Trabajo Práctico: Composite

Programación Orientada a Objetos II
Comisión 2
2º Cuatrimestre de 2024

Nicolás Salvanescki

Ejercicio 1 - Desde el libro

1. Según el libro de Gamma et al. El patrón Composite logra esto creando una clase abstracta que represente a las demás y que incluya los mensajes que comparten entre todas ellas. También, tiene que incluir métodos para agregar, eliminar y obtener las subclases de las cuales se compone la actual. Esto es similar a las estructuras de datos como árboles y gráfos.
2. Composite se aplicaría en los siguientes dos casos:
   - Para representar una jerarquía de partes-todo, es decir un gran todo constituído por partes que se componen entre sí para formarlo.
   - Para que los clientes puedan ignorar las diferencias entre composiciones y objetos individuales (hoja) y puedan tratarlos de la misma forma.
3. Que el cliente no pueda ni quiera diferenciar entre objetos individuales y compuestos es gracias a que la POO permite abstracciones (clases abstractas e interfaces) y polimorfismo, ya que todas las clases deben ser capaces de responder a los mismos mensajes de manera diferente para ser tratadas de igual manera (abstracta). Siendo más específico, también es importante, aunque no indispensable, que puedan usarse expresiones lambda para poder propagar el mensaje de manera más directa y eficiente a las subclases (nodos hijos) de la actual. Los conceptos de programación funcional, como las high order functions (lambda) son muy buenas cuando se trata de representaciones recursivas, ya que reducen el tiempo de ejecución de los algoritmos de exponenciales a logarítmicos.

Ejercicio 2 - Cultivos

En el diseño no genera impacto, creo que es un detalle más de la implementación que del diseño. Al responder siempre el número de ganancia anual, la soja y el trigo no necesitan ser varias instancias de sus clases, con una sola alcanza. Las parcelas mixtas, por otro lado, pueden varias unas a otras, por lo que si se requiere de instancias varias.

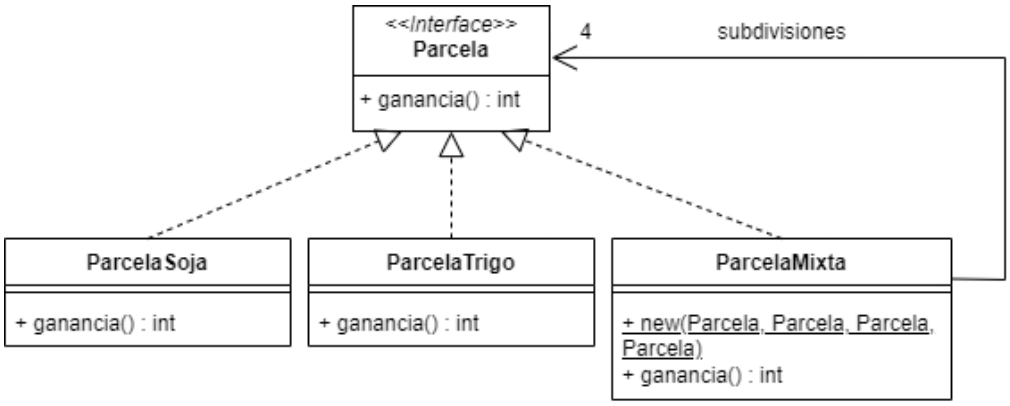

Figure 1: Diagrama de clases UML del ejercicio 2

4. Las operaciones de agregado y borrado de hijos las tienen que implementar las instancias de la clase ParcelaMixta, aunque Parcela debe entender los mensajes. En mi implementación en particular, no hay mensajes de agregar o borrar, en la instanciacion se pasan por parámetro directamente las subparcelas.

5. Parcela es **Component**, ParcelaSoja y ParcelaTrigo son **Leaves** (hojas) y ParcelaMixta es **Composite**.

Ejercicio 3 - Juego de estrategia

1. Modelo de la solución:

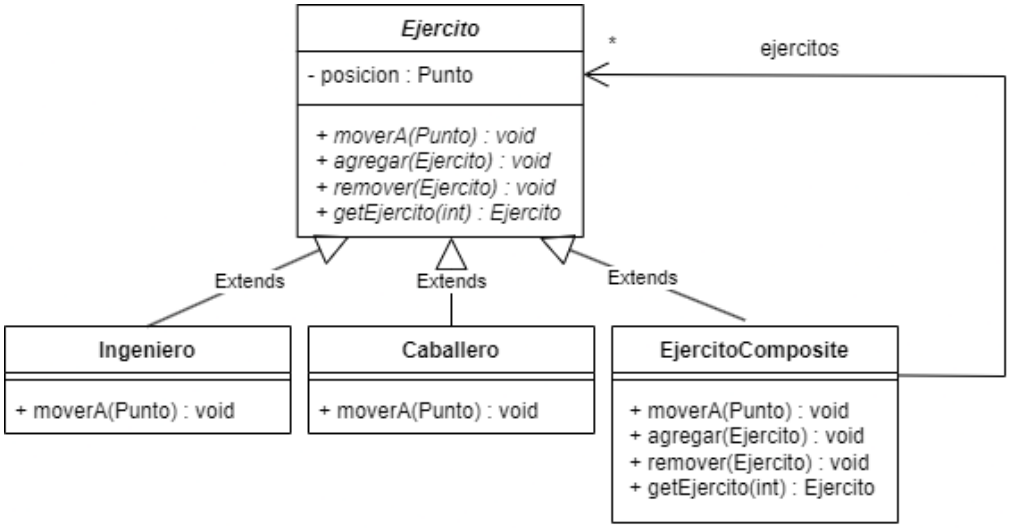

Figure 2: Diagrama de clases UML del ejercicio 3

3. Ejercito es **Component**, Ingeniero y Caballero son **Leaves** (hojas) y EjercitoComposite es **Composite**.

Ejercicio 6 - ShapeShifter

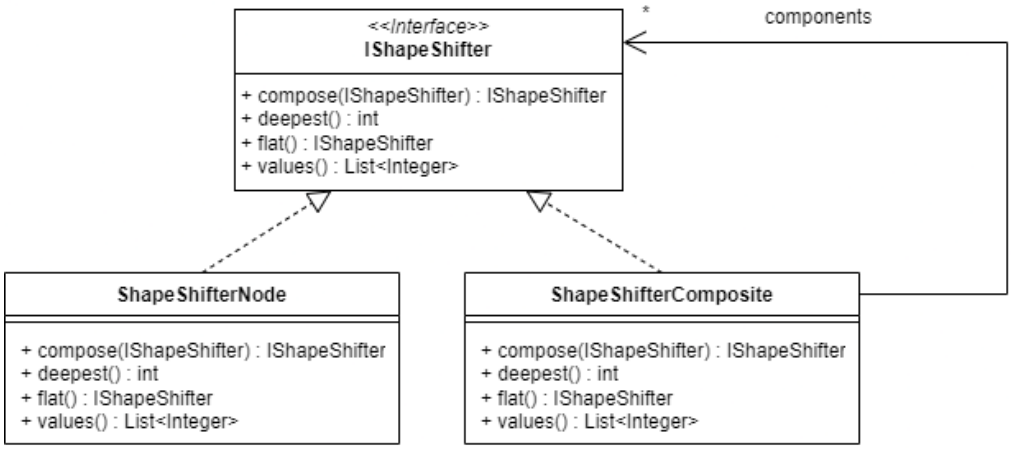

Figure 3: Diagrama de clases UML del ejercicio 6