UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica 0 : Repaso de POO y Java

ALUMNO

Javier Alejandro Rivera Zavala - `311288876`

PROFESOR TITULAR

Gilde Valeria Rodríguez Jiménez

PROFESORES ADJUNTOS

Rogelio Alcantar Arenas
Gibran Aguilar Zuñiga
Luis Ángel Leyva Castillo

ASIGNATURA

Computación concurrente

07 de Febrero del 2024

Patrones de diseño elegidos y justificación

Los patrones de diseño empleados fueron Decorator y Strategy.

Decorator por un lado, nos permite añadir elementos a nuestros platillos de forma dinámica, dichos elementos pueden modificar el comportamiento de los objetos que modelan a los platillos en el menú. Por el momento, el método que nos brinda el precio total del platillo es alterado para que decore a los platillos, así mismo, el método que nos permite obtener el nombre del platillo. Al usar Decorator, evitamos tener que contemplar desde un inicio a los elementos adicionales dentro de la estructura del platillo mismo, además, los cambios sobre el comportamiento antes mencionados, se acumulan para así brindarnos un comportamiento general más complejo.

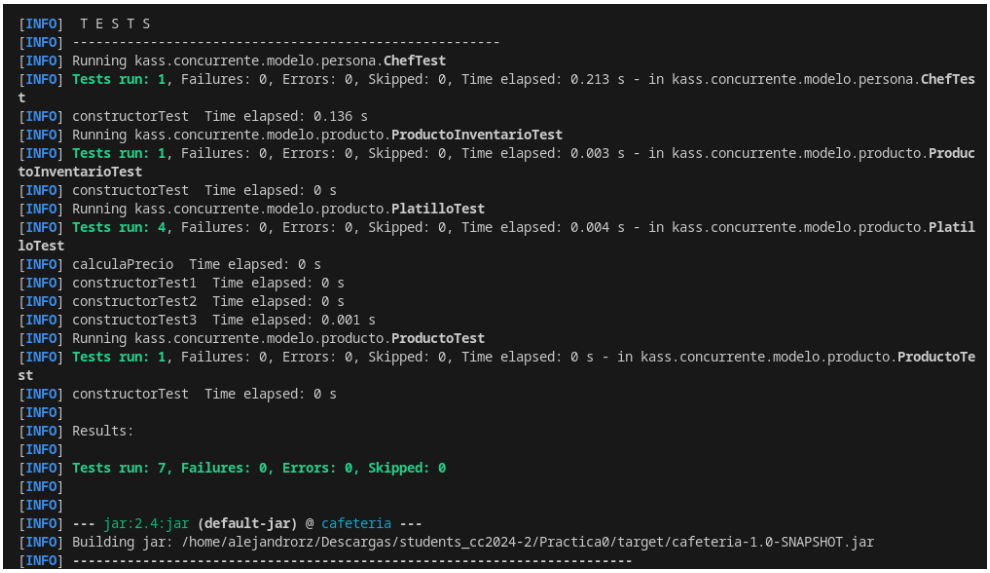
He de hacer notar que la estructura requerida para los test, impedía definir clases abstractas coherentes con el resto del diseño, por lo tanto, mi implementación de Decorator varía un poco respecto del estándar para dicho patrón. Así mismo, la exigencia de construir los platillos como listas de productos, también supuso hacer algunos ajustes. Otro pequeño inconveniente lo encontré a la hora de controlar los accesos a los métodos, pues los test obligaban a que dicho acceso fuera público, esto para diversos métodos importantes en el esquema general.

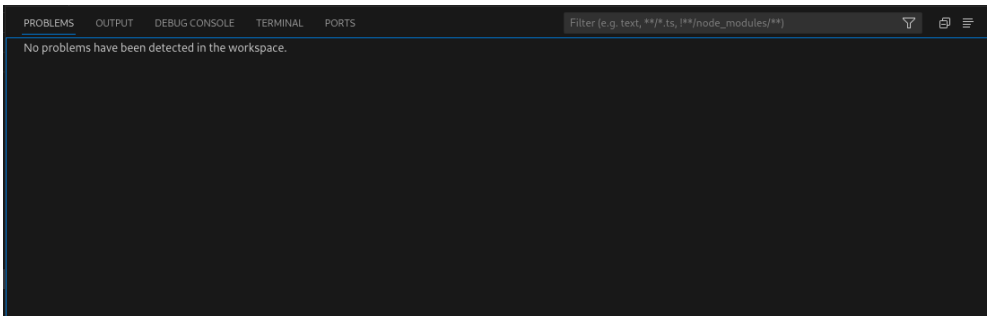
En el caso del patrón Strategy, este nos permite ajustar la velocidad con la que "cocinamos" nuestros platillos, según se vaya necesitando durante la labor de nuestro chef. Un platillo sencillo quizás no requiere de apresurar su preparación, sin embargo, un platillo más elaborado puede requerir de una optimización respecto a dicho tiempo, para así mantener satisfechos a nuestros clientes. La idea es poder cambiar el comportamiento del Chef de forma dinámica, esto durante la ejecución de una tarea en concreto, por ello es que el patrón Strategy resulta una elección natural conforme lo expuesto en la práctica.

Oro punto a señalar, es que quizás por lo limitada que es la práctica, no se nota tanto el impacto que el uso de los anteriores patrones de diseño pudiera tener, pues los mismos ajustes podrían realizarse directamente sobre las clases que modelan objetos genéricos. A pesar de lo anterior, el uso de patrones nos permite asegurar una mayor escalabilidad del proyecto y asegurar que el mismo sea más fácil de mantener.

Diagrama UML, tests y errores de Sonar Lint

El diagrama UML es bastante grande, adjunto el archivo. Se presentan aquí capturas de la compilación y los resultados de los test, así como de los resultados en terminal arrojados por SonarLint.





Referencias

No hay referencias en esta ocasión pues todo se hizo a partir de los conocimientos que he adquirido a lo largo de la carrera.