Modelado y programación 2022-1

Práctica 4: Implementación del patrón Factory, Abstract Factory o Builder

Bernal Márquez Erick. 317042522
Deloya Andrade Ana Valeria. 317277582

- ***Parte teórica:***

1. Menciona los principios de diseño esenciales de los patrones Factory, Abstract Factory y Builder.

   Factory: Define una interfaz para crear un objeto pero deja que las subclases decidan cual
   Los principios que sigue este patrón son:
   - ★ Principio de inversión de dependencia: Depende de las abstracciones, no de las clases concretas

   Abstract Factory: Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas
   Los principios que sigue este patrón son:
   - ★ Principio de inversión de dependencia: Depende de las abstracciones, no de las clases concretas

   Builder: Permite construir distintas representaciones de objetos complejos paso a paso usando constructores de objetos simples. La construcción de los objetos complejos dependerá de un director que tratará a cada componente (que debe estar modularizado) como una etapa de la construcción.
   Los principios que sigue este patrón son:
   - ★ Responsabilidad única

2. *Menciona una desventaja de cada patrón.*

   Factory:
   - ❖ Puede complicarse si tenemos varias subclases

   Abstract Factory:
   - ❖ Puede complicarse aún más si tenemos varias subclases y varios diseños

   Builder:
   - ❖ La complejidad puede aumentar puesto que hay que crear varias clases para un objeto

Justificación

El patrón *fábrica* definitivamente no es conveniente, pues hay que crear varios tipos de objetos de las partes de un auto, *fabrica* solo funcionaria si tuviéramos un tipo de componente Armas o Blindaje o Carrocería…, pero en este caso tenemos 5 tipos de componente.
Builder presenta un problema similar pero "al revés", es decir, podemos tener los 5 tipos de componentes pero no subtipos para crear el auto.
Es por esto que decidimos implementar *abstract factory* porque nos permite crear distintos tipos, así como subtipos de los mismos.

- ***Sobre la práctica:***
  No se necesita de algún software adicional para el funcionamiento de la práctica. Puede ser fácilmente compilada con:

  ```
  $javac *.java
  ```

  Y ejecutada con

  ```
  $java Main.java
  ```

  El menú te da la opción de crear tu propio auto o elegir uno de 3 tipos predefinidos.
  En caso de querer crear tu propio auto se pide que elijas el tipo de cada parte del auto, es decir, el tipo de arma, blindaje, carrocería, motor y llantas.