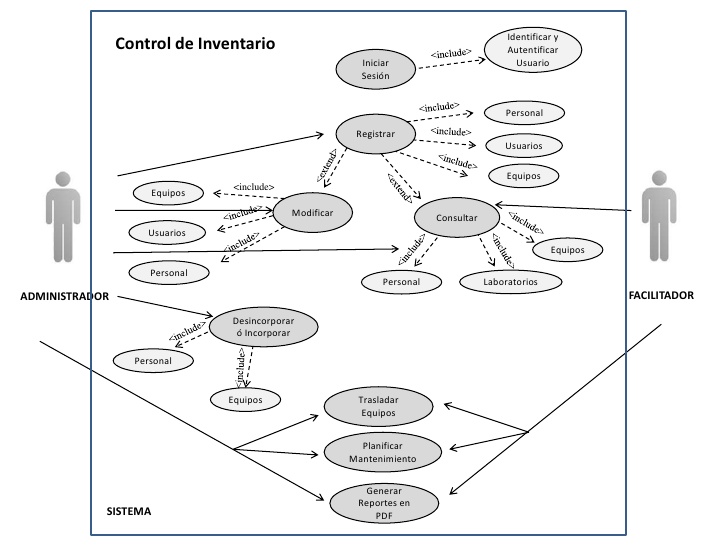
Código R008: El sistema debe llevar un registro histórico de compras por usuario.

**NOTA:** Hay que tener en cuenta que solo se deben enumerar los requerimientos funcionales, los no funcionales no son tenidos en cuenta.

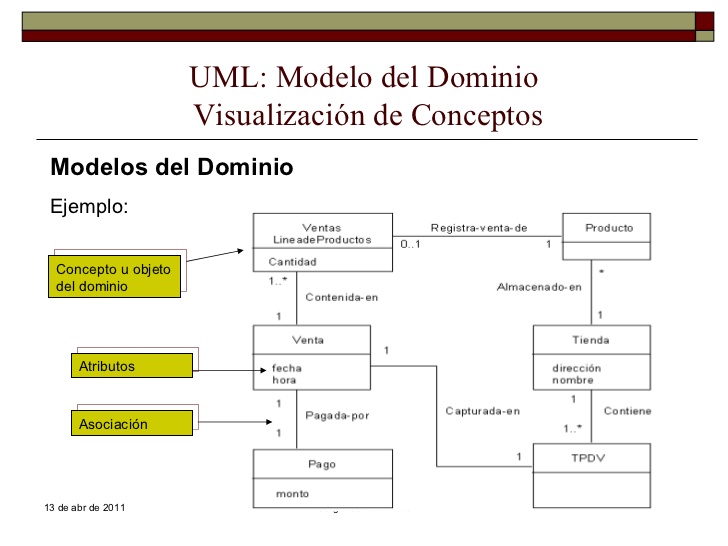
**Especificación de los casos de uso:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | | | | | CU001 | |
| Nombre | | | | | Armar pedido (Realizar la compra) | |
| Actor | | | | | Cliente | |
| Descripción | | | | | Realiza la compra de uno o más libros | |
| Precondición | | | | | El actor debe estar logueado al sistema | |
| Postcondicion | | | | | * El sistema registra la compra. * El sistema actualiza el stock de libros. | |
| Flujo normal | | | | | | |
| 1 | El actor lista los libros a través del “CU002-Listar libros” | | | | | |
| 2 | El actor selecciona los libros que desea comprar y su cantidad. | | | | | |
| 3 | El sistema guarda temporalmente los libros seleccionados por el actor. (Los va mostrando en una pantalla. Esto es un req. No funcional, no interesa) | | | | | |
| 4 | Se repiten los pasos del 1 al 3 hasta que el actor hace clic en el botón “compra” | | | | | |
| 5 | Se registra el pago mediante el caso de “CU003-Realizar pago” | | | | | |
| 6 | El sistema registra la compra y actualiza los totales del stock de libros. | | | | | |
| 7 | Fin del flujo normal | | | | | |
| Flujos alternativos | | | | | | |
| FA001 | | | | | | |
| 1 | | En cualquier momento el actor puede cerrar la sesión, saliendo del sistema. | | | | |
| FA002 | | | | | | |
| 1 | | En cualquier momento el actor puede presionar en “cancelar” | | | | |
| 2 | | Se vuelve al flujo normal, paso 1 | | | | |
| FA003 | | | | | | |
| 2.1 | | | No existen libros para el rubro seleccionado, con lo cual se muestra un mensaje indicando tal situación. | | | |
| 2.2 | | | Se vuelve al flujo normal, paso 1 | | | |
| FA004(Opcion 1) | | | | | | |
| 6.1 | | | El pago no se realizo con éxito, mostrándole al actor un mensaje aclarándolo. | | | |
| 6.2 | | | Se vuelve al flujo normal, paso 4 | | | |
| FA004(Opcion 2) | | | | | | |
| 6.1 | | | | El pago no se realiza con éxito, se muestran las opciones “reintentar” y “cancelar” | |
| 6.2 | | | | El actor preciona en reintentar y vuelve al flujo normal, paso 5 | |
| Si se elige la opción FA004(Opcion 2), el nombre de este flujo alternativo es FA005 | | | | | | |
| 6.2.1 | | | | El actor selecciona “cancelar” | |
| 6.2.2 | | | | El sistema limpia la compra y vuelve al flujo normal, paso 1 | |

**Diagrama de casos de uso**



**Modelo de dominio**

****

**Diagrama de interacción**

* Secuencia:
  + Pro: Muestra claramente la secuencia de mensajes en el tiempo.
  + Contra: Fuerza a extender el diagrama a la derecha al agregar un nuevo objeto.
* Colaboración:
  + Pro: Economiza espacio y es fácil agregar nuevos objetos.
  + Contra: Es difícil ver la secuencia de mensajes.

Estos sirven para mostrar la secuencia de mensajes que se envían los diferentes objetos.

**NOTA:** Un mensaje es una llamada a alguna función de un objeto.

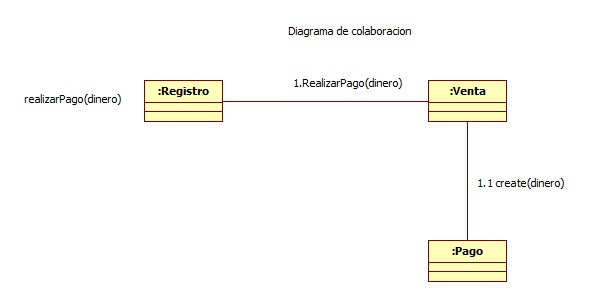
Clase Instancia Instancia con nombre

:Venta

v1:Venta

Venta

**Diagrama de colaboración**



En el diagrama de colaboración se debe poner una flecha sobre el nombre del método y deben estar numerados (Salvo el primero de todos). También se pueden realizar mensajes a si mismos.

El método créate equivale a invocar el constructor de la clase.

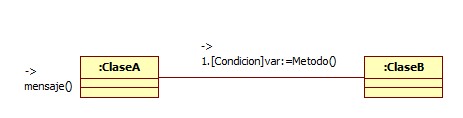
Sintaxis de los mensajes:

retorno:= Nombre\_Metodo (nombre\_parametro[:TipoRetorno\_Param],…, nombre\_parametro\_n[:TipoRetorno\_Param]):Tipo\_Retorno

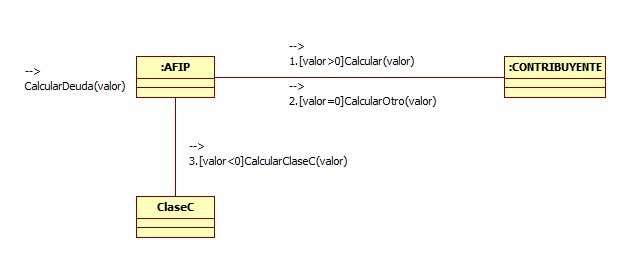
Ej:

Espec:=getEspecificacionProducto(id)

Mensajes Condicionales:

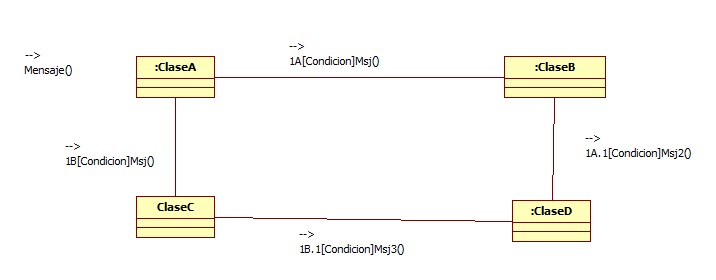


Ej.:

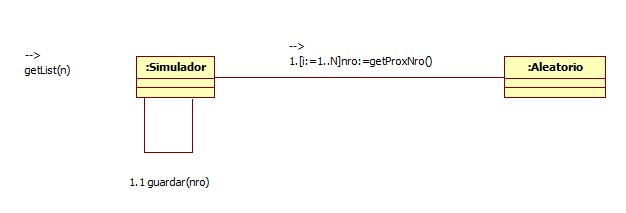


La condición va entre corchetes

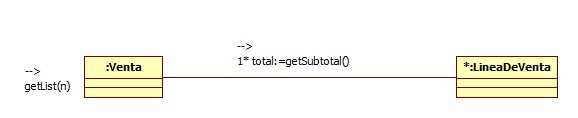
Cuando la condición elige el camino a seguir, por un lado y no por otro se usa:



Iteraciones:

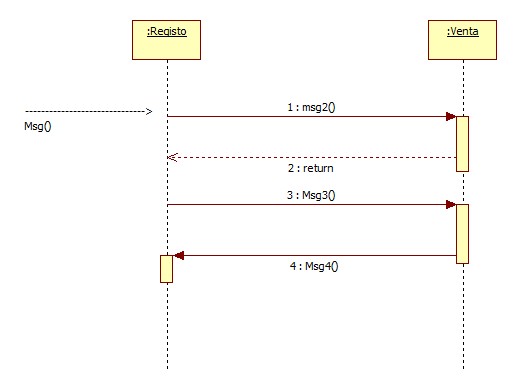


Colecciones:



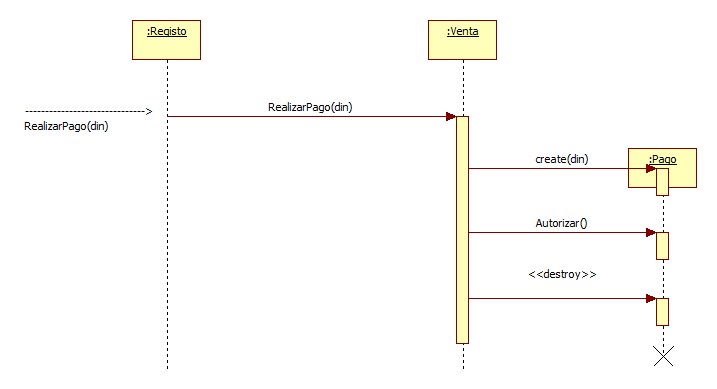
Línea de ventas es un arreglo. Se debe hacer con un cuadrado doble y colocar un asterisco para que se lo distinga como un arreglo.

**Diagrama de secuencia:**



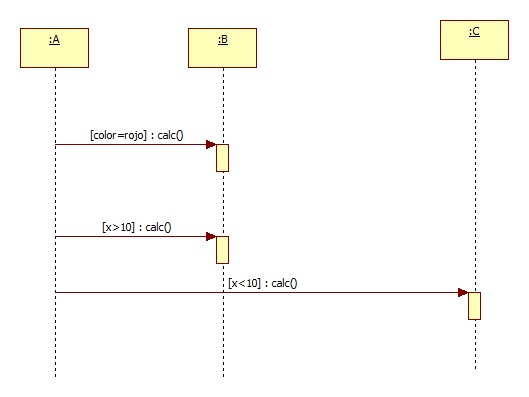
Generalmente no se utiliza el return en los diagramas

Equivalente el primer diagrama de colaboración:

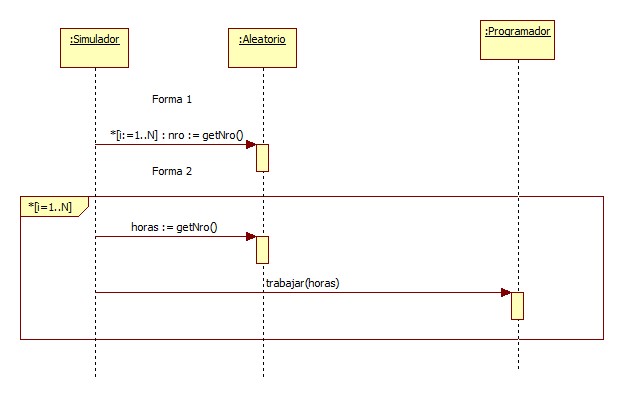


Generalmente el destroy no se utiliza.

Condicionales:



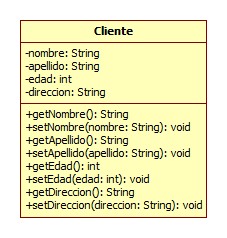
Iteraciones:



**Diagrama de clases**

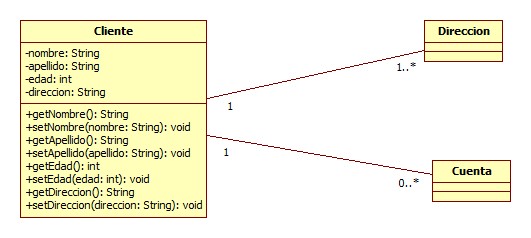
**Representación de una clase:**

1. Primera parte: Es el nombre de la clase. Siempre es en singular y empieza en mayúscula.
2. Segunda parte: Es donde van los atributos.
3. Tercera parte: Es donde van los métodos.

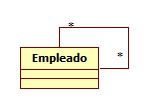


**Relación entre clases:**

En las relaciones entre clases se debe poner la cardinalidad.



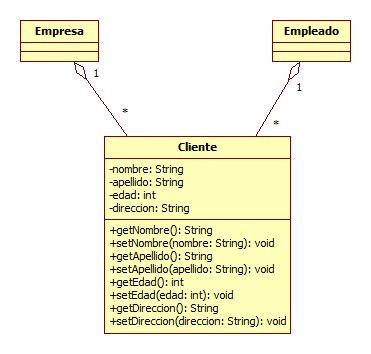
También una clase se puede relacionar con ella misma.



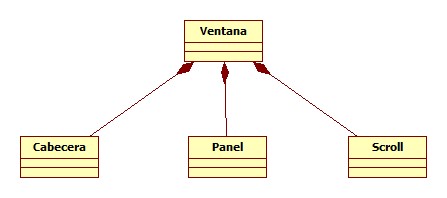
**Tipo de relaciones entre clases:**

Cuando una de las clases forma parte de otra, es decir, una clase está compuesta de variad otras.

1. Agregación: Es una composición débil:



1. Composición: Es una composición fuerte:



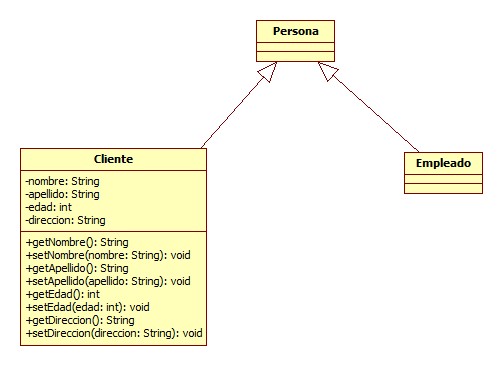
**Dependencia entre clases:**

Es cuando una clase necesita los métodos o la información de otra clase para resolver sus propios métodos.



**Generalización y especificación de clases:**

Es la herencia de una clase.



Los métodos del tipo protected solo los pueden usar la clase padre y las clases que heredan desde el padre. Solo en herencia directa.

**Patrones de diseño**

**Responsabilidades y Métodos:**

* Metodos: Son las funciones de una clase.
* Responsabilidades de un objeto: Son los limites de esa clase, por ej. Persistir un dato en una BD. Es el contrato de lo que se espera que realice ese objeto. La responsabilidad es distinto a un método, pero los métodos se usan para poder llevar a cabo una responsabilidad. Se define en términos de:
  + Conocer: Conose sus datos.
  + Hacer: Crear otras calses.

**Patrones:**

Proporciona una solución a un problema conocido y repetitivo.

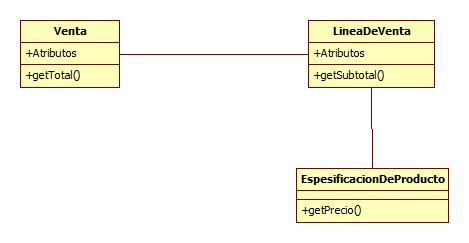
**Patrones GRASP (General Responsibility Assignment Software Patterns):**

* **Experto en Informacion:** ¿Cuál es la manera general de asignar responsabilidades? Asigno la responsabilidad al objeto que tiene toda la información necesaria.

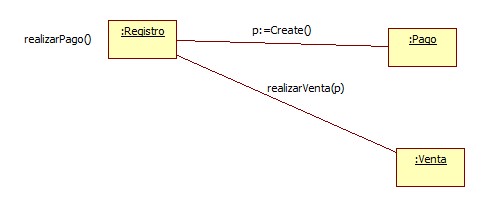


Ej.:

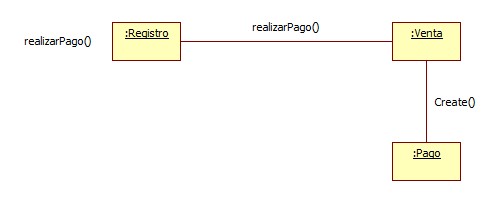
La responsabilidad de conocer el total de una venta lo tiene la clase venta. La clase podría tener un método llamado getTotal() para poder realizar esto. La clase LineaDeVenta debe conocer el subtotal de una línea de venta, podría tener un método llamado getSubtotal(). A su vez esta clase debe conocer a la clase EspecificacionDeProducto ya que esta conoce el precio del producto y tiene un método getPrecio().



* **Creator:** ¿Quién debería ser responsable de crear nuevos objetos? La clase B tiene la responsabilidad de crear un objeto de la clase a si:
  + B contiene a A.
  + B tiene los datos de inicialización de A.
  + B utiliza a A.
  + B registra/almacena a A.
  + B es un creador de A.
* **Bajo acoplamiento:** ¿Cómo puedo solucionar/minimizar las dependencias? Asignar las responsabilidades de modo tal que el acoplamiento parezca bajo.
  + Ej. De alto acoplamiento



* + Ej. De bajo acoplamiento



* **Alta cohesion:** ¿Cómo mantener un objeto manejable? Asignar responsabilidades de modo tal que la cohesion permanezca alta.
* **Controller:** ¿Quién es el responsable de gestionar un evento de entrada al sistema? Asignar la responsabilidad de manejar eventos de entrada al sistema a una clase que:
  + Representa al sistema global
  + Representa a un escenario de caso de uso (los llamados <CasosdeUso>Controller)
    - Usa el mismo controlador para todos los eventos del caso de uso.