PATRONES GRASP

Ingeniería de Software 1

INTRODUCCIÓN

Funcionalidad de Sistemas Orientados a Objetos está definida por los **mensajes entre objetos** para realizar operaciones

Una adecuada asignación de responsabilidades entre objetos contribuye al desarrollo de sistemas robustos

¿QUÉ SON LOS PATRONES?

Son **pares etiquetados problema/solución,** que pueden ser aplicados en nuevos contextos

Son **consejos de anteriores diseñadores** para ayudar a los actuales en nuevas situaciones

PATRONES GRASP

General Responsibility Assignment Software Patterns

Patrones que ayudan a distribuir mensajes entre los objetos de un sistema

Algunos son:

- Experto
- Creador
- Bajo Acoplamiento
- Alta Cohesión
- Controlador

EXPERTO (EN INFORMACIÓN)

Problema:

¿Cuál es el principio general para asignar responsabilidades a los objetos?

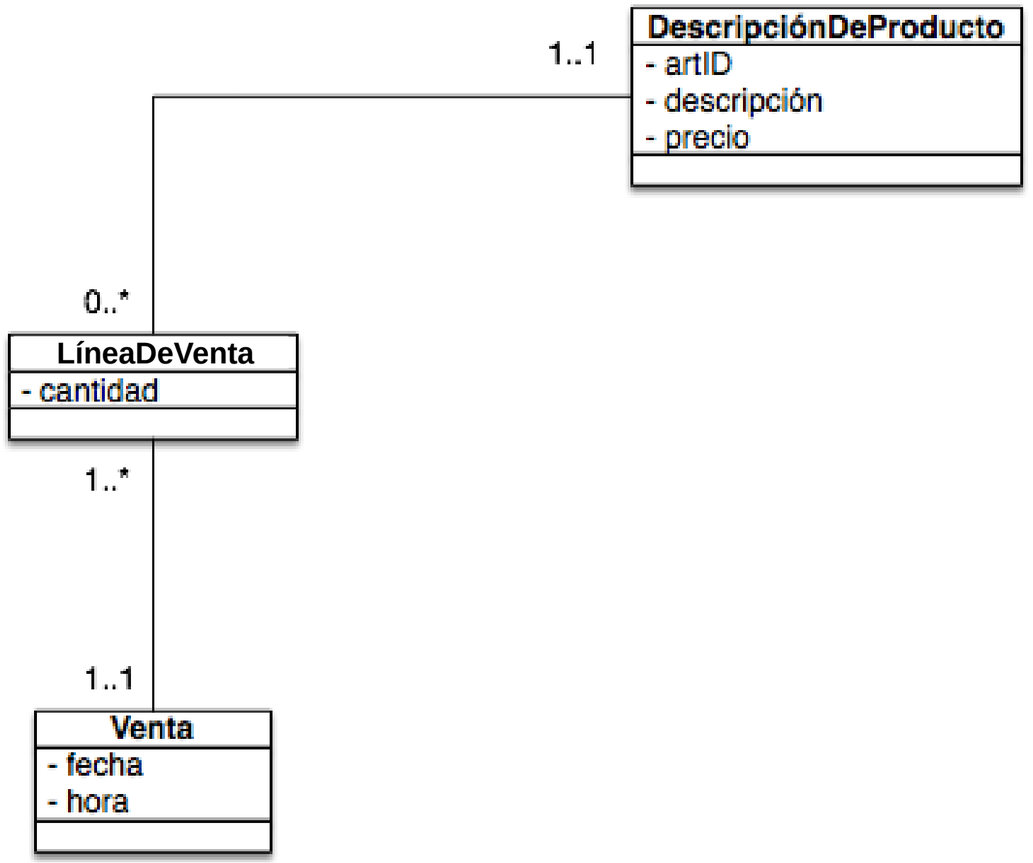
¿Quién debiera ser el responsable de conocer la información?

Solución:

Asignar una responsabilidad a la clase que tiene la información necesaria para cumplir la responsabilidad (experto)

EXPERTO: EJEMPLO

¿Quién es el responsable de conocer el monto total de la Venta?



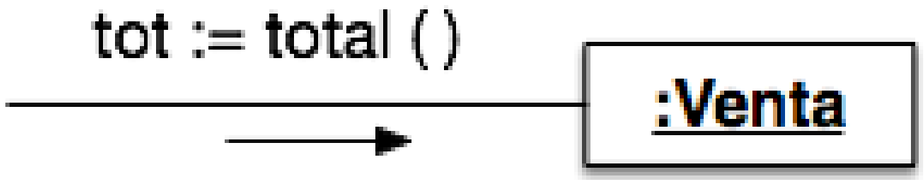
EXPERTO: EJEMPLO

P: ¿Qué información necesito?

R: Todas las Líneas de Venta y sus respectivos subtotales

P: ¿De qué clase del Esquema Conceptual puedo obtener esa información?

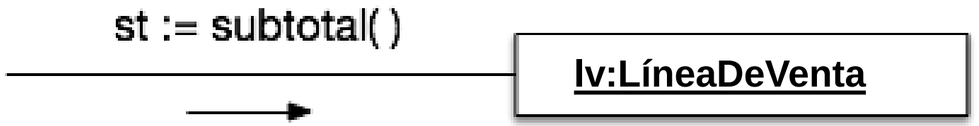
R:



EXPERTO: EJEMPLO

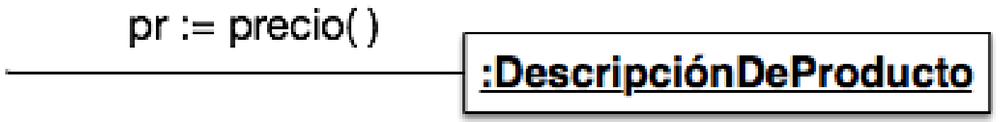
P: ¿Y los respectivos subtotales (cantidad x precio) de cada Línea de Venta?

R:

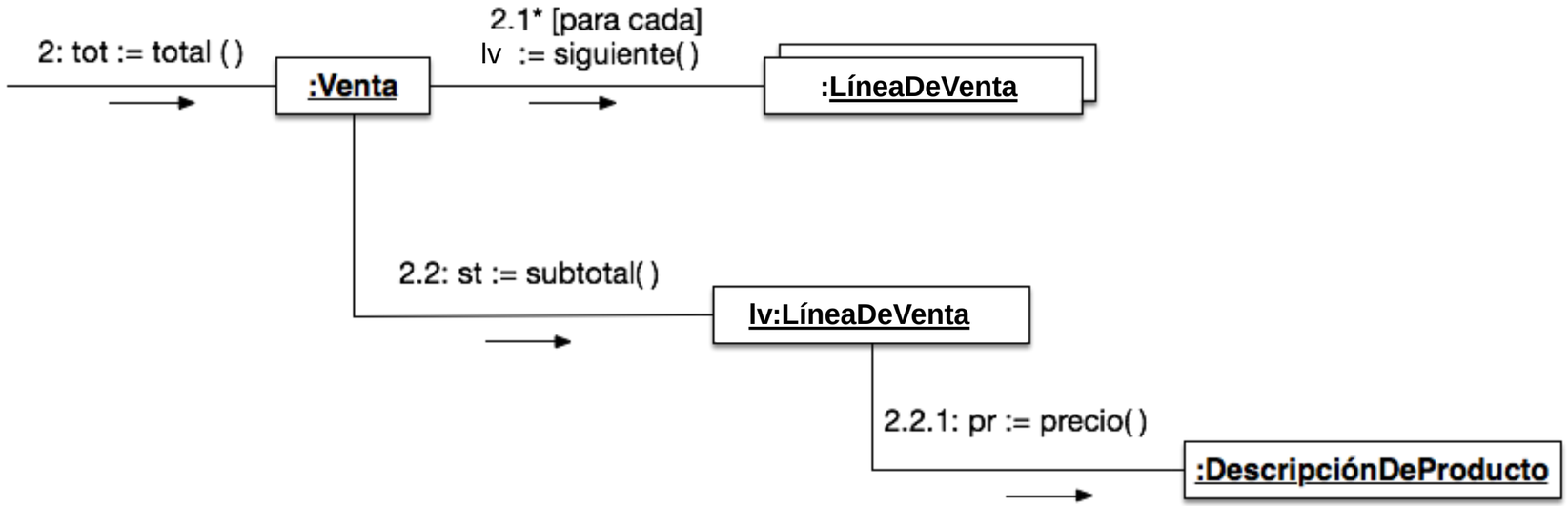


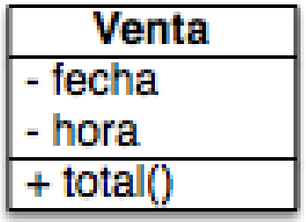
P: ¿Y el Precio de cada producto?

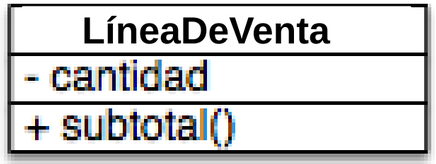
R:

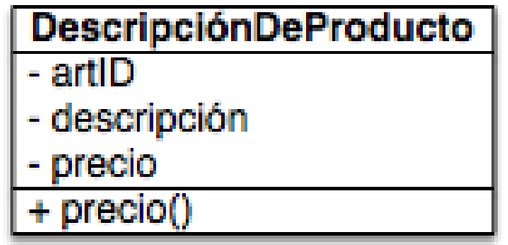


RESULTADO


2: tot := total ( )
:Venta
2.1* [para cada]
lv := siguiente( )
:LíneaDeVenta
2.2: st := subtotal( )
lv:LíneaDeVenta
2.2.1: pr := precio( )
:DescripciónDeProducto


Venta
- fecha
- hora
+ total()


LíneaDeVenta
- cantidad
+ subtotal()


DescripciónDeProducto
- artID
- descripción
- precio
+ precio()

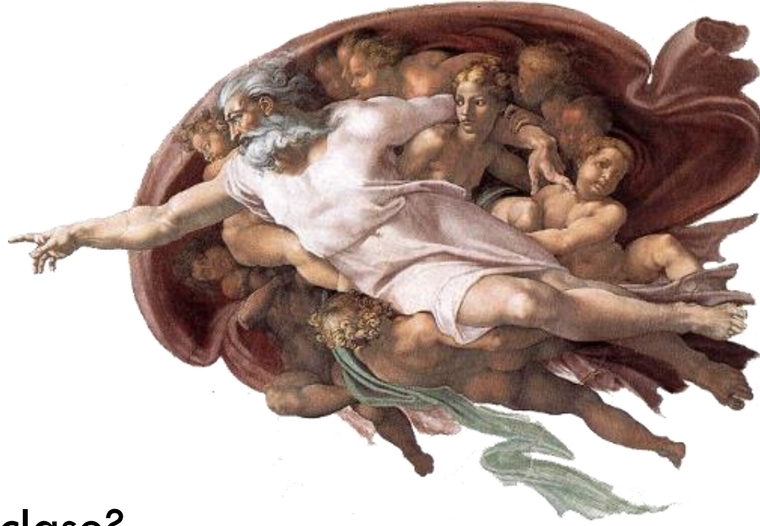
EXPERTO (EN INFORMACIÓN)

Mantiene el encapsulado de la información

Favorece el bajo acoplamiento

Puede originar clases excesivamente complejas

CREADOR



Problema:

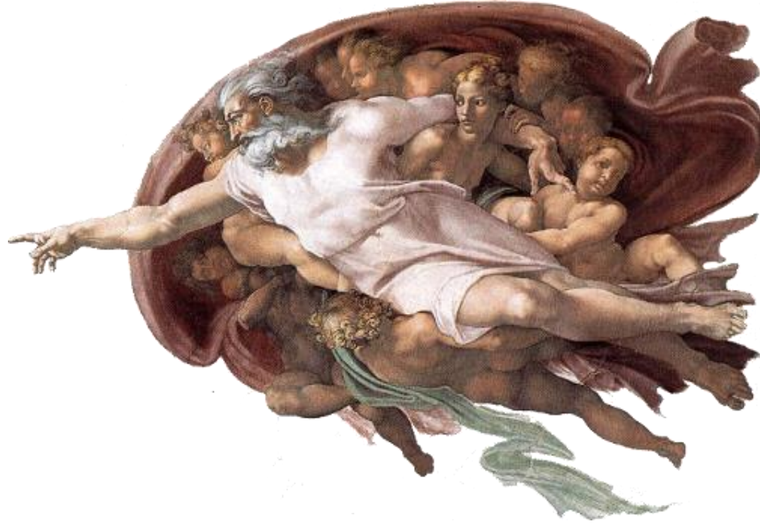
¿Quién debe ser el responsable de **crear** una nueva instancia de alguna clase?

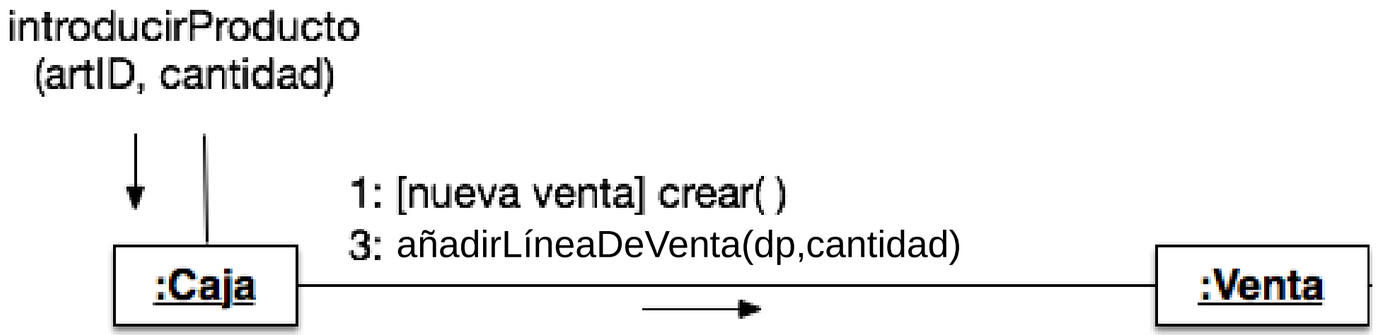
Solución:

B es un **Creador** de A si se cumple uno o más de las siguientes condiciones:

- B agrega objetos de A
- B contiene objetos de A
- B registra objetos de A
- B utiliza intensivamente objetos de A
- B tiene los datos de inicialización de objetos de A (B es Experto en la creación de A)

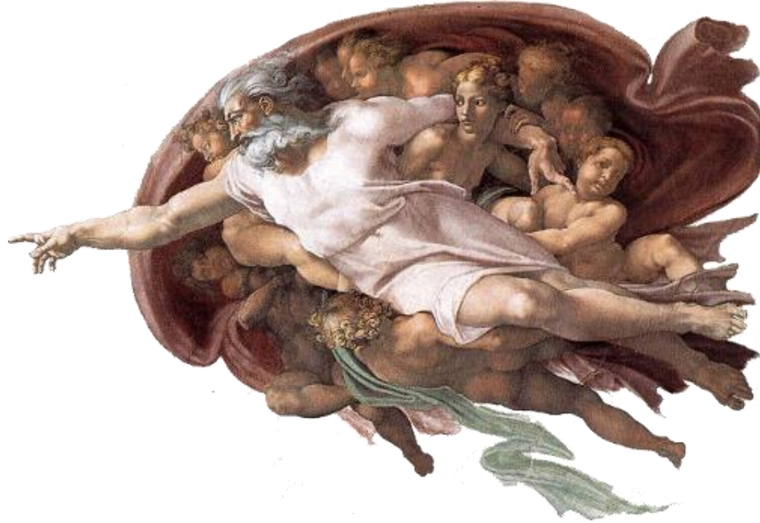
CREADOR: EJEMPLO

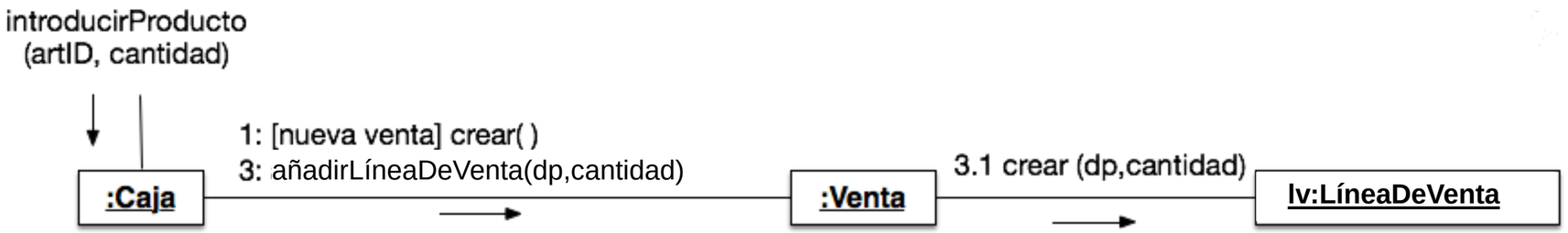




- Caja registra objetos Venta
- Caja utiliza intensivamente objetos Venta

CREADOR: EJEMPLO

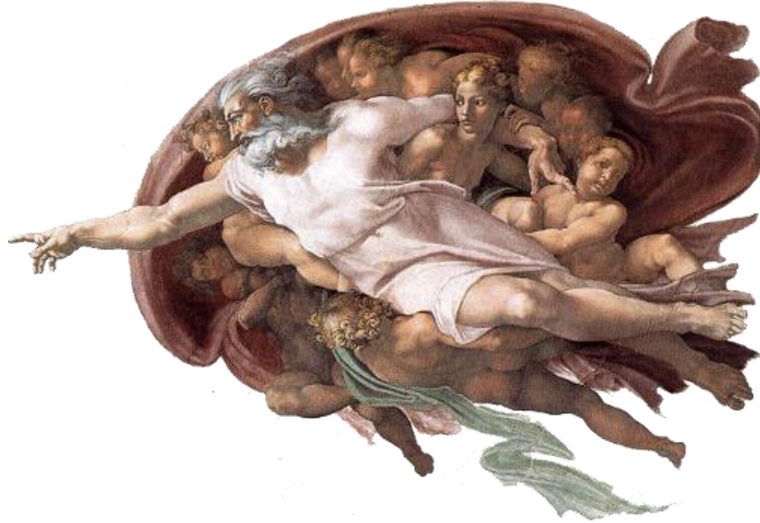




- Caja registra objetos Venta
- Caja utiliza intensivamente objetos Venta

- Venta contiene objetos LíneaDeVenta
- Venta utiliza intensivamente objetos LíneaDeVenta
- Venta tiene los datos de inicialización de objetos LíneaDeVenta

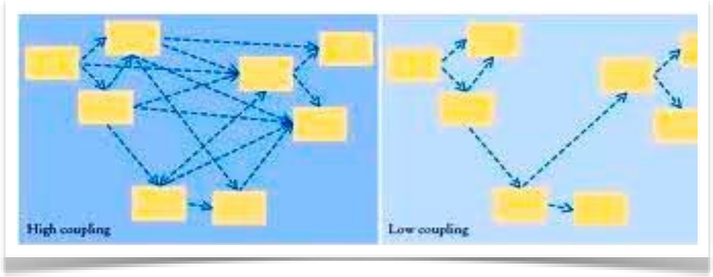
CREADOR



Encontrar un Creador que necesite conectarse al objeto creado en alguna situación

Promover el bajo acoplamiento, al hacer responsable a una clase de la creación de objetos que necesita referenciar

BAJO ACOPLAMIENTO



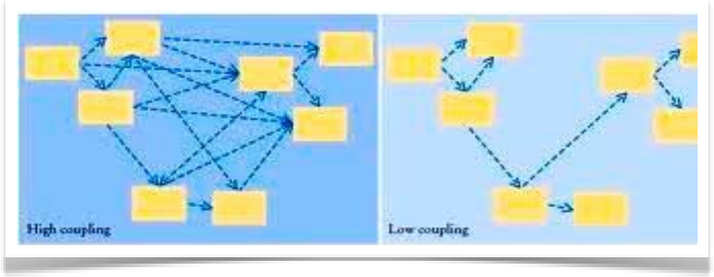
Problema:

¿Cómo promover una baja dependencia y una alta reutilización de clases?

Solución:

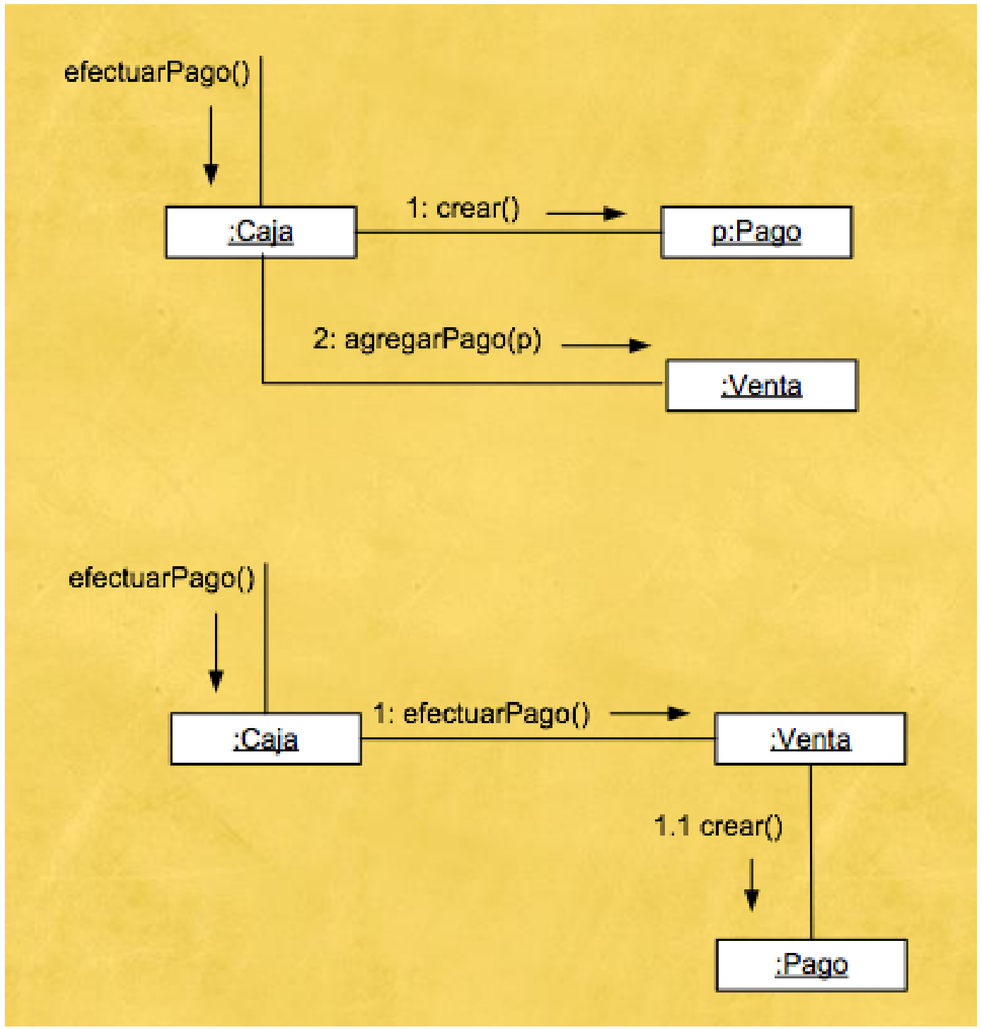
Asignar responsabilidades manteniendo bajo el acoplamiento

BAJO ACOPLAMIENTO:EJEMPLO

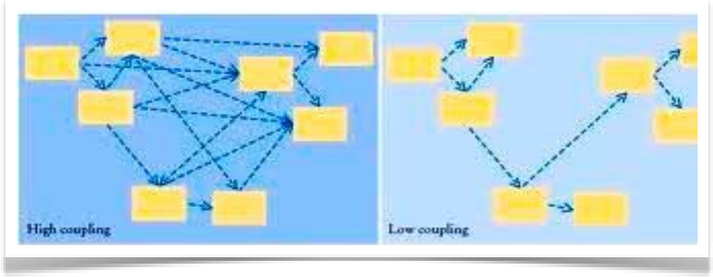


Deseamos crear un Pago y asociarlo a Venta

¿Qué alternativa de diseño soporta el patrón Bajo Acoplamiento?



BAJO ACOPLAMIENTO

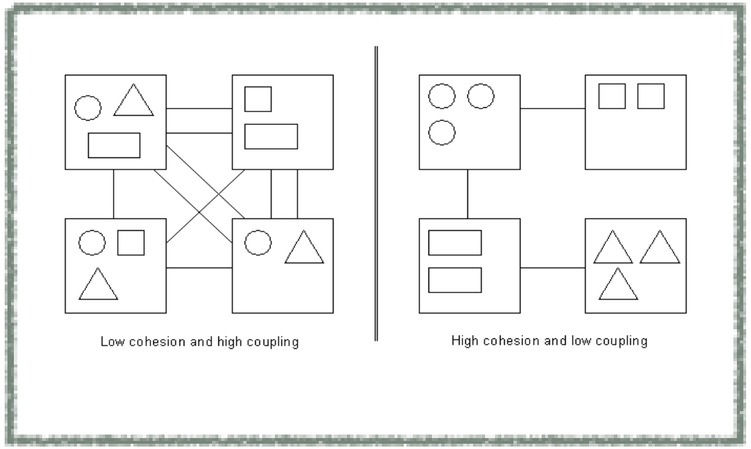


Formas de acoplamiento

- Clase A posee un atributo que refiere a una Clase B
- Clase A posee un método que referencia a Clase B, mediante parámetro o retorno
- Clase A es subclase de B

Soporta el diseño de clases más independientes, que reducen el impacto de cambios y acrecientan la reutilización

ALTA COHESIÓN



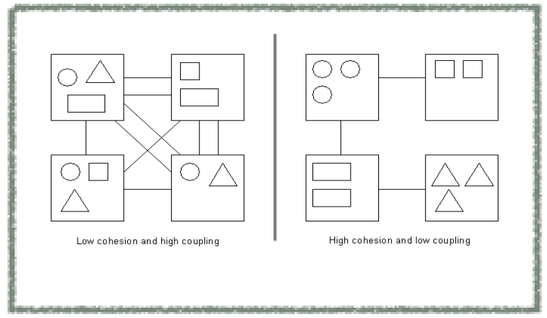
Problema:

¿Cómo promover una complejidad manejable de clases?

Solución:

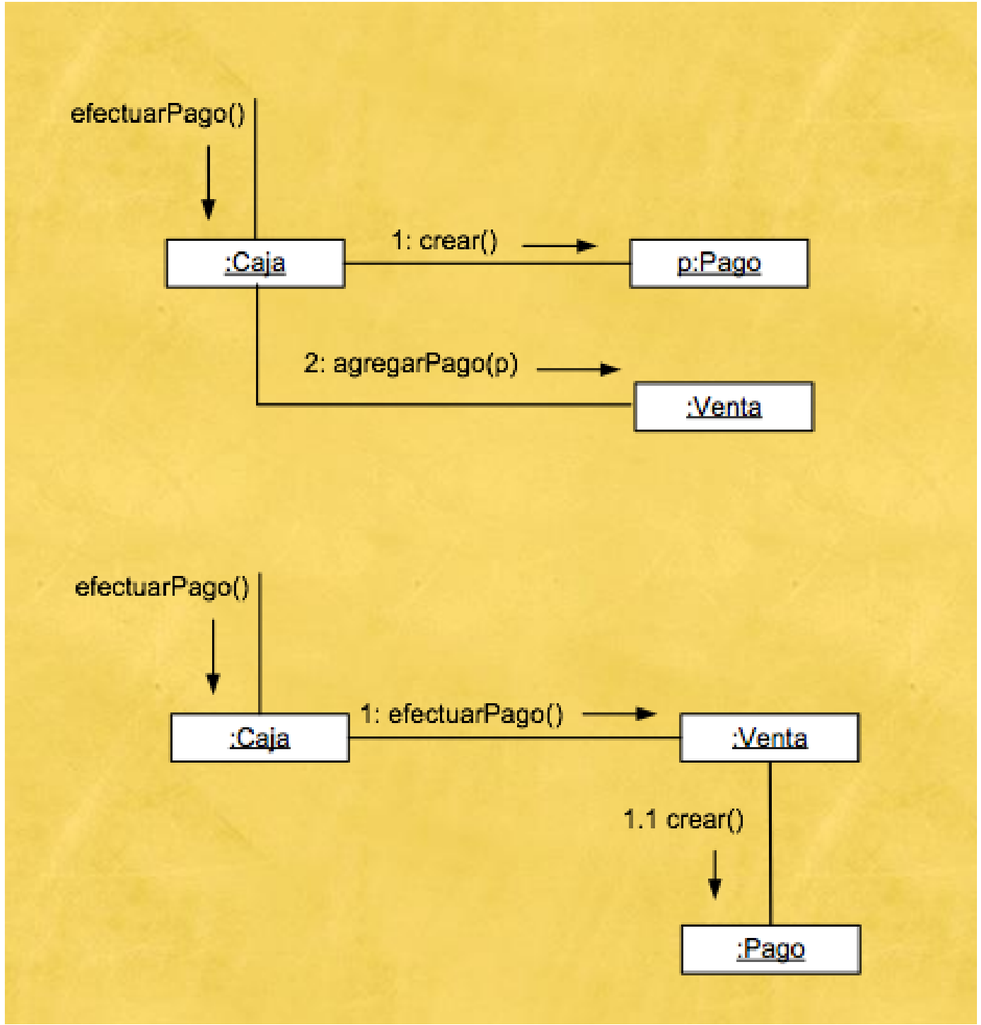
Asignar responsabilidades manteniendo alta la cohesión

ALTA COHESIÓN: EJEMPLO

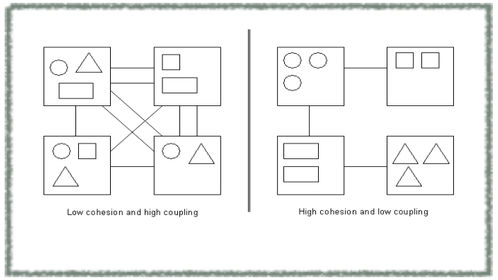


Deseamos crear un Pago y asociarlo a Venta

¿Qué alternativa de diseño soporta el patrón Alta Cohesión?



ALTA COHESIÓN



Muy Baja Cohesión: una Clase, única responsable de muchas cosas en áreas funcionales muy diversas

Baja Cohesión: una Clase, única responsable de una tarea compleja en un área funcional

Alta Cohesión: una Clase con responsabilidades moderadas en un área funcional, colaborando con otras para concretar tareas. Diseño más claro y comprensible.

CONTROLADOR

Problema:

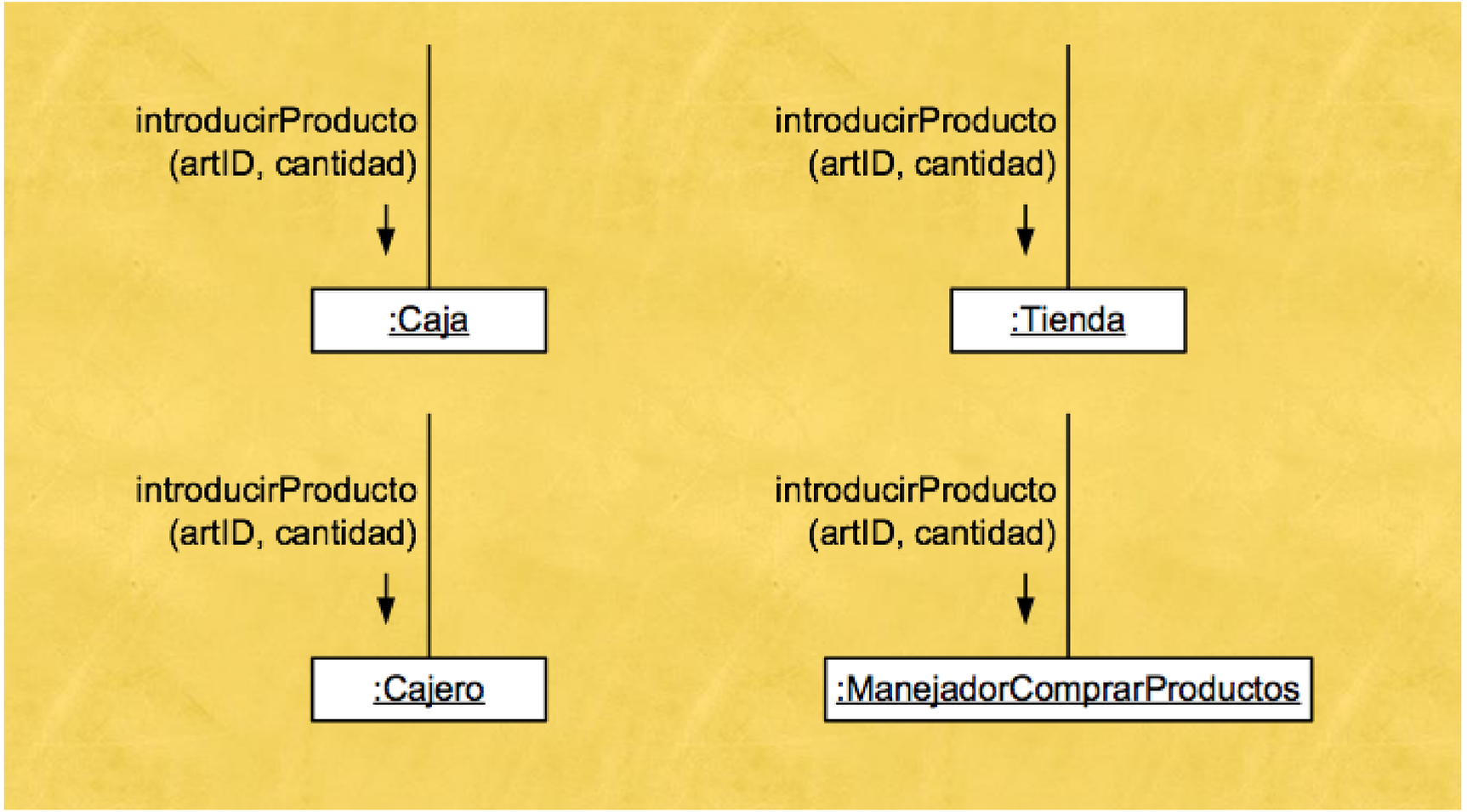
¿Quién debe ser el responsable de gestionar un evento de entrada del sistema?

Solución:

Una clase que represente una de las siguientes opciones:

- El sistema global
- Empresa u organización global
- Un rol controlador del mundo real
- Un manejador artificial de los eventos del sistema

CONTROLADOR: EJEMPLO



CONTROLADOR

Aspectos a considerar en la elección del Controlador:

La misma clase para todos los eventos de un caso de uso

No asignar demasiada responsabilidad a un solo controlador

Desaturar controladores

Minimizar controladores humanos

PATRONES GRASP

Ingeniería de Software 1