Desarrollo de Software
Prof. Carlos Alonzo
Segundo Examen Parcial
31-05-2024 **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: SOLID, Patrones de Diseño y genérico Either (6 puntos)

Nos han asignado para desarrollar un nuevo sistema CRM y que dentro de su funcionalidad está la creación de órdenes de venta, las órdenes de venta serán objetos que representan los artículos que le venderemos a nuestro cliente y la información de nuestro cliente como su teléfono, dirección, información crediticia, etcétera.

También debemos diseñar un mecanismo que nos permita validar que la orden está correctamente armada. Las reglas para la creación de una orden son las siguientes:

1. El cliente deberá tener el estatus de Activo
2. Se validará la dirección del cliente dejando como obligatorio el campo dirección, código postal y país
3. Se validará el teléfono del cliente por lo cual se tendrá que tener el prefijo de la operadora (3 dígitos) y el teléfono a siete dígitos
4. Validación de la línea de crédito del cliente, por lo que el total de la venta, más el saldo actual no deberá ser mayor que el límite de crédito del cliente
5. Se validarán los items de la orden de compra asegurándonos de que la cantidad de productos sea mayor que cero y el precio de venta del producto no sea 20 por ciento menor al precio en la lista de precios de los productos

Se nos ha solicitado de momento estas reglas, sin embargo, nos indica que las órdenes son objetos muy complejos que pueden tener más cosas dentro y que de seguro podrían necesitar validaciones adicionales, es por eso que nos han pedido que diseñemos una solución que le permita realizar las validaciones y estar diseñado para agregar nuevas en un futuro próximo.

Como ya vimos, una orden es un objeto complejo que puede tener varias cosas dentro que necesitarán validación y todas estas reglas junto con las nuevas deberán ser soportadas sin tener un gran impacto sobre la solución. **Usted debe usar el patrón de diseño Cadena de Responsabilidad - Chain of Responsability**. Para darle un poco más de emoción a este problema hemos definido un *Tree of Responsability (Árbol de Responsabilidad)* el cual en lo único en que varía es que en vez de tener un sólo Handler como siguiente objeto a

procesar el mensaje, esta variante nos permite tener varios Handler como siguiente objeto a procesar el mensaje.

En el patrón Cadena de Responsabilidad original, los validadores terminan su ejecución cuando un error es encontrado, en ese momento el código cliente es informado del error. Usted debe aplicar un cambio para que en lugar de regresar el primer error, éstos sean almacenados para ser devueltos todos los errores al código cliente. Su diseño debe permitir al cliente obtener todos los errores encontrados durante las validaciones, en caso de existir dichos errores. **Usted debe usar la abstracción genérica** $Either < L, R >$.

Usted debe modelar el problema planteado **completamente** mediante un **Diagrama de Clases UML** especificando con claridad la firma de los métodos con sus parámetros y valores de retorno. **Su solución NO debe hacer uso de abstracciones genéricas definidas por usted, solo necesitará usar el Either**.

2 Pregunta 2: Modelo de dominio, Servicios de Dominio y Patrones de Diseño (4 puntos)

Considere que usted participa en el desarrollo de la novedosa aplicación **DuBook** cuyo lanzamiento será el próximo primero de junio. Esta aplicación funciona como una biblioteca virtual, donde los usuarios que se registran pueden pedir prestados los libros por varios días para leerlos. No todos los libros se pueden pedir prestados por varios días, es decir, hay libros que solamente se pueden leer durante el día, si estaba disponible.

Puede ocurrir que un lector (usuario) no puede pedir prestado el libro ya que dispone de alguna restricción en la aplicación o plataforma (usuario inhabilitado). Las copias de un libro podrían ser finitas (solo disponer de un número limitado de ejemplares) o el lector puede tener ya el número máximo de libros que puede pedir prestado.

La aplicación **DuBook** posee un sistema de recomendaciones. El sistema de recomendación ayuda a los usuarios en la búsqueda de algo relacionado con sus gustos. Los sistemas de recomendación manejan el problema de la sobrecarga de información que encuentran los usuarios, proporcionándoles contenido personalizado y dirigido. Para construir estos sistemas, se han desarrollado distintos enfoques. Pueden ser filtrado colaborativo, filtrado basado en contenido o filtrado híbrido. El filtrado colaborativo recomienda libros identificando a otros usuarios con gustos similares. Por otro lado, el filtrado basado en contenido recomienda libros que son similares en contenido a aquellos que el usuario apreció en el pasado o que coinciden con los atributos del usuario. El método híbrido, como su nombre lo

indica, combina técnicas colaborativas y basadas en contenido para reducir y superar algunas limitaciones de cada enfoque.

Usted debe identificar del modelo de dominio del problema descrito los siguientes:

Modelo de Dominio	
Preguntas	***Respuestas***
¿ Cuáles entidades u objetos del dominio identifica ?	
¿ Cuáles servicios de dominio identifica en el enunciado ? Utilice nombres para sus servicos con valor semántico en el dominio	
¿ Cuál es el patrón de diseño que utilizará para modelar las recomendaciones ?	

3 Pregunta 3: Modelo de dominio, AOP y Arquitectura Hexagonal (10 puntos)

Usted debe modelar en un **Diagrama de Clases UML** el siguiente problema, usando todos sus conocimientos de Programación Orientada a Objetos, Programación Genérica, Programación Orientada a Aspectos (AOP), Patrones de Diseño y Arquitectura Hexagonal, y es mandatorio el uso de los genéricos $Optional < T >$ y $Either < L, R >$. **Su respuesta se considera válida si especifica los métodos con sus parámetros y valores de retorno**.

Considere una aplicación donde los usuarios pueden hacer publicaciones escritas (un *post*).

Desarrollo de Software
Segundo Examen Parcial

Por simplicidad ese *post* esta conformado de un texto y número de **likes**. Usted debe modelar el caso de uso que se describe a continuación, que corresponde a **denunciar un post**:

1. El usuario que denuncia un *post*, debe escribir obligatoriamente un texto explicativo y seleccionar una categoría para la denuncia (categorías predeterminadas)

2. Es importante tener presente que **no** todas las denuncias proceden, por ende solo se registran las denuncias que si procedan

3. En este caso de uso es importante llevar un log especial donde se registran todas las excepciones que se generen cuando se intenta hacer una denuncia. Es obvio, que si se produce una excepción, pues la denuncia no procede y no se registra

4. Se nos exige que nuestro diseño este preparado para ser robusto frente a un *Man-in-the-middle attack*. **No** se le ocurra preguntar qué signifca este ataque, ya vimos en clases como su diseño debe prepararse para esto.

5. Cuando la denuncia procede, el autor del post recibe un email donde se le indica que una publicación suya ha recibido una denuncia. En dicho correo se indica el username del denunciante, y se comparte el texto de la denuncia y la categoría

6. Las denuncias para que procedan deben cumplir ciertos criterios. Por ahora, describimos las reglas que nos interesan implementar en el diseño: si el *post* tiene más de 1000 likes, la denuncia no procede. El texto que escribe el denunciante se pasa junto a la categoría a un servicio de AI (*Artificial Inteligence*) de ChatGPT que se encarga de analizar si hay coherencia entre el texto de la denuncia y la categoría indicada. Si hay coherencia (hace *match*), la denucia procede automáticamente. En el caso contrario, se desestima la denuncia.

7. Los usuarios pueden estar en alguno de estos tres (3) estados: *active*, *warned* o *suspended*. Esto se modela como un tipo enumerado asociado al usuario.

8. Tenga presente que si se recibe una denuncia y el autor se encuentra en estado *warned*, la denuncia se acepta automáticamente si la validación AI es positiva.

Desarrollo de Software
Segundo Examen Parcial

Usted debe hacer uso en su diseño, de manera acertada y pertinente, de las siguientes abstracciones (patrón Observador):

```
interface Listener<T> {
 update(message: T) : void;
}
```

```
class Producer<T> {
 private _listeners: Listener<T>[] = [];
 public notify(message: T) {
  this._listeners.forEach(
   l => l.update(message)
  );
 }
}
```

Recuerde hacer uso de las excepciones de dominio, de los DTOs y los repositorios pertinentes.

RECUERDE que es **mandatorio** que al menos en pseudo-código o lenguaje natural pueda enunciar las responsabilidades de los servicios de dominio o aplicación que usted considere en su diseño.