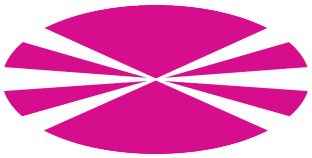

UNIVERSIDADE DA CORUÑA

Diseño Software

Práctica de Diseño (2019-2020)

INSTRUCCIONES:
Fecha límite de entrega: 13 de diciembre de 2019 (hasta las 23:59).

- **Los problemas se han de resolver aplicando principios y patrones de diseño. No será válida una solución que no los use.**
- **Informe**: Para cada problema hay que hacer un informe en el que se incluya:
  - Explicación de los **principios de diseño** usados (en particular los **SOLID**) y dónde en concreto se han usado (nombrar clases específicas de vuestro código).
  - Explicación del **patrón o patrones de diseño** usados. Para cada patrón se incluirá lo siguiente:
    - **Breve explicación del patrón elegido** y justificación de su utilización.
    - **Diagrama de clases** en el que se muestren las clases involucradas en el patrón. Es importante señalar en el propio diagrama el rol que juega cada clase propia en el patrón con anotaciones UML.
    - **Diagramas dinámicos** (secuencia, comunicación o estados) que muestren el funcionamiento dinámico de aspectos fundamentales del código. Deberéis decidir qué tipo de diagrama es el más adecuado para cada problema.
- **Código y forma de entrega**:
  - El código desarrollado debe reflejar fielmente el diseño del informe.
  - Deberá incluir pruebas y se debe comprobar la cobertura de las mismas.
  - Se entregará en vuestro repositorio SVN en un proyecto de NetBeans cuyo nombre sea vuestro grupo con el sufijo `-PD`. Por ejemplo: `DS-11-01-PD`.
  - La documentación explicando el diseño y los principios y patrones utilizados en ambos problemas se entregará como un fichero `PDF` dentro de un directorio `doc` del proyecto NetBeans.
- **Evaluación**:
  - Esta práctica corresponde a 1/3 de la nota final de prácticas que consistirá en una evaluación de la memoria y el código según los siguientes criterios.
  - **Calidad de la documentación**: selección del patrón y principios adecuados, explicaciones claras de su uso, calidad y claridad de los diagramas entregados, correspondencia con el código, etc.
  - **Calidad del código**: Aplicación correcta de los patrones y principios, seguimiento correcto de la filosofía orientada a objetos, correspondencia con el diseño, pruebas adecuadas, etc.

1. **Gestión de máquinas de *vending***

   Tenemos un código desarrollado que implementa el funcionamiento de una máquina de *vending* (máquina electrónica auto-expendedora para la venta de diversos productos). Inicialmente existe una clase `VendingMachine` que ofrece en su interfaz los siguientes métodos:

   - `void insertProduct(String product, int price)`. Inserta un nuevo producto en la máquina con su descripción y su precio
   - `void insertCoin(EuroCoin e)`. Introduce una moneda de Euro en la máquina (aprovecha el tipo enumerado que hayas hecho en ejercicios anteriores).
   - `List<EuroCoin> buy(String product)`. Compra el producto indicado usando las monedas introducidas previamente devolviendo una lista de monedas que representa el cambio.
   - `List<EuroCoin> cancel()`. Cancela la compra devolviendo las monedas introducidas previamente.

   Nuestra versión original de la máquina de *vending* incluye un sistema de **cambio simple** que consiste en aceptar unas monedas de entrada, esperar una notificación de gasto (la compra de un producto) y después devolver al cliente las monedas que no hayan sido utilizadas de aquellas que inicialmente se insertaron. De tal forma que se introducen tres monedas de 1€ y el gasto fue de 1,50€ al final se devuelve sólo una moneda de 1€). Este sistema se considera adecuado para máquinas que pueden quedar expuestas por la noche al vandalismo, ya que no almacenarán muchas monedas en su interior.

   Una vez implementado el sistema nos piden realizar una nueva variante para calcular el cambio, el **cambio depósito**, que consiste en que algunas máquinas tendrán un deposito adicional de monedas que se usará para dar siempre el cambio exacto. Este sistema será adecuado para máquinas situadas en entornos vigilados (un centro comercial, por ejemplo).

   También nos indican que en el futuro **se plantean nuevas modificaciones en la forma de dar el cambio**, por ejemplo, podría redondearse para evitar tener que usar monedas de 1 o 2 céntimos de euro. Incluso una máquina de *vending* ya existente **podría dinámicamente modificar su sistema de cálculo del cambio**. Por ejemplo, por el día la máquina da cambio con depósito de monedas, pero por la noche, después de recoger la recaudación, da un cambio simple. También una máquina de cambio con depósito, que se quede sin monedas, puede decidir cambiar su tipo de cambio al cambio simple.

   Desarrolla una solución, basada en principios y patrones de diseño, que nos permita representar adecuadamente la máquina de `vending` y los distintos tipos de cambio. Permitiendo que a una máquina se le pueda cambiar dinámicamente su tipo de cambio y permitiendo la adición sencilla de nuevos tipos de cambio en el futuro.

2. **Cotización de acciones en el mercado bursátil**

En el mercado bursátil las acciones de los distintas compañías cotizan a unos determinados precios que van fluctuando a lo largo del día. De una determinada acción estamos interesados en saber la siguiente información:

- **Símbolo**: Hasta cuatro letras que identifican a la acción. Por ejemplo el símbolo de Apple es `AAPL`, el de Microsoft `MSFT`, etc.
- **Cierre**: Cotización de la acción al cierre del mercado.
- **Máximo**: Cotización máxima alcanzada por la acción ese día.
- **Mínimo**: Cotización mínima alcanzada por la acción ese día.
- **Volumen**: Número de acciones negociadas ese día.

Existen muchos clientes que están interesados en obtener los datos correspondientes a las distintas acciones. Pero no todos los clientes están interesados en la misma información. Por ejemplo, puede haber **clientes sencillos** que solo están interesados en el precio de cierre de la acción para luego mostrarla en un teletipo bursátil (el típico *ticker* que aparece en las noticias de los canales con información económica).

Por otro lado, pueden existir **clientes detallados** que necesitan toda la información de una determinada acción, para mostrarla por ejemplo en aplicaciones dedicadas a mostrar información bursátil como en la imagen de arriba.

Otros clientes **pueden estar solo interesados en los datos de una acción o acciones en concreto** y no en todos los datos de la información bursátil. En el futuro pueden fácilmente aparecer nuevos clientes con distintos requisitos.

Desarrolla una solución, basada en principios y patrones de diseño, que nos permita representar adecuadamente a los distintos clientes mencionados, permitir la fácil incorporación de nuevos y que, cada vez que haya nuevos datos bursátiles, todos los clientes interesados en dicha información se actualicen. En caso de haber distintas variaciones del patrón indica cuál en concreto estás usando y por qué.