Análisis y Diseño con Patrones

Práctica 1: Patrones de Análisis y Arquitectónicos

Una tienda que vende componentes hardware (ratones, pantallas, etc…) por internet nos ha pedido que le diseñemos una parte de un sistema software para gestionar las ventas de estos componentes.

Los clientes de la tienda se identifican con su dni y también proporcionan su nombre. Un subconjunto de clientes son socios de la tienda. Para estos, el sistema registra el número de socio y el año en el que se dieron de alta como socios de la tienda.

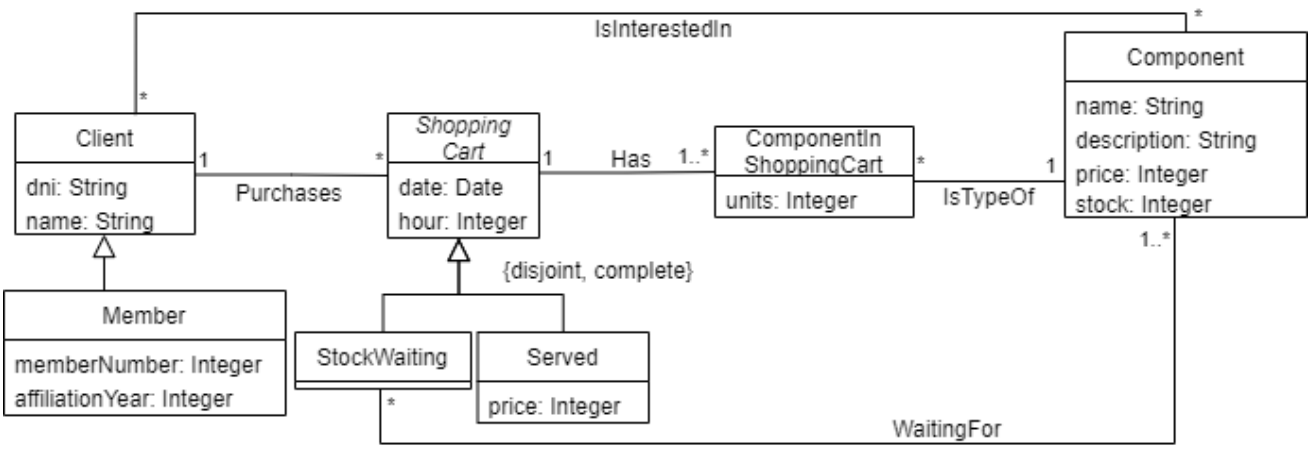
Los componentes que ofrece la tienda se identifican por su nombre y conocemos también su descripción, su precio y el stock. Por ejemplo, el componente con nombre RatonLogitechg34 tiene la descripción del ratón, un precio de 35 euros y un stock de 7 unidades.

Las compras que hacen los clientes se identifican por la fecha y la hora. Las compras pueden estar en espera de stock de alguno de los componentes o servidas. En el caso de las compras en espera de stock, el sistema registra los componentes que se están esperando. Asumimos que si la compra está en espera de un componente, está a la espera de todas las unidades del componente. En el caso de las compras servidas, el sistema almacena su precio.

Los clientes pueden mostrar su interés en componentes para recibir noticias de estos.

Disponemos ya de un análisis previo que podemos utilizar como punto de partida, pero habrá que corregir y mejorar. A continuación disponéis del diagrama estático de análisis por esta parte del software:

Diagrama estático de análisis



Claves de las clases:

- *Client: dni*
- *Member: memberNumber*
- *ShoppingCart: date + hour*
- *ComponentInShoppingCart: date + hour (ShoppingCart) + name (Component)*
- *Component: name*

Restricciones de integridad:

- El stock y precio de un componente es superior a 0.
- El número de unidades de un componente en una compra es superior a 0.
- Los componentes en espera de stock de una compra tienen que ser componentes de aquella compra.
- El precio de una compra servida es la suma de los precios de sus componentes multiplicado por la cantidad de los componentes. Si la compra la hace un socio, el importe se decrementarà en un 10%.

Pregunta 1 (15%)

Enunciado

Queremos diseñar la operación *getPrice(): Integer* de la clase *ShoppingCart* que calcula el precio de una compra. El precio de una compra es la suma de los precios de los componentes multiplicado por la cantidad de los componentes. Si la compra la hace un socio, este precio se decrementarà en un 10%. Se pide:

a) Propón el pseudocódigo de la operación. Hace falta que proporciones el pseudocódigo de todas las clases y operaciones involucradas en la definición de la operación. La operación tiene que cumplir con el principio de abierto y cerrado, sabiendo que en el futuro otros tipo de clientes serán añadidos y cada tipo de cliente podrá tener descuentos diferentes a los que tienen los socios.
b) Revisa el código anterior y explica claramente, con un ejemplo, como habéis garantizado el bajo acoplamiento y la alta cohesión.

Pregunta 2 (30%)

Enunciado

La tienda para la cual estamos diseñando este sistema software se ha dado cuenta que solo está registrando los intereses actuales de sus clientes en los componentes que vende. Para hacer acciones de marketing e incrementar las ventas de sus componentes, la tienda quiere registrar los intereses que tenían sus clientes en el pasado. Notáis que durante el tiempo, un cliente puede estar interesado en un componente, dejar de estarlo y volver a estar interesado.

Nos proponen buscar una solución que nos permita representar esta información en nuestro sistema. En concreto, se pide:

a) Aplica uno de los patrones de análisis que hay explicados a los materiales para representar esta información, explica cual y razona su conveniencia.
b) Propón una solución detallada en forma de diagrama estático de análisis (incluyendo los nuevos atributos que aparezcan, las restricciones de integridad y las multiplicidades de las asociaciones). Muestra solo las clases que cambien respecto al diagrama del enunciado.
c) Propón el pseudocódigo de la operación *interestedComponentsPerDate(date: Date): Set(String)* de la clase *Client* que devuelve los nombres de los componentes en los que estaba interesado el cliente en la fecha indicada en el parámetro. Podéis asumir que el parámetro es una fecha del pasado. Si lo necesitáis, podéis suponer la existencia de la operación *insidePeriod(initialDate: Date, endDate:Date, date: Date): Boolean* que devuelve cierto, si la fecha date está dentro del periodo [initialDate, endDate] y falso, en caso

contrario. Esta operación la podéis invocar desde donde la necesitáis y no hace falta que deis el pseudocódigo.

Pregunta 3 (15%)

Enunciado

La tienda de componentes decide expandir su negocio vendiendo, además de los componentes básicos (ratones, pantallas, etc…) que vendía hasta ahora, componentes ensamblados de componentes básicos y/u otros productos ensamblados. Por ejemplo, una CPU es un componente ensamblado de dos componentes básicos una ALU y una unidad de control. Un ordenador es un componente ensamblado de varios componentes básicos, como por ejemplo, un ratón y otros componentes ensamblados, como por ejemplo una CPU. Para simplificar, asumimos que en estos nuevos componentes ensamblados solo hay **una** unidad de cada uno de los componentes y/o componentes ensamblados que forman parte. Los componentes ensamblados, igual que los componentes básicos que se vendían hasta el momento, se identifican por su nombre, tienen una descripción, un precio y un stock. En un producto ensamblado no puede haber ni de forma directa ni de forma indirecta el mismo producto ensamblado.

Nos proponen buscar una solución que nos permita representar estos nuevos componentes ensamblados en nuestro sistema (podéis partir del diagrama estático de análisis del enunciado). En concreto, se pide:

a) Aplica uno de los patrones de análisis que hay explicados a los materiales para representar esta información, explica cual y razona su conveniencia.
b) Propón una solución detallada en forma de diagrama estático de análisis (incluyendo los nuevos atributos que aparezcan y las multiplicidades de las asociaciones). Muestra solo las clases que cambien respecto al diagrama del enunciado.

Pregunta 4 (10%)

Enunciado

Queremos definir la capa de presentación de nuestro sistema y aplicar el patrón de arquitectura Modelo, Vista y Controlador (MVC) para el caso de uso *Añadir un nuevo interés para un componente*. Este caso de uso permite a un cliente de la tienda, desde su área privada, seleccionar la funcionalidad/opción para añadir un nuevo interés. En respuesta, el sistema le mostrará todos los componentes en los que no está interesado actualmente el cliente. El cliente seleccionará el componente sobre el que quiere registrar su interés, el sistema registrará este interés y le mostrará el mensaje “Interest successfully registered”. Para hacer este ejercicio tenéis que partir del diagrama estático del enunciado.

Se pide:

a) Diagrama de clases del MVC para este caso de uso incluyendo las operaciones que sean necesarias (las operaciones tienen que contener los parámetros que necesiten y los tipos de retorno, si tienen). El diagrama de clases y las operaciones tienen que ser específicas para este caso de uso.
b) Describe brevemente qué hace cada operación que aparece en el diagrama de clases anterior.

Pregunta 5 (10%)

Enunciado

Propón el diagrama de secuencia o pseudocódigo del caso de uso *Añadir un nuevo interés para un componente* (del ejercicio anterior), desde que se muestra la pantalla para que el cliente pueda seleccionar del menú la funcionalidad/opción de añadir un nuevo interés para un componente hasta que el sistema le muestra el mensaje “Interest successfully registered”.

Pregunta 6 (10%)

Enunciado

Seguimos diseñando la arquitectura del sistema y hemos decidido utilizar una arquitectura con 3 capas: presentación, dominio y servicios técnicos (en nuestro caso, capa de de datos). Se pide:

a) Haced un diseño (con un diagrama de clases) donde se muestre cómo se podrían comunicar las clases que ya tenemos definidas (mostradas en el diagrama estático de este enunciado) y que pertenecen a la capa de dominio con la base de datos. Esta comunicación se tiene que hacer mediante la capa de datos y tiene que permitir obtener y guardar la información de las clases definidas a la capa de dominio. Hacedlo solo para una clase, por ejemplo, la clase *Client*, el resto de clases se podrían tratar de forma similar. La solución que propongáis tiene que satisfacer el principio de inversión de dependencias.
b) Explicad claramente cómo habéis garantizado, en el apartado anterior, el principio de inversión de dependencias.

Pregunta 7 (10%)

Enunciado

Para acabar de diseñar la arquitectura en 3 capas del sistema, usando las tres capas clásicas de muchos sistemas de información:

1. Presentación
2. Dominio
3. Servicios técnicos (Datos)

Queremos reflexionar sobre a qué capa pertenece cada una de las clases que han ido surgiendo en el enunciado (y a la solución que has elaborado hasta ahora):

- Clases del diagrama de clases del ejercicio 2.
- Clases del diagrama de clases del ejercicio 4.

- Clases del diagrama de clases del ejercicio 6.

a) Indica qué clases (e interfaces, si hay) pertenecen a la capa de dominio. Razona tu respuesta.

b) Indica qué clases (e interfaces, si hay) pertenecen a la capa de presentación. Razona tu respuesta.

c) Indica qué clases (e interfaces, si hay) pertenecen a la capa de servicios técnicos (en nuestro caso, capa de datos). Razona tu respuesta.