UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Modelado y programación.

Rosa Victoria Villa Padilla

[2025-1] (22/09/2024)

Práctica 3:

Builder, Adapter y Abstract Factory

Equipo: Christian.

- Leon Navarrete Adam Edmundo
- Rubio Resendiz Marco Antonio
- Valencia Pérez Guillermo Emanuel

Ciudad Universitaria, Cd. Mx. 2024

Anotaciones sobre la estructura de la práctica.

La práctica consistió en desarrollar una aplicación de ventas de computadoras de nombre “Los amantes de Intel” mediante el uso de tres patrones: Builder, Adapter y Factory (Method o Abstract).

Elección de patrones de diseño.

Abstract Factory.

Elegimos el patrón de diseño Abstract Factory para todo lo relacionado a los componentes de las computadoras, el objetivo fue plantear una interfaz ComponenteFactory con distintas “fabricas” de componentes según la gama de la computadora (Gamer, Standard, Baja), planteando una familia de clases para cada tipo de componente. Donde cada categoría de componente es una interfaz y los productos son clases concretas.

Los componentes implementados fueron los siguientes:

1. Procesadores.
    a. Intel Core i9
    b. Intel Core i5
    c. Intel Core i3
2. Tarjetas gráficas.
    a. NVIDIA RTX 3090
    b. NVIDIA RTX 4070Ti
    c. NVIDIA GTX 1030
3. Memorias RAM.
    a. Corsair Vengeance
    b. Kinston Fury
    c. Adata XPG
4. Discos duros.
    a. Kingston A400
    b. Samsung 870EVO
    c. Seagate PORTR1
5. Tarjetas madre.
    a. Gigabyte ATX Z790
    b. MSI Pro Micro B760
    c. ASUS Micro-ATX H470
6. Fuentes de poder.
    a. Cosair CX750
    b. MSI Mag650
    c. Cooler Master Gold850

Adapter.

Empleamos el patrón de diseño Adapter para el caso de querer ensamblar una computadora con piezas de la marca AMD (CPU's y GPU's). El planteamiento general es que estas piezas tienen esencialmente los mismos métodos que los componentes definidos en factory pero se encuentran definidos en inglés y la estructura de la clase es un tanto distinta.

Para cumplir con la posibilidad de ensamblar una computadora con este tipo de piezas se emplea una clase AdaptadorCPU y AdaptadorGPU que reciben mediante el constructor el componente que deseamos adaptar (en este caso solo se cuenta con un producto concreto). Y es a partir de esta instancia que se sobrecargan ciertos métodos para que el Adaptador implemente la interfaz del componente sin que modifiquemos los productos concretos.

Los componentes de AMD son:

1. Procesadores.
    a. AMD Ryzen7
2. Tarjetas gráficas.
    a. AMD Radeon RX7600

Builder.

Usamos el patrón de diseño Builder para todo lo relacionado a el ensamble de computadoras. Para ello nos basamos en una clase PC que cuenta con todos los atributos acorde a las interfaces de componentes definidas en factory. Y contamos con una interfaz Builder y cuatro clases que la implementan para definir un ensamble diferente acorde a cada implementación de la clase Builder. Para centrarlizar el armado de la computadora se utiliza una clase Director que cuenta con una instancia de Builder y un método de armado que es el mismo para todas las computadoras, así como un método para obtener la instancia de PC que ha sido modificada.

Además, todas las implementaciones de Builder (Excepto PCPersonalizableBuilder) cuentan con una instancia de ComponenteFactory para definir un modelo predeterminado de computadora.

Las clases de Builder son las siguientes:

- PCBajaBuilder: Para computadoras de gama baja, utiliza una instancia de FactoryComponenteBajo.
- PCStandarBuilder: Para computadoras de gama media, utiliza una instancia de FactoryComponenteStandar
- PCGamerBuilder: Para computadoras de gama alta, utiliza una instancia de FactoryComponenteGamer.
- PCPersonalizableBuilder: Para darle al usuario la posibilidad de escoger los componentes que desee.

Nota: La clase PCPersonalizableBuilder no emplea ninguna instancia de ComponenteFactory debido a una consulta que hicimos con el ayudante. El diseño original era emplear una lista de fábricas para iterarla según fuera el componente que tuviera que elegir el usuario, pero al comentar esta idea con los ayudantes nos aconsejaron que no hiciéramos eso y mejor emplearamos una lista de componentes. Se llegó a la conclusión de que el uso de fábricas era innecesario si el punto era usar directamente los componentes dado una lista de ellos, por ello que el seteo de estos se hace mediante varias listas de componentes

Respecto al uso del proyecto.

El programa fue desarrollado en la versión de java 11.0.22 y se empleó linux (debian y ubuntu) durante su desarrollo. Para ejecutarlo en linux (para distribuciones basadas en debian) es necesario usar los siguientes comandos en terminal:

```
$ javac -d bin $(find src -name "*.java")
$ java -cp bin mx.unam.ciencias.modelado.practica3.Main
```