Desarrollo de Software
Prof. Carlos Alonzo
Primer Examen Parcial
26-04-2024 **Nombre:** _____

El examen es individual. Lea con detenimiento y tranquilidad cada uno de los enunciados. Justifique sus respuestas de manera clara y resumida.

1 Pregunta 1 (4 puntos)

(Selección Simple) Usted está desarrollando un framework de cotización de acciones. Alguna de las aplicaciones que utilizan este framework querrán cotizaciones de acciones para mostrarse en una pantalla cuando estén disponibles; otras aplicaciones querrán que nuevas cotizaciones activen ciertas aplicaciones financieras. Sin embargo, es posible que otras aplicaciones quieran ambos de los anteriores. ¿ Cuál patrón se debe usar de modo que varias piezas diferentes del código de la aplicación puede reaccionar a su manera a la llegada de nuevas cotizaciones?

- ☐ Command
- ☐ Chain of responsability
- ☐ Observer
- ☐ Decorator

(Selección Simple) Tu programa manipula imágenes que ocupan mucho espacio en memoria. ¿ Cuál patrón debes usar en tu programa para que las imágenes solo estén en la memoria cuando son necesarias, y en caso contrario, solo se puede encontrar en archivos?

- ☐ Decorator
- ☐ Memento
- ☐ Proxy
- ☐ State

(Selección Múltiple) Marque cuáles de las siguientes son ventajas del patrón Decorador:

- ☐ Los decoradores son componentes reutilizables. Puede crear una libreria de clases de decorador y aplicarlas a diferentes objetos y clases según sea necesario, reduciendo la duplicación de código.
- ☐ Los decoradores usan el principio de favorecer la composición sobre herencia: a diferencia de la herencia tradicional, que puede conducir a una jerarquía de clases profunda e inflexible, el patrón decorador utiliza la composición. Puede componer objetos con diferentes decoradores para lograr la funcionalidad deseada, evitando los inconvenientes de la herencia, como el acoplamiento fuerte y las jerarquías rígidas.
- ☐ El orden en que se aplican los decoradores no afecta el comportamiento final del objeto. Gestionar el orden de los decoradores no representa un problema.
- ☐ Debido a que es fácil agregar decoradores a los objetos, existe el riesgo de usar en exceso el patrón Decorador, lo que hace que el código base sea innecesariamente complejo.

(Selección Múltiple) Marque en cuáles de los siguientes casos de uso se puede usar el patrón Memento:

- ☐ Cuando necesita implementar una función de deshacer (undo) en su aplicación que permita a los usuarios revertir los cambios realizados en el estado de un objeto.

- ☐ Cuando necesita guardar el estado de un objeto en varios momentos para admitir funciones como control de versiones.
- ☐ Cuando el estado del objeto es inmutable, hay beneficios al usar el patrón Memento para capturar y restaurar su estado.
- ☐ Cuando necesita revertir cambios en el estado de un objeto en caso de errores o excepciones, como en transacciones de bases de datos.
- ☐ Cuando desea almacenar en caché el estado de un objeto para mejorar el rendimiento.

2 Pregunta 2 (4 puntos)

Jacinto Chapuza es un joven programador latinoamericano reconocido por su ignorancia de las mejores prácticas de programación, ni hablar del uso de los principios SOLID, tal como se evidencia en el siguiente código. Jacinto desarrolló una clase que se encarga de autenticar las credenciales de un usuario contra una base de datos. Ni siquiera se le pasó por la mente, que en el futuro de puede validar contra otro sistema, por ejemplo un servidor LDAP u otro sistema externo.

```
class Authentication
{
   private connection;
   public construct(connection: Connection){
        this.connection = connection;
    }

   checkCredentials(username: string, password: string): void {
        let user = this.connection.fetchQuery('SELECT * FROM users WHERE username = ?', username);
        if (user === null) { throw new InvalidCredentialsException('User not found'); }
       // validate password
    }
}
```

Evidentemente se trata de un diseño deficiente y usted debe mejorarlo para que cumpla con los siguientes principios SOLID: SRP, OCP y DIP. En particular, debe poder cumplir a cabalidad el principio de inversión de dependencias, para que a futuro la responsabilidad de autenticación se haga no contra una base de datos, sino contra un archivo de texto, un servidor LDAP u otro medio. Debe mostrar su diseño en un **Diagrama de Clases UML** con todos los detalles necesarios para corroborar que cumple con los Principios SOLID requeridos.

Desarrollo de Software
Primer Examen Parcial

3 Pregunta 3 (6 puntos)

El bodegón **Merca2** tiene las siguientes políticas vigentes hoy (viernes):

1. Política de un 20 % de descuento para personas mayores de 65 años
2. Descuento para clientes especiales del 15 % en ventas superiores a $400
3. El lunes, hay $50 de descuento en compras superiores a $500
4. Compra 1 caja de Bombones La Muerte y obtenga un 15% de descuento en todo

Es decir, existen estrategias de precios que se vinculan a la venta en virtud de tres factores:

1. Período de tiempo (por ejemplo, lunes o viernes, lo el día que sea)
2. Tipo de cliente
3. Una línea de producto en particular (por ejemplo, los bombones)

Supongamos que una persona mayor que también es cliente especial compra 1 caja de los bombones y 600 dólares en bolsas de papas fritas (definitivamente tiene problemas de ansiedad). ¿Qué política de precios se debe aplicar?.

Parte de la respuesta a este problema requiere definir la estrategia de resolución de conflictos del bodegón. Por lo general, el bodegón aplica la estrategia de resolución de conflictos **"lo mejor para el cliente"** (el precio más bajo), pero esto no es obligatorio y podría cambiar. Por ejemplo, durante un período de crisis, es posible que el bodegón tenga que utilizar una estrategia de resolución de conflictos de **"precio más alto"**.

El primer punto a tener en cuenta es que pueden existir múltiples estrategias coexistentes, es decir, una venta puede tener varias estrategias de precios. Otro punto a tener en cuenta es que una estrategia de precios puede estar relacionada con el tipo de cliente (por ejemplo, una persona mayor).

De manera similar, una estrategia de precios puede estar relacionada con el tipo de producto que se compra (por ejemplo, los bombones).

Usted debe modelar su diseño construyendo el **Diagrama de clases UML** de la solución utilizando los dos (2) patrones de diseño que están involucrados en este problema: el patrón **Composite** y el patrón **Strategy**. Puede justificar su diseño agregando la implementación en TypeScript de alguno de los métodos principales, si lo considera necesario. Es muy importante que defina con claridad el nombre las clases y especifique los métodos (con los tipos de los paramétros y valor de retorno).

4 Pregunta 4 (6 puntos)

Usted debe modelar el siguiente problema, usando sus conocimientos de Programación Orientada a Objetos, Genéricos y Patrones de Diseño.

Considere que usted dispone de una abstracción **Box**, que representa una caja que puede contener dentro un valor de cualquier tipo (NO puede modelar con **any** ni con **unknow**). Adicionalmente, tiene otra abstracción que representa un contenedor de cajas (**BoxContainer**) que posee la responsabilidad de agregar una caja (**arrangeBox**) al contenedor. Por último, definiremos un contrato o interface que llamaremos **BoxArranger** que se encarga de modelar la manera como se agrega una caja. Por ejemplo, una manera de agregar una caja es colocándola de primero en el contenedor, o colocándola de última.

Desarrollo de Software
Primer Examen Parcial

Usted debe construir su diseño teniendo en cuenta que la solución correcta se modela con uno, y solo uno, de los siguientes patrones: ***Iterator, Bridge o Decorator***. Su respuesta debe presentarla en un **Diagrama de Clases UML** mostrando el código de los métodos principales. También, es importante tener presente que si no se colocan los tipos de los parámetros y el tipo del valor de retorno de los métodos, se considera que su diseño es deficiente o incompleto. Si selecciona el patrón incorrecto para modelar este problema, la calificación será cero.

5 Bonus: it is worth a try

Match Time !