Práctica 3

Raúl Nuño Valdés, 317204580

26 de noviembre de 2021

Parte teórica

1. **Menciona los principios de diseño esenciales del patrón Factory**

Se tienen clases que heredan de una que tiene el esqueleto de como crear un objeto (las llamaremos fabricas) Cada una de estas clases es distinta en el objeto que crean y como lo crean.

También hay clases que heredan de un objeto (las llamaremos productos) que si bien tienen los mismos métodos, sus implementaciones son distintas.

Evita que tengas que usar el operador `new`, solo debes usar la fábrica concreta que quieras `Producto = FabricaConcreta.crearProducto`

2. **Menciona una desventaja de Factory**

Se van a necesitar una gran cantidad de subclases. Para cada producto nuevo, una fabrica nueva.

1. **Menciona los principios de diseño esenciales del patrón Abstract Factory**

Se crea una clase abstracta (fábrica abstracta) que puede construir ciertos objetos (productos abstractos), y hay clases (fábricas concretas) que heredan de la fábrica abstracta. Las fábricas concretas crean objetos concretos que heredan de los productos abstractos.

Ahora, un cliente debe tomar cierta fábrica para hacer lo que tenga que hacer con ella.

2. **Menciona una desventaja de Abstract Factory**

Este patrón hace que tengas una gran cantidad de subclases. Cada vez que quieras agregar un producto nuevo, requieres también de una fábrica nueva.

Hago nota que Factory y Abstract Factoy se parecen bastante para mí, pero la diferencia según entiendo es que Factory llama a la fábrica concreta y esta crea un objeto concreto así, mientras que Abstract Factory necesita de un cliente que tenga como atributo a la fábrica concreta y ya pueda crear el objeto concreto y es el mismo cliente quien lo manipula.

1. **Menciona los principios de diseño esenciales del patrón Builder**

Builder tiene una o varias clases que heredan de un constructor que es abstracto. Cada uno de estos constructores concretos crean un objeto que tienen los mismos parámetros, pero distintas en funcionalidades, además, los objetos que crean no necesariamente heredan de una misma súper clase.

Además se requiere de un director que va a recibir a uno de estos constructores como parámetro de sus métodos. Estos métodos crean un objeto con ciertos parámetros ya definidos, pero usando al constructor concreto que se le asignó.

2. **Menciona una desventaja de Builder**

Al igual que los otros, crece demasiado. Aunque cada producto totalmente nuevo que se necesite va a tener un constructor nuevo también, si se quiere un producto que solo cambie en cuanto a los parámetros, sólo habrá que definir un nuevo método en el director.

Elección de Builder

Usé Builder porque al final sólo se trataba de construir un coche que tenía parámetros distintos, no una cosa nueva (como un avión u otro medio de transporte).

Además, de haber usado Factory o Abstract factory, es probable que se hubiera requerido crear una fábrica nueva para cada carro en con una combinación de componentes distinta (en este caso, 4 clases nuevas ya que necesitábamos

3 carros con un modelo prediseñado y uno libre). En cambio, usando builder tenemos un mismo creador, y donde cambia es en el director que es 1 clase con 4 métodos.

Notas para ejecutar

Si se ejecuta en CMD de Windows, es probable que aparezcan caracteres ANSI. Es porque se usaron colores para imprimir la información.

Si quiere ver los colores, use

```
reg add HKEY_CURRENT_USER\Console /v VirtualTerminalLevel
/t REG_DWORD /d 0x00000001 /f
```

en el CMD como administrador.

Pra revertilo, use

```
reg add HKEY_CURRENT_USER\Console /v VirtualTerminalLevel
/t REG_DWORD /d 0x00000000 /f
```

en el CMD como administrador.

De esta pagina se obtuvo la información:

https://www.codeproject.com/Tips/5255355/How-to-Put-Color-on-Windows-Console

Ahora, para compilar se usa `javac -d bin -cp src src/main/Main.java` desde la carpeta `./proyecto`

Para ejecutar se usa `java -cp bin main/Main` también desde la carpeta `./proyecto`