Objetivos

1. Aprender a aplicar un patrón estructural a un contexto de diseño específico.
2. Comprender los detalles del patrón "Composición" en su versión dinámica.

Contexto del problema

Suponga que se va a usar el framework EVAH para desarrollar una aplicación similar a SINVAS UML, específicamente un editor de diagramas de clases como los usados en este curso.

Planteamiento del problema

La edición de diagramas y modelos de clases implica que tanto un solo nodo o arista UML, así como grupos de nodos o aristas UML seleccionados dinámicamente (es decir, por el usuario durante una sesión), son susceptibles de operaciones tales como: "seleccionar", "cortar", "pegar", "eliminar", de un diagrama o paquete, así como "aumentar" (incrementar visualmente su tamaño), "reducir" y "dibujar" (o graficar). Siendo operaciones que se pueden aplicar tanto a elementos individuales como a grupos, es evidentemente este es un contexto adecuado para aplicar el patrón "Composición" ("Composite"). ¿Qué principio SOLID estaría implícito en la aplicación de este patrón a este contexto?

Descripción de la solución

Reutilice el modelo de clases de EVAH extendido en el laboratorio sobre el Decorador mediante el diseño de las clases Grafo, Nodo, *Arista*, GrafoUML, NodoUML, AristaUML así como los diferentes tipos de aristas UML y con base en ese modelo agregue:

1. Una clase para representar un diagrama de clases UML.
2. Una clase para representar un paquete UML. Un paquete puede contener clases y diagramas.
3. Un atributo en todas las clases que relacione a una instancia con la figura correspondiente al ícono, tanto para clases, paquetes como los diferentes tipos de relaciones UML.
4. Las clases que haga falta para la aplicación del patrón Composición en este contexto.
5. Variables de clase (nombre, tipo y multiplicidad si aplica) para todas las clases anteriores.
6 Firmas de métodos (nombre, tipo a retornar, tipos de los parámetros) para todas las clases anteriores.

Parte #1: haga lo anterior usando SOLAMENTE herencia, genere un modelo de clases.
Parte #2: haga lo anterior usando alguna plantilla, aplique el "Curiously Recurring Template Pattern" de Nesteruk, genere un modelo de clases.

En ambos casos agregue código en el main() para probar las clases diseñadas. Finalmente incluya en su modelo un comentario indicando cuál principio SOLID se está aplicando implícitamente y expliqué brevemente por qué se aplica.

Entrega y evaluación:

Parte #1 ... código y modelo generado: 50%
Parte #2 ... código y modelo generado: 50%

Fecha de entrega: domingo 14 de abril a las 23:59 por mediacionvirtual.