Modelado y Programación

Práctica 4: Implementación de los patrones de diseño Factory, Abstract Factory ó Builder.

25 de noviembre de 2019

Integrantes:

- Hernández Aguilar Luis Alberto, 314208682
- Pérez Hernández Antonio, 418003499

Uso:

- Compilamos la práctica y generamos un archivo .jar con:
  $ ant practica4.jar
- Ejecutamos con:
  $ java -jar practica4.jar

PARTE TEÓRICA

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

   FACTORY:
   Sirve para crear una jerarquía de clases, permitiendonos la creación de un sub-tipo determinado, ocultando así los edtalles de creación del objeto. Permite delegar la responsabilidad de la creación de los objetos, para que no sea el programados el que decida qué clase instanciará.
   Una de las desventajas del patrón Factory es que se obliga al cliente a definir subclases de la clase *Creador* sólo para crear un producto concreto y esto puede no ser apropiado siempre.

   ABSTRACT FACTORY:
   Busca agrupar un conjunto de clases que tienen un funcionamiento en común llamado *jerarquía*, las cuales son creadas mediante un **Factory**. Este patrón es especialmente útil cuando requerimos tener ciertas jerarquías de clases para resolver un problema, por lo que podríamos decir en base al patrón Factory que Abstract Factory es una fábrica de fábricas.
   Una de las desventajas del patrón Abstract Factory es que es difícil dar cabida a nuevos tipos de productos. Ampliar las fábricas abstractas para producir nuevos tipos de productos no es fácil. Esto se debe a que la interfaz *AbstractFactory* dija el conjunto de productos que se pueden usar. Permitir nuevos tipos de productos requiere ampliar la interfaz de la fábrica, lo que a su vez implica cambiar la fábrica abstracta y todas sus subclases.

   BUILDER:
   Nos permite crear objetos que habitualmente son complejos utilizando otro objeto más simple que los construye paso por paso. El objeto fuente se compone de una variedad de partes que contribuyen individualmente a la creación de cada objeto a través de un conjunto de llamadas interfaces comúnes de la clase *Abstract Builder*.
   Una de las desventajas del patrón Builder es que hay que crear un BuilderConcreto para cada representación de un producto, lo que puede acabar con multitud de clases.

Bibliografía.

- Welicki, León. (2008). *Patrones de Fabricación: Fábricas de Objetos.* 2019, de Microsoft, en `https://docs.microsoft.com/es-es/previous-versions/bb972258(v=msdn.10)?redirectedfrom=MSDN`
- ytapanes. (2018). *Patrones de Diseño (III) - Abstract Factory.* 2019, de Webtech, en `https://webtech.cubava.cu/2018/02/08/patrones-de-diseno-iii-abstract-factory/`
- Capdevila, Albert. (2019). *PATRÓN BUILDER EN C#.NET* 2019, de albertcapdevila.net, en `https://albertcapdevila.net/patron-builder-csharp-net/#consecuencias`