Práctica 4, Factory, Abstract Factory y Builder

Rojas Reyes Saúl Adrián.

114006224.

1. Menciona los principios de diseño esenciales de los patrones Factory, Abstract Factory y Builder. Menciona una desventaja de cada patrón.

El patrón Factory se basa en crear una clase abstracta, o interfaz, a conveniencia, que defina la creación de de uno objeto o más, digamos el método fabrica, todas las clases que extiendan a esta sobrecargan al método fabrica, cada una de decidirá cómo y qué clase de objeto instanciar al momento de llamar al método fabrica. De esta manera se darán los objetos necesarios en cada momento usando una misma estructura de creación delegada a las subclases. Su desventaja es que solo funciona en sistemas jerárquicos, además de que se puede complicar el código con la gran cantidad de clases que acompañan al patrón.

El patrón Abstract Factory guarda la misma idea que Factory pero ampliada y flexible, se basa en crear una clase abstracta o interfaz, que defina la creación de uno o más objetos, ahora cada clase que implemente o extienda de esta creara una familia de objetos, de esta manera todos los objetos en una misma familia son compatibles entre sí. Así podemos no especificar que clase concreta pero que cumplan con una funcionalidad. Su desventaja es que la cantidad de interfaces y objetos creados es mucho más alta, más que factory, por lo que el código se puede complicar mucho.

El patrón Builder se basa en crear objetos complejos paso a paso, así cambiando las propiedades de cada objeto de la misma clase a nuestra conveniencia. Se crea un constructor, digamos Builder, al cual se le delega el armado de este objeto complejo, el Builder tiene métodos para ir construyendo paso a paso este objeto, el objeto complejo tiene su constructor privado para evitar la instanciación de otra manera que no sea el Builder, el builder es dirigido por un director, el director es el que interactúa directamente con el cliente para determinar qué debe tener el objeto. Su desventaja es que el código es bastante complejo y extenso, además de que los objetos creados suelen ser finales, por que puede crear inflexibilidad en algunos casos, no en su creación pero si en su uso.

Sobre la práctica. Esta se maneja, como fue solicitado, de manera interactiva desde la consola. Las instrucciones se imprimen en pantalla en todo momento por lo que el cliente siempre sabrá qué hacer. Se puede salir de la simulación cuando se solicite si el cliente lo desea. La práctica usa Builder para la construcción general y Factory para la creación de las partes, así la construcción del objeto completo Nave es delegado a builder y este ocupa las partes ya construidas que otorga Factory.

La práctica se ejecuta con la clase Practica4, la simulación se encuentra en su main, yo usé el método javac Practica4.java para compilarla, luego java Practica4, debería funcionar correctamente.