Práctica 3: Patrones de Diseño

Fecha y hora de entrega de las distintas partes de la práctica:

- **Opción 1: Práctica predefinida: 15 Mayo a las 10 de la noche.**
- **Opción 2: Práctica libre (sujeta a la pre-aprobación por parte del profesor de prácticas): 22 Mayo a las 10 de la noche.**
- *Las semanas de corrección de prácticas serán las del 16-20 de mayo (práctica predefinida) y las del 23-27 de mayo (práctica libre). La práctica deberá quedar disponible online como parte del portafolios virtual.*

Entregables:

1.- Diagrama completo de clases de diseño donde se hayan identificado los patrones GOF aplicados, comportamiento dinámico de cada uno de ellos y ventajas/inconvenientes de su uso.

2.- Código de la aplicación

3.- (opc): diseño y/o implementación corregida de la aplicación (debe incluir informe de mejoras)

4.- (solo práctica libre): Power Point con presentación de la aplicación diseñada

5.- El resto de entregables comunes a todas las prácticas: rúbricas de corrección, carga de trabajo y opinión de la actividad.

Evaluación: Rúbrica incluida con la práctica, y aplicada por el profesor. La no asistencia a clase de prácticas, la entrega en un directorio equivocado o la no entrega de la práctica en plazo hará que esta práctica sea evaluada como NO PRESENTADA. El acceso a la clase los días de la corrección se cerrará cinco minutos después del comienzo de la clase.

Herramientas y lenguajes a utilizar: Cualquier herramienta de modelado UML que soporte la versión UML 2.0. y cualquier lenguaje OO.

OPCIÓN 1: TPV

Supongamos que nos han pedido implementar un prototipo de un Terminal Punto de Venta (TPV) para una gran superficie de material de oficina. La arquitectura de la aplicación es de tres capas, y para el prototipo se sustituirá la interfaz gráfica de usuario y la base de datos de la aplicación por un sistema de E/S de ficheros XML.

Además, el prototipo debe implementar un solo caso de uso de negocio, el CU **ProcesarVenta**.

Caso de uso UC1: Procesar Venta
Alcance:
Aplicación TPV
Actor principal:
Cajero
Stakeholders e interesados:

- Cajero: desea introducción de datos rápida, precisa, sin errores de pago ya que las inconsistencias en caja se deducen de su salario
- Cliente: quiere comprar rápido y con esfuerzo mínimo. Le gusta ver en el display los items comprados con su descripción y precio
- Empresa: desea que se graben todas las transacciones y se cumplan los intereses del cliente.
- Hacienda Pública: quiere recaudar los impuestos de cada compra.

Precondiciones:
El cajero está autenticado en el sistema.
Postcondiciones:
La venta queda guardada. Los impuestos están calculados. Se genera un ticket de la compra.
Flujo básico:
1. El cliente llega a caja a pagar los productos que va a comprar
2. El cajero comienza una nueva venta
3. El cajero pasa el producto por el lector de código de barras (o introduce por teclado el código EAN-13)
4. El sistema graba la línea de venta y presenta en pantalla una descripción del producto, el precio y el importe total hasta el momento.
El cajero repite los pasos 3-4 hasta que indica finalización
5. El sistema calcula el total y los impuestos de la venta
6. El cajero pide el importe a pagar al cliente
7. El cliente paga en efectivo
8. El cajero introduce la cantidad pagada por teclado
9. El sistema presenta el importe a devolver
10. El cajero guarda el dinero del cliente y devuelve el cambio
11. El sistema registra el pago en caja
12. Se imprime en pantalla un recibo
13. Fin
Flujos alternativos:
1a. El ítem no existe en el catálogo: ignoramos el producto.
4a: El producto tiene algún tipo de descuento: el sistema calcula el precio con descuento

A partir del caso de uso **ProcesarVenta** se han extraído los siguientes contratos de operación:

.

1. Contrato de operación **Sistema::inicializar(…):** la inicialización se encarga de cargar el catálogo de productos y de leer el fichero de configuración para preparar el sistema
2. Contrato de operación **Sistema::crearNuevaVenta(…):** debe crear una nueva instancia de Venta y asociarla a la caja registradora donde se está produciendo dicha venta.
3. Contrato de operación **Sistema::anyadirLinVenta(…):** Incorpora la línea de venta a la Venta.
4. Contrato de operación **Sistema::cerrarVenta(…),** que permite calcular los descuentos y los impuestos.

Estos contratos se implementan dentro de una clase **controlador-fachada llamada Sistema.** El main() será el encargado de crear el controlador (un objeto de la clase Sistema), y a continuación ir llamando sucesivamente a sus métodos inicializar(), crearNuevaVenta(), anyadirLinVenta() y cerrarVenta().

El catálogo del sistema, **que contiene el código del producto, su descripción y su PVP (incluidos impuestos),** se encuentra ubicado en el servidor central de la empresa, y puede ser accedido a través de una URL que se encuentra, junto con el resto de parámetros de configuración del sistema, en un fichero de propiedades llamado **miAplicacion.properties**.

Ejemplo de miAplicacion.properties

```
#URL del catálogo
Catalogo.URL= http://www.dlsi.ua.es/~ccachero/catalogo.csv
# WSDL del servicio web de impuestos
CalculadoraImpuestos.WSDL=http://www.dlsi.ua.es/~rafamt/calculadora.wsdl
# Tamaño del catálogo en memoria
Catalogo.numItemsMemoria=100
#Dcto por tener tarjeta de fidelización la empresa (%)
DctoTarjetaFid.porcentaje=5
#Dcto por ser empleado
DctoEmpleado.porcentaje=15
#Dcto por comprar en días y horas especiales
DctoMiercoles.porcentaje=8
DctoMiercoles.horaInicio=10
DctoMiercoles.horaFin=12
#Política de cálculo de dcto global activa en la empresa
DescuentosAplicables.politicaAplicacion=NO_ACUMULABLE
```

Estudio de Variabilidad

Como parte de nuestro análisis, hemos preguntado a los analistas qué partes del sistema es probable que varíen en un futuro, y esto es lo que nos han contestado:

A: CÁLCULO IMPUESTOS. Queremos que nuestro sistema pueda interactuar con distintos sistemas (externos) de cálculo de impuestos. Estos sistemas de cálculo de impuestos nos devuelven, dada una línea de venta (tras haberle aplicado al precio los descuentos que correspondan), qué impuestos tenemos que pagar sobre ella. Sobre una misma venta podemos tener que pagar varios impuestos, por lo que el sistema de cálculo de impuestos nos debe devolver colecciones de líneas de impuesto. Como ejemplo, los productos como alcohol o tabaco están sujetos a impuestos especiales, además del IVA. Además, distintos tipos de producto están sujetos a distintos tipos de IVA. Actualmente el sistema de cálculo de impuestos que utiliza nuestra empresa tiene una interfaz de servicio web, aunque la empresa podría variar de sistema en un futuro.

B. REGLAS PRECIOS. Además, nuestro sistema debe poder soportar reglas de precios complejas.

B.1. La primera regla de precios establece distintos tipos de descuento según el tipo de cliente. E.g. en la actualidad a los clientes de TARJETA se les aplica un descuento de un 5% de descuento sobre todos los productos que compren, mientras que a los EMPLEADOS se les aplica un 15% sobre cualquier compra que realicen. Esta regla podría variar en un futuro.

B2: La segunda regla de precios establece descuentos de un 8% los miércoles de 10 a 12. Nuevamente, esta regla podría cambiar en un futuro, y/o añadirse nuevas franjas horarias.

B3: Además, en caso de que más de una regla de precios sea aplicable, la empresa desea que el sistema para permita configurar qué política aplicar.

- Una posibilidad sería aplicar una política de NO_ACUMULABLE, donde solo se aplica el descuento más ventajoso para el cliente (solo uno)
- Otra posibilidad sería aplicar una política ACUMULABLE, i.e. acumular los descuentos
- Estas posibilidades podrían sufrir variaciones en un futuro.

Por ejemplo, un cliente puede tener tarjeta del establecimiento y haber comprado el miércoles a las 11:20 En ese caso, nuestro sistema podría estar configurado de manera que el cliente solo pudiese beneficiarse de uno de los descuentos (en nuestro ejemplo, el 8% de los miércoles o el 5% por

tener tarjeta. Con una política de NO_ACUMULABLE, el cliente se beneficiaría finalmente de un 8% de dcto sobre el total. Con una política de ACUMULABLE, el cliente se beneficiaría de un descuento total de un 13%.

C. ROLLBACK DE COMANDOS
El sistema debe poder deshacer la creación de ventas y la creación de líneas de venta en cualquier momento. En un futuro, según se vayan añadiendo nuevas funcionalidades al sistema, se desea poder también añadir de manera sencilla nuevas capacidades para 'deshacer' el efecto de dichas acciones (e.g. deshacer la adición de un medio de pago, deshacer la apertura de una caja registradora, etc.).

D. SISTEMA DE E/S. Tanto el formato de entrada como de salida de datos del sistema va a ir cambiando a medida que se vayan sucediendo las iteraciones de desarrollo, por lo que necesitamos que dicha E/S esté desacoplada del resto del sistema.

Requisitos adicionales

El sistema debe redondear todos los precios, impuestos, etc. a la centésima más próxima.

Además, la empresa desea que cada establecimiento maneje una única instancia del catálogo de productos, que sea accesible desde cualquier parte del sistema, con el fin de asegurar la consistencia de los datos. Además, dado el tamaño potencial de dicho catálogo, se desea que el sistema inicialice el catálogo con los n primeros ítems (cuyo número se encuentra en el fichero de configuración), y a partir de ahí solo cargue nuevos productos en memoria a medida que se le van pidiendo productos que no están en esa caché.

E/S DEL SISTEMA

Para la introducción de la venta en el sistema, asumimos que de momento no vamos a implementar una interfaz gráfica de usuario. En su lugar, y para poder probar que todo funciona correctamente, el programa recibirá como entrada un fichero **XML** con el siguiente formato:

```
<venta>
     <fecha dia=”L|M|X|J|V|S|D” hora=”HH:MM”/>
          <linventa codProd=”X” cant=”Y”/>
          <linventa codProd=”X” cant=”Y”/>
     <cliente tarjetaFid=”true” empleado=”false”>
</venta>
```

Además, en cualquier parte del fichero de entrada pueden aparecer instrucciones del tipo `<deshacerLinVenta/>` (que deshace la última línea añadida a la venta) o `<cancelarVenta/>`, que cancela la venta. En un ticket pueden aparecer tantos `<deshacerLinVenta/>` seguidos como se quiera. Para una venta el programa puede como máximo deshacer tantas líneas de venta como hayan sido creadas. Cualquier solicitud de `<deshacerLinVenta/>` sobre una venta sin líneas es simplemente ignorada. Una venta abierta puede además ser cancelada en cualquier momento, en cuyo caso se debe eliminar la venta y finalizar la lectura del fichero, ignorando el resto del mismo. Una venta cancelada es por tanto equivalente a un fichero de entrada XML vacío, mientras que una venta donde se han deshecho todas las líneas de venta es equivalente a:

```
<venta>
</venta>
```

Como salida, el programa debe generar un fichero en formato XML con el siguiente formato:

```
<ticket>
     <linTicket>
          <descr>XXX</descr>
          <cant>X</cant>
          <pUnit>X</pUnit>
          <dctoLin>X</dctoLin>
          <pLin>X</pLin>
     </linTicket>
     <linTicket>
     …
     </linTicket>
     …
     <totalAPagar cant=”X”/>
     <dctoAcumulado cant=”X”/>
     <impuestos cant=”X”/>
</ticket>
```

Si la venta no tiene líneas de venta, el fichero de ticket generado será

```
<ticket>
</ticket>
```

Si la venta ha sido cancelada, el fichero de ticket estará vacío.

El programa debe ser ejecutable mediante la instrucción:

```
tpv <nombreFichVenta.xml> <nombreFichTicket.xml>
```

OPCIÓN 2: PRÁCTICA LIBRE

Los alumnos deberán proponer una especificación que incluya requisitos no funcionales solucionables mediante la aplicación de patrones GOF. El límite para proponer dicha especificación al profesor es el **11 de Abril**. A partir de esa especificación, los alumnos deberán definir un diseño que incluya los patrones relevantes. El diseño tendrá que ser nuevamente discutido con el profesor antes de Semana Santa (como muy tarde el **20 de Abril**). El profesor tendrá que aceptar de manera explícita tanto la especificación como el diseño para que la práctica se considere válida para la evaluación de la asignatura. Para que un sistema sea aceptable, debe presentar una complejidad suficiente y ser coherente. Como guía, la complejidad del sistema debería ser similar a la de la práctica obligatoria. Los alumnos que opten por esta modalidad presentarán su aplicación y los patrones utilizados en clase con ayuda de una presentación power point el último día de clase, y su evaluación tendrá en cuenta tanto los aspectos técnicos como la creatividad y originalidad de su propuesta y su fluidez a la hora de presentar el sistema en clase.

Criterio	Sobresaliente	Notable	Suficiente	Insuficiente
Corrección y completitud del Diagrama De Clases de Diseño (10%)	El Diagrama de clases tiene una estructura correcta e incluye todos los métodos con parámetros y tipos de retorno, atributos con tipos, roles y cardinalidades que aparecen en el código. Código y diagrama están sincronizados (1)	El diagrama de clases tiene una estructura correcta e incluye métodos con parámetros y tipos de retorno, atributos y roles (0.7)	El diagrama de clases tiene una estructura correcta e incluye métodos, atributos y roles (0.5)	El diagrama está incompleto o tiene errores graves de diseño (confusión en tipos de relaciones, etc.) (0)
Identificación y Aplicación correcta de los patrones (20%)	Todos los patrones han sido correctamente identificados y aplicados, y se ha JUSTIFICADO claramente el porqué de su uso (fuerzas) (2)[i]	Todos los patrones menos uno han sido correctamente identificados y aplicados, y/o se ha JUSTIFICADO suficientemente el porqué de su uso. (1.5)	Todos los patrones menos uno han sido correctamente identificados y aplicados (1)	No se han identificado correctamente al menos dos patrones, y/o no se ha proporcionado una explicación convincente del porqué de la elección de patrones realizada (0)
Ventajas/Inconvenientes Diseño (10%)	Se han razonado de manera clara, específica (en el contexto del sistema implementado) y concisa acerca de las ventajas/inconvenientes de todos patrones aplicados (1)	Se ha razonado de manera específica y concisa acerca de las ventajas/inconvenientes de todos los patrones aplicados (0.75)	Se ha razonado de manera específica y concisa acerca de las ventajas/inconvenientes de al menos tres de los patrones aplicados (0.5)	Se ha razonado de manera insuficiente o confusa acerca de las ventajas/inconvenientes de los patrones aplicados, o se han copiado ventajas/inconvenientes del patrón genérico (0)
Implementación y defensa (50%)	La práctica ha pasado a la primera todos los tests, y el alumno ha sabido contestar a todas las preguntas de seguimiento (4)	La práctica ha pasado los tests en la primera media hora de la clase, y el alumno ha sabido contestar a todas las preguntas de seguimiento (3)	La práctica ha pasado los tests tras trabajar en ella durante la clase, y el alumno ha sabido contestar a todas las preguntas de seguimiento (2)	La práctica no ha pasado los tests, y/o el alumno no ha sido capaz de contestar a todas las preguntas de seguimiento (0)
Presentación (10%)	La práctica presenta una presentación pulcra y sin faltas de ortografía. Los diagramas se han realizado con la ayuda de alguna herramienta (1)	La práctica presenta una presentación pulcra y con un máximo de dos faltas de ortografía. Los diagramas se han realizado con la ayuda de alguna herramienta (0.75)	La práctica posee una presentación aceptable y con no más de tres faltas de ortografía. Diagramas y texto son perfectamente legibles (0.5)	La presentación es sucia, desordenada, ilegible y/o con más de tres faltas de ortografía (0)

[i] Se considerarán patrones correctos aquéllos que, aunque no sean los de la solución original, se basen en un razonamiento de diseño convincente.