Facultad de Ciencias, UNAM
Modelado y Programación
Práctica 3

Hernández Morales José Ángel
No. de cuenta: 315137903
Rubí Rojas Tania Michelle
No. de cuenta: 315121719

16 de agosto de 2020

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

   Solución:

   a) Factory

   Este patrón nos permite la creación de un subtipo determinado por medio de una clase de Factoría, la cual oculta los detalles de creación del objeto.

   El objeto real creado es enmascarado detrás de una interfaz común entre todos los objetos que pueden ser creados, con la finalidad de que éstos puedan variar sin afectar la forma en que el cliente interactúa con ellos.

   Es normal que un Factory pueda crear varios subtipos de una determinada interfaz y que todos los objetos concretos fabricados hagan una tarea similar pero con detalles de implementación diferentes.

   La intención del *Factory Method* es tener una clase a la cual delegar la responsabilidad de la creación de los objetos, para que no sea el mismo programador el que decida qué clase instanciará, si no que delegará esta responsabilidad al Factory confiando en que este le regresará la clase adecuada para trabajar.

   Una desventaja de este patrón es que hace que el código sea más difícil de leer, ya que todo su código está detrás de una abstracción que a su vez puede ocultar las abstracciones. Otra desventaja es que, al proporcionar sólo métodos de fábrica estáticos, las clases sin constructores públicos o protegidos no pueden ser subclasificadas.

   b) Abstract Factory

   Este patrón nos permite crear, mediante una interfaz, conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cual es el objeto concreto.

   Abstract Factory puede ser usado cuando queremos devolver una de varias clases de objetos relacionados, cada uno de los cuales puede devolver varios objetos diferentes a petición. En otras palabras, regresa uno de varios grupos de clases. Además, puede ser usado cuando un sistema debe ser independiente de cómo sus objetos son creados, cuando un sistema debe ser *configurado* con una cierta familia de productos; o cuando se necesita reforzar la noción de dependencia mutua entre ciertos objetos.

Una desventaja de este patrón es que para agregar nuevos objetos se deben modificar tanto las fábricas abstractas como las concretas. Otra desventaja es que la decisión sobre qué fábrica usar se toma en tiempo de ejecución.

c) Builder

Este patrón permite crear objetos que habitualmente son complejos, utilizando otro objeto más simple que los construye paso por paso.

Se utiliza en situaciones en las que debe construirse un objeto repetidas veces o cuando este objeto tiene gran cantidad de atributos y objetos asociados, y en donde usar constructores para crear el objeto no es una solución cómoda.

Además, se trata de un patrón que es bastante útil en la ejecución de test (unit test, por ejemplo) en donde debemos crear el objeto con atributos válidos o por defecto.

Una desventaja de este patrón es que requiere que las clases de constructores sean mutables. Otra desventaja es la necesidad de mantener la duplicidad de los atributos que deben estar en la clase destino y en el Builder.

2. Justificación del patrón

   Utilizamos el patrón *Abstract Factory* porque nos permitió crear una fábrica para cada componente de los autos: llantas, motor, carrocería, blindaje y armas. Además, cada fábrica se encarga de fabricar cada producto del mismo tipo pero con características distintas entre ellos.

3. Funcionalidad del programa

   a) Primero, debemos posicionarnos dentro de la carpeta **src**. Aquí se encuentran todas las clases de nuestra práctica.

   b) Compilamos todas nuestras clases.

   ```
   $ javac *.java
   ```

   c) Ejecutamos nuestro programa principal.

   ```
   $ java Main
   ```

Referencias

[1] https://informaticapc.com/patrones-de-diseno/factory-method.php

[2] https://www.oscarblancarteblog.com/2014/07/18/patron-de-diseno-factory/

[3] https://www.quora.com/What-are-the-pros-and-cons-of-the-factory-design-pattern?share=1

[4] https://datacadamia.com/code/design_pattern/factory#disadvantage

[5] https://lineadecodigo.com/patrones/patron-abstract-factory/

[6] https://www.c-sharpcorner.com/article/abstract-factory-design-pattern-in-ado-net-2-0/

[7] https://www.dineshonjava.com/abstract-factory-design-pattern/

[8] https://experto.dev/patron-de-diseno-builder-en-java/

[9] https://en.wikipedia.org/wiki/Builder_pattern#Disadvantages