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.

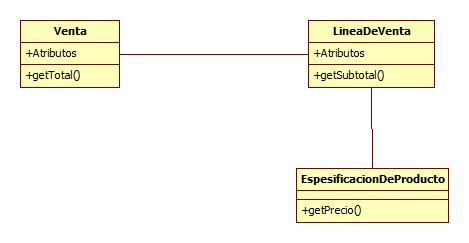
Patrones GRASP (General Responsibility Assignment Software Patterns):

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

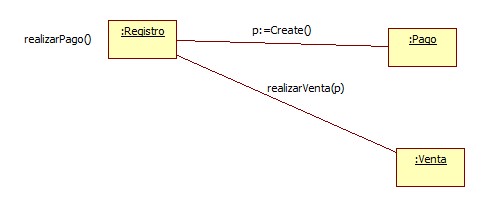


Ej.:

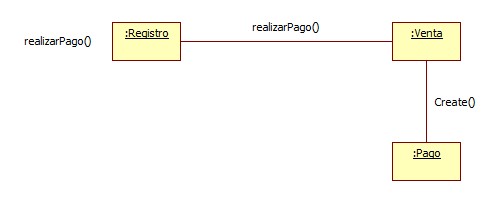
La responsabilidad de conocer el total de una venta lo tiene la clase venta. La clase podría tener un método llamado getTotal() para poder realizar esto. La clase LineaDeVenta debe conocer el subtotal de una línea de venta, podría tener un método llamado getSubtotal(). A su vez esta clase debe conocer 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.