**Documentación**

**“*Jerarquías*”**

**Programación Avanzada (2013)**

**Conclusiones extraídas en la implementación de PilaHL/PilaCL y ColaHL/ColaCL**

Estos ejercicios sirvieron para marcar la diferencia entre entre composición de clases (relación *has-a*) y herencia de clases (relación *is-a*). Es decir, mediante composición de clases se establece una relación entre clases donde una clase es parte de o contiene a otra. Este es el caso, en la vida real, de un Auto y un Motor, por citar un ejemplo. Por otro lado,  en la herencia de clase se establece una relación entre clases donde una es una especialización o una generalización de la otra (dependiendo de que lado de la relación se lo mire).

En nuestro caso, al utilizar la relación de composición entre Lista y PilaCL/ColaCL, la implementación se redujo a traducir los métodos requeridos por la interfaz Pila/Cola correspondiente. Mediante manipulaciones internas de la lista contenida dentro de nuestra clase PilaCL/ColaCL, se obtienen métodos concisos y fáciles de seguir a primera vista. Para nosotros implicó menos código porque el comportamiento de PilaCL/ColaCL está limitado por la interfaz correspondiente, y existen métodos de sobra en la Lista como para respetar el comportamiento de una u otra.

Por otro lado, usando la relación de herencia entre Lista y PilaHL/ColaHL, la implementación se traduce en no solamente utilizar la estructura interna de nodos de Lista para respetar el contrato de la interfaz Pila/Cola correspondiente, sino a limitar el uso indebido de ciertos métodos que son exclusivos de Lista. Lo cual se traduce en mucho más código que las implementaciones anteriores y, por ende, mayor probabilidad de error al implementar algo sencillo como es el comportamiento de una Pila o una Cola. Por otro lado, tiene la ventaja de aportar polimorfismo, es decir, con PilaHL/ColaHL podríamos tener un método o una clase que posean una referencia a Lista, e intercambiar PilaHL/ColaHL en forma transparente, pero no tiene mucho sentido ya que son dos estructuras bien diferenciadas a la hora de elegir una u otra.

En conclusión, en nuestro caso, el uso de herencia estaría sumando trabajo por realizar y no simplificando las cosas, lo cual es propenso a errores y suma mayor cantidad de horas de trabajo. En nuestra opinión Pila/Cola no son una Lista, y por ende, no tiene demasiado sentido utilizar herencia; es mucho más entendible que una Pila/Cole use una Lista o un Array (y eso queda expresado mejor por la relación de composición).

Quizá si existiese una clase/interfaz de mayor jerarquía como lo es en Java, la interfaz Collections, se podría aprovechar las ventajas de ambos ejercicios (el polimorfismo y el código sencillo y al punto). Pero no es nuestro caso.