PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IIC2113 - Diseño Detallado de Software (2020-2)

Pauta Interrogación 2

Indicaciones

- **La prueba es individual**. Si se detecta copia será evaluado con nota mínima.
- El formato aceptado será PDF. Pueden elegir si escriben sus respuestas en un procesador de texto, un markdown, en papel y escanearlo, o lo que les acomode. Lo importante es que al final lo unan para subir un archivo PDF que sea **legible**.
- Pueden usar cualquier material (apuntes) que les acomoden para responder la prueba, siempre que si usan material externo al del curso lo citen. **Si se suben respuestas sacadas de internet sin su respectiva cita, se considerará como copia**.
- Todas las preguntas deben ser vía Issues de Github con el tag [I2 Teórico]. Solo se aceptarán dudas de enunciado. Así mismo, solo el equipo docente puede responder dudas de este tipo.
- Durante el horario de clases, desde las 14:00 hasta las 17:00, en el zoom de la clase correspondiente, se leerá el enunciado y se responderán consultas del mismo. Esta sesión será grabada y subida inmediatamente después al canal de Youtube del curso.
- Si alguien sufre un problema de fuerza mayor durante el día Viernes de la interrogación, deben escribir cuanto antes a mfsepulveda@uc.cl

Plazo de entrega: 14 de Noviembre a las 9:00 am vía buzón de tareas en Canvas.

Pregunta 1 [15 puntos totales]

Responda las siguientes preguntas justificando el por qué de su respuesta (4 puntos cada respuesta). Además explique cómo llegó a esta conclusión citando la fuente (1 punto). Considere no más de 5 líneas en cada una de sus respuestas, y una sola línea referenciando la fuente (clases, lectura, sitio, etc.-).

a) ¿Cuál es la ventaja de usar Domain Driven Design? ¿Por qué?
Otorgar un lenguaje común a todos los miembros del equipo, tanto al equipo técnico como a los expertos del dominio, a través de abstracciones del modelo o problema que se está solucionando.
Se considera el puntaje completo si:
   - Menciona de alguna forma que facilita la comunicación (2 puntos).
   - Menciona acerca de la abstracción del problema (2 punto).
   - **NO** se da el punto base si es que:
      - No cita la fuente. Algunas fuentes posibles: clases, charla de buda, lecturas (podrían haber más si lo investigan).
      - Escribe más de 5 líneas, por no leer las instrucciones.

b) El libro Clean de Uncle Bob salió en 2008. A 12 años de su salida ¿sigue vigente? ¿Por qué?
Siguen siendo vigentes sus bases respecto a las buenas prácticas y principios fundamentales, ya que siguen siendo aplicables. No es un manual o un libro de recetas de cocina, ya que el software va cambiando a gran velocidad, por lo que es importante al usar este tipo de referencias tener esta consideración.
Se considera el puntaje completo si:
   - Da lo mismo si defiende si está vigente o no. Es una opinión personal. Lo importante es la justificación.
      - Si justifica acerca de los principios fundamentales siguen siendo vigentes dar 3 puntos. Si además añade que a pesar de lo anterior el software va cambiando constantemente y hay que tener en cuenta dar 1 punto. (4 puntos)
      - Si justifica que el software va cambiando constantemente y por lo tanto no sigue siendo vigente dar 3 puntos. Si además añade que a pesar de lo anterior los principios del libro se pueden seguir aplicando, dar 1 punto. (4 puntos).
   - **NO** se da el punto base si es que:
      - No cita la fuente. Algunas fuentes posibles: clases, libro clean code, página de clean code (podrían haber más si lo investigan).
      - Escribe más de 5 líneas, por no leer las instrucciones.

c) ¿Por qué hay posturas que dicen que testing asegura la calidad del código? Justifique.
*El testing **no** asegura la calidad del código, hay posturas que defienden lo contrario ya que el testing ayuda a validar y verificar y eso da una sensación de correctitud. Sin embargo, a pesar de lo anterior, los tests son insuficientes para asegurar la calidad.*
Se considera el puntaje completo si:
   - Menciona que el testing no asegura la calidad (2 puntos). → Era una pregunta con trampa, probablemente muchos justificaron la postura erróneamente.
   - Menciona alguno de los atributos del testing para su justificación: validación, verificación, correctitud, etc.- (2 puntos).

- **NO** se da el punto base si es que:
    - No cita la fuente. Algunas fuentes posibles: clases, lecturas (podrían haber más si lo investigan).
    - Escribe más de 5 líneas, por no leer las instrucciones.

Pregunta 2 [15 puntos totales]

Refactoriza el siguiente código para hacerlo más mantenible y para disminuir el código repetido, mejorando también las buenas prácticas. Tu respuesta debe contener:

1. Líneas o funciones donde hiciste los cambios. Debes explicar el por qué de los cambios.
2. Debes subir tu código refactorizado funcional, en tipografía Courier new tamaño 8.

[Código con comentarios adjunto]

Se considera el puntaje completo si **NO**:

- Mantiene el comportamiento duplicado, desaprovechando clases (-5 puntos)
- Mantiene código largo y no refactoriza para aplicar buenas prácticas (-5 puntos)
- Cambia el comportamiento original y añade funcionalidades nuevas que afecten el flujo (-5 puntos)

Pregunta 3 [15 puntos totales]

Una clase `Notificator` necesita enviar mensajes de un evento (`Event`) cada una hora a distintos tipos de usuario. Esta información base es la misma, pero varía lo que se decide mostrar o lo que se calcula dependiendo del tipo de usuario. Hay 3 tipos de usuarios de momento: `manager`, `officer` y `supervisor`. Los 3 usuarios reciben el cuerpo del evento que contiene un indicador decimal que va del -10 al 10. Las diferencias en la notificación son que el supervisor recibe la lista de id's de usuarios `officer` que recibirán la notificación y el `manager` recibe el valor máximo y mínimo de los indicadores de eventos enviados de los últimos 100 eventos.

Utilizando algún patrón de diseño, implemente **solamente** la lógica de la notificación a los usuarios en base al patrón escogido. Puede usar un diagrama de clases o escribir el código. Si decide usar un diagrama, cualquier información faltante le restará puntaje. Si escribe código, puede abstraerse de algunos cálculos a través de funciones no implementadas. Finalmente, justifique el por qué del patrón utilizado.

[Código con comentarios adjunto]

Se considera el puntaje completo si:

- Implementa 2 patrones acorde al formato de la notificación respecto al tipo de usuario y formato (podrían justificarse: Adapter, Decorator, u otros bien justificados) y además la gestión de notificaciones y eventos (uso de patrón Observer, Mediator, u otros bien justificados).

- Implementa 1 solo patrón que incluya tanto el formato de la notificación como la gestión de notificaciones y eventos.
- **De acuerdo a lo dicho durante la interrogación** se considerará el puntaje completo si solo se focalizó en implementar el cuerpo de la notificación y los usuarios (usando Adapter, Decorator, o similar). Por el contrario, **NO** se considera el puntaje completo, si **solo** implementa la gestión de notificaciones y eventos, En tal caso será la mitad del puntaje (7.5 puntos).

Pregunta 4 [15 puntos totales]

Identifique el o los code smells presentes en el siguiente código. **Con un solo code smell basta**. Debe señalar, **explicar el por qué** de este code smell **y las implicancias** que podrían significar.

Contexto: Google es un buscador web que además ofrece servicios de correo como Gmail. Facebook es una red social. La aplicación actualmente permite login con Facebook y Google, además de compartir en redes sociales algunos avances de la aplicación. También realiza búsquedas a través de google con el buscador público de la aplicación.

```
public class Client
{
    protected void Login() { ... }
    protected void NewPost() { ... }
    protected void Search() { ... }
    protected void AddFriend() { ... }
}

public class Google : Client
{
    ...
}

public class Facebook : Client
{
    ...
}
```

El code smell más evidente es Refused Bequest, porque en el uso de la herencia del cliente, según el contexto dado, las funcionalidades no se comparten entre si. Según el contexto, lo único en común que tienen es Login(), haciendo que el resto de las funciones no se justifiquen bien. Por ejemplo el Search() en el contexto de buscar con el buscador público no hará sentido en el código de la clase padre. Este tipo de code smell indican o podrían implicar una mala utilización de los objetos (herencia) y además falta de diseño inicial.
Se considera el puntaje completo si:

- Elige el code smell acorde. Si elige otro bien justificado se considera correcto (5 puntos).
- Explica cómo se identifica en el código ese code smell (5 puntos).

- Explica la implicancia de este code smell (5 puntos).