UNIVERSIDADE DA CORUÑA

## Diseño Software

# Práctica de Diseño (2022-2023)

**INSTRUCCIONES:**
**Fecha límite de entrega: 16 de diciembre de 2022 (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/es de diseño** usados explicando para cada uno:
    - **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 el rol que juega cada clase propia en el patrón con anotaciones UML en el propio diagrama.
    - **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**:
  - Los ejercicios se entregarán usando *GitHub Classroom*. En concreto en el *assignment* `2223-PD` del *classroom* `GEI-DS-614G010152223-OP1`.
  - Se entregará un proyecto de IntelliJ (ficheros de configuración incluidos) con el nombre del repositorio de *classroom* y dos paquetes (`e1` y `e2`) por cada ejercicio.
  - Deberá incluir pruebas y se deberá comprobar la cobertura de las mismas.
  - 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 IntelliJ IDEA.
- **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. **Sistema de compra online**

   Para la gestión de pedidos de un sistema de compra online planteamos un diseño que permite realizar el proceso en los siguientes pasos:

   - *ShoppingCart*: Se podrá añadir ítems al carrito siempre y cuando el stock disponible del producto seleccionado sea suficiente. Una vez añadidos ítems al carrito será posible eliminarlos. En el momento en el que se desee terminar la compra será posible realizar el *check-out* para completar el proceso de compra.
   - *CheckOut*: En este punto el cliente puede decidir volver al paso previo y continuar la gestión del carrito (añadir nuevos ítems). Aun así, para evitar olvidos o confusiones por parte de los clientes, el sistema proporciona la opción de eliminar ítems seleccionados en la fase previa o bien modificar su cantidad (siempre y cuando el stock lo permita).
   - *Payment*: Concluida la selección de productos y sus respectivas cantidades, el cliente realiza el pago de su pedido, quedando de esta forma confirmado/pagado.
   - *Cancelled*: Durante un periodo de 24 horas desde que el cliente efectúa el pago de su pedido, el sistema permite su cancelación.
   - *Completed*: Efectuado el pago y pasadas 24 horas sin que se realice una cancelación, el pedido entra en preparación y se considera como completado.

   La clase `Product` debe incluir, entre otros, un atributo stock que permita conocer el número de ítems disponibles. Todas las operaciones que afecten al stock de un producto, deben actualizar el mismo de forma adecuada.

   La clase `Order` contará con el método `screenInfo()` que proporciona una información básica requerida para la aplicación. A continuación, se indica los datos a mostrar para cada fase por la que pasa el pedido y el formato solicitado. Además, se facilitan algunos ejemplos concretos.

```
* Al iniciar un nuevo pedido
Ejemplo: Order Number: 1111
Phase: Shopping -- Welcome to online shop

* ShoppingCart, CheckOut: número de productos en el carrito
Ejemplo 1: Order Number: 1111
Phase: Shopping -- 3 products
Ejemplo 2: Order Number: 1111
Phase: Chek Out: 4 products

* Payment order: número de productos en el carrito, hora del pedido
Ejemplo: Order Number: 1111
Phase: Paid Order: 4 products -- date 2022-10-31 19:06:37

* Completed order: número de productos en el carrito
Ejemplo: Order Number: 1111
Phase: Completed Order: 4 products

* Cancelled o Completed order
Ejemplo:  Order Number: 1111
Phase: Cancelled Order
```

   Además, es preciso registrar cada evento (un cambio en el carrito de compra, una modificación en la fase del pedido, etc.). De esta forma se facilita la obtención de

un *log* con la información sobre el pedido. Teniendo en cuenta el formato indicado para el registro de eventos, un ejemplo de dicho *log* de salida podría ser el siguiente:

```
Order 1111: Shopping Phase
- Add: Item: 111 - Quantity: 10 -> Shopping Cart -- Products : 1
- Add: Item: 222 - Quantity: 2 -> Shopping Cart -- Products : 2
- Add: Item: 333 - Quantity: 3 -> Shopping Cart -- Products : 3
- Add: Item: 444 - Quantity: 4 -> Shopping Cart -- Products : 4
- Remove: Item: 444 -> Shopping Cart -- Products : 3
- Add: Item: 444 - Quantity: 5 -> Shopping Cart -- Products : 4
Order 1111: Check Out Phase
- Modify: Item: 111 - Quantity: 1 -> CheckOut Order -- Products: 4
Order 1111: Payment Phase
```

En el futuro se plantea ampliar el problema para tener en cuenta también el proceso de entrega, es decir, saber si el producto se ha entregado al cliente, o se ha entregado a un intermediario (correos) esperando a ser recogido, ha sido rechazado, se ha perdido en la entrega, etc.

**Desarrolla una solución, basada en principios y patrones de diseño, que permita representar adecuadamente cada pedido y sus fases, representando adecuadamente los cambios entre ellas, y que permita añadir fácilmente nuevas fases en el futuro para nuestro sistema de venta online básico.**

2. **Gestión de alertas de los parámetros de los tanques de un acuario**

   Nos han encargado un nuevo sistema de gestión de alertas para el control de los parámetros de los tanques del Aquarium Finisterrae de A Coruña.

   Cada tanque tiene instalados una serie de sensores. Cada sensor se encarga de medir un único parámetro numérico de manera periódica. Los sensores pueden ser, por ejemplo, para medir el nivel de oxígeno, de PH o de temperatura. Para simular un cambio en un parámetro se hará uso del método set de la clase que lo represente.

   Para cada sensor se pueden configurar una serie de alertas, que pueden ser de dos tipos: naranjas y rojas. Una alerta se asocia a un único sensor, aunque un sensor puede tener varias alertas asociadas. Cada alerta define dos rangos numéricos. El primer rango indica los valores normales del parámetro medido por el sensor. Si el valor se encuentra dentro de este rango, la alerta no se dispara, es decir, está desactivada. Si el valor se sale de dicho rango, se dispara la alerta naranja. El segundo rango, más amplio que el primero, define los niveles mínimo y máximo de la alerta naranja, si el valor se sale de ese rango se dispara la alerta roja.

   En el acuario existen dispositivos de control actuadores instalados en los tanques. Estos dispositivos están asociados a una serie de alarmas de alertas de forma que cuando se dispara una alerta naranja o roja pueden actuar de manera instantánea en el tanque.

   Además, existe diverso personal encargado del mantenimiento de los tanques. Cada persona puede suscribirse a un conjunto determinado de alertas. Cuando se dispara una alerta naranja o roja se añade un informe al personal suscrito. Los informes se almacenan en una cola de prioridad (según el tipo de alerta) para que puedan ser tratados durante las tareas de mantenimiento. Cada informe contendrá la siguiente información: tipo de alerta (roja o naranja), nombre y ubicación del tanque, nombre de la alerta, nombre del parámetro, nivel del parámetro y fecha y hora de la alerta. Solo se comunica una alerta si esta cambia de estado excepto si se desactiva, en este caso no se comunica. Se muestra a continuación un posible informe de ejemplo:

```
Alertas de Mantenimiento Focas
Alertas ROJAS:
* Alerta ROJA:
  Piscina de las focas, Exterior
  Control de oxígeno focas: parámetro Oxígeno, nivel -1,000
  17:08:09 11/11/2022

Alertas NARANJAS:
* Alerta NARANJA:
  Piscina de las focas, Exterior
  Control de oxígeno focas: parámetro Oxígeno, nivel 4,000
  17:08:09 11/11/2022

* Alerta NARANJA:
  Piscina de las focas, Exterior
  Control de oxígeno focas: parámetro Oxígeno, nivel 3,000
  17:08:09 11/11/2022
```

   El sistema tiene que estar preparado para incluir fácilmente nuevos sensores que generen alertas (por ejemplo, el nivel del agua) y nuevos elementos que recojan dichas alertas y actúen ante ellas.

   **Desarrolla una solución, basada en principios y patrones de diseño, que nos permita resolver eficientemente el problema planteado.**