Práctica 02: Equipo Tepaches

Luis Adrian Galindo Ruvalcaba 322200890
José Carlos Pinzón Chan 425122448 Enrique Velez 424099273

Modelado y programación 2026-1

September 13, 2025

1. Antecedentes

El objetivo de esta práctica es implementar adecuadamente los patrones State, Template y Decorator para resolver el problema propuesto, que en esta ocasión, consiste en programar un robot para la famosa franquicia de pizzerías "El Pequeño Cesarín".

2. Compilación y ejecución

Para compilar y ejecutar el proyecto, el usuario debe de tener cierto conocimiento previo utilizando la terminal en sistemas Linux. Además, es necesario contar con la versión 24 de Java (JDK 24).

Pasos para compilar y ejecutar el código.

I. Después de desempaquetar el archivo *.zip*, abrimos una terminal en la carpeta Practica02_Tepaches.

II. Introducimos el siguiente comando en la terminal, para movernos hasta la carpeta practica2:
cd src/myp/practica2/

III. Estando en la carpeta practica2, utilizamos el siguiente comando para compilar todos los archivos *.java* :
javac */*.java Practica2.java

IV. Para realizar la ejecución del programa copiamos en terminal:
java Practica2.java

Note que la clase que contiene el método main es la clase Practica2.

3. Uso de los patrones de diseño

Los patrones de diseño empleados en esta práctica fueron: State, Template y Decorator. A continuación se describe la implementación de cada uno.

- ▷ State: Este patrón se utilizó principalmente para modelar las distintas fases por las que pasa un Robot a lo largo de la ejecución, ya que en cada fase las opciones disponibles para el usuario varían (es decir, no todas se pueden llevar a cabo).

  En nuestra implementación, se definió la interfaz RobotState, que en general, contiene los mismos métodos que la clase Robot. Las distintas implementaciones de RobotState, como lo son SleepingState, OrderConfirmedState, etc... proporcionan una implementación distinta para cada método.

  La clase Robot tiene como atributos individuales a todos los estados (implementaciones de RobotState), sin embargo, el atributo *state* representa el estado actual del robot y define su comportamiento. De modo que conforme avanza la ejecución del programa, para modificar el comportamiento del robot (modificar las opciones del usuario)lo único que se hace es cambiar el *estado interno del robot.* (atributo *state*).

- ▷ Template: La razón detrás de utilizar template para la creación de las pizzas es bastante sencilla. Como cada pizza requiere de los mismos pasos para su elaboración con tal vez ligeros cambios en el proceso y en las cualidades de la pizza, entonces, podemos usar siempre este *algoritmo* variando sólamente ciertos pasos.

  Así, creamos una clase abstracta Pizza, con todos los atributos que consideramos necesarios. En ella definimos el método preparePizza() que es el algoritmo general que describe el proceso de preparar una pizza, y siempre es el mismo. En este algoritmo identificamos que los pasos: *agregar queso* y *agregar proteína* son los únicos que varían (que una pizza sea vegetariana cambia sólo en algunas pizzas). Por ejemplo las clases que heredan de Pizza, como MexicanPizza o SupremePizza, redefinen sólo sus métodos addCheese() y addProtein().

  Cabe destacar que aunque todas las clases que heredan de Pizza tienen los "mismos atributos", los valores de estos generalmente son distintos.

- ▷ Decorator: El uso de Decorator fue un poco más evidente que el resto, porque bueno, en clase vimos un ejemplo muy parecido. Lo que se buscó en este caso, fue envolver a los distintos helados con los ingredientes extra, para facilitar la descripción y el cálculo del precio final, en el ticket.

  Se creó una clase abtracta IceCream. Las clases que heredan de IceCream evidentemente son nuestros helados, no obstante la clase abtracta DecoratingIngredient también hereda de IceCream y es la que usaremos para nuestros decoradores. Los decoradores tienen un atributo *icecream* que les permite envolver a los helados y a otros decoradores. Sus métodos getCost() y getDescription() son de cierto modo "recursivos" y exclusivamente terminarán cuando se "alcance" una instancia de StrawberryIceCream, VanillaIceCream o ChocolateIceCream.

4. Funcionamiento del programa

La clase Practica2 contiene nuesto método main y dirige la ejecución del programa utilizando una instancia de Robot y sus métodos. Dependiendo de la opción seleccionada por el usuario, el Robot trata de realizar dicha acción, y puede (o no) cambiar de estado. Asimismo, el usuario es libre de detener la ejecución en cualquier momento.

5. Otros detalles

Cuando el robot se encuentra en el estado: “preparando pedido”, osea, una instancia de PreparingState, para terminar la preparación del pedido, se debe seleccionar la opción: “(6) Preparar orden”, nuevamente. Esta fue la manera que encontramos de “simular”, que al robot le toma un tiempo preparar el pedido. Otro aspecto importante es que si el usuario agrega una pizza o un helado y no lo confirma, si intenta agregar otra pizza (o helado) podrá hacerlo, pero sólo estará sobreescribiendo su pedido anterior. Por ejemplo si pidió una pizza de pepperoni y después pide una pizza Mexicana, cuando confirme su pedido, este sólo dirá que pidió una pizza Mexicana.

Para una mejor comprensión, puede consultar el diagrama de estados del robot.