Daniel Piñeros 202013147

TALLER 5 PATRONES - DPOO

- **Información general del proyecto, deben incluir la URL para consultar el proyecto.**
  URL del proyecto: https://github.com/smmehrab/coffee-vending-machine.git
  Patrón de diseño del proyecto: Patrón de comportamiento STATE

- **Información y estructura del fragmento del proyecto donde aparece el patrón. No se limiten únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente van a tener que incluir elementos cercanos que sirvan para contextualizar.**
  El patrón de diseño State se utiliza para permitir que un objeto cambie su comportamiento interno cuando su estado interno cambia. Se basa en la idea de representar los diferentes estados de un objeto como clases separadas que implementan una interfaz común. El objeto principal tiene una referencia a uno de estos estados y delega el comportamiento específico a la clase de estado actual.

  Como se puede observar en la siguiente imagen para el programa del repositorio se crearon 3 macro estados en los que se acopla TODO el funcionamiento de la máquina dependiendo del estado en el que se encuentre.

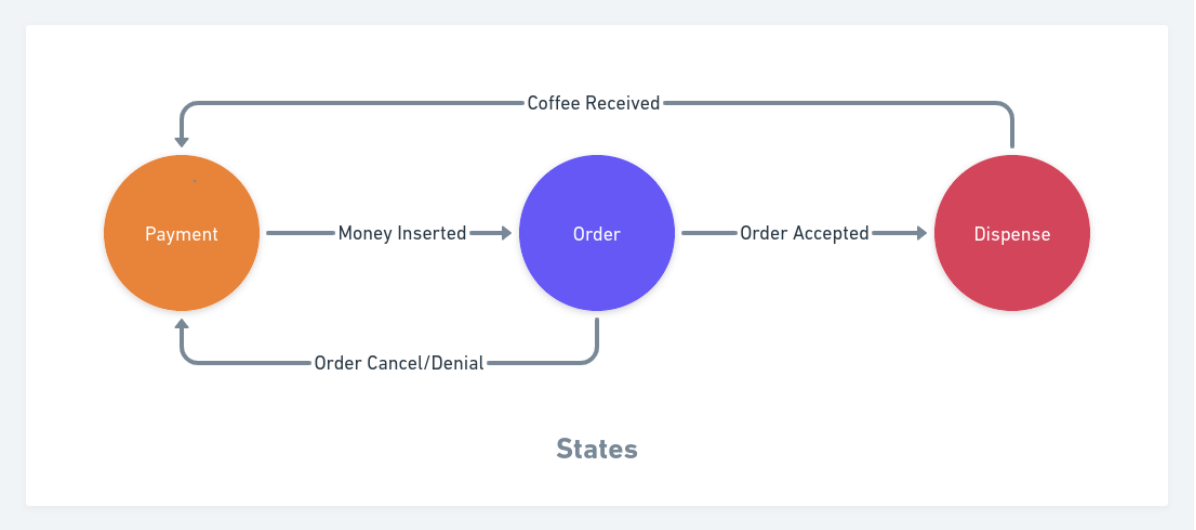
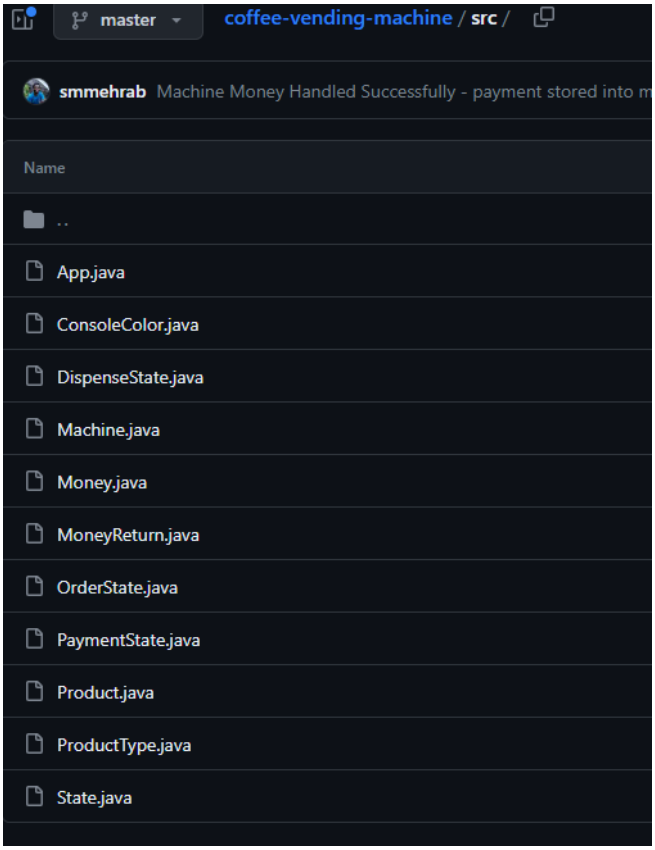


Figura 1. Mapa de estados en máquina de café

- **Información general sobre el patrón: qué patrón es y para qué se usa usualmente**
  El patrón de diseño “STATE” se caracteriza por crear estados para un mismo objeto dentro del programa, por lo cual lo que tradicionalmente (sin usar patrones de diseño/comportamiento) se haría con una cantidad abrumadora de condicionales (IF/ELSE), sin embargo al realizar el código con el patrón anteriormente mencionado el objeto reacciona dependiendo del estado en el cual se encuentre, por su modo de funcionamiento es bastante útil para la realización de videojuegos debido a que se generan múltiples cambios en cada momento.

- **Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto**
  Los estados de la máquina son PAGO (PAYMENT), ORDEN (ORDEN), y DISPENSAR (DISPENSE), por lo cual se generan transiciones entre ellos tan simples o complejas como se requieran, en este caso las transiciones dependen de el dinero insertado para pagar y si la orden es aceptada (si hay dinero suficiente para cubrir el cambio, o si hay producto para cubrir la orden misma), como el patrón define varias clases para manejar cada estado general en el proyecto de este modo se evidencia *(Figura 2)*.



  La clase App es la que inicializa la aplicación, la clase machine es la clase principal donde se encuentran las clases que hacen posibles los estados y las transiciones entre los mismos

- **¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?**
  El proyecto tiene por objetivo final imitar el comportamiento de una máquina de café, para cumplir dicho objetivo divide el problema y lo resuelve con 3 sub máquinas que cada una tiene acciones distintas, en este caso concreto de la máquina de café al implementar un patrón de comportamiento dependiendo de lo que se requiera hacer con la máquina en dicho momento hace mucho más simples varios procesos.

El patrón State ofrece ventajas como modularidad y mantenibilidad al representar los diferentes estados de la máquina de café como clases separadas. Esto permite una fácil modificación y ampliación de los estados existentes, así como la adición de nuevos estados en el futuro. Además, el patrón proporciona flexibilidad y extensibilidad al permitir agregar nuevos estados o modificar el comportamiento existente sin afectar al resto del código. También brinda un control claro del flujo de la máquina de café, ya que cada estado puede determinar su transición al siguiente estado y establecer reglas específicas según su comportamiento.

- **¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?**
  El uso del patrón State puede introducir complejidad adicional debido a la necesidad de administrar y cambiar entre diferentes estados. Esto puede aumentar la cantidad de código y dificultar su comprensión. También existe la posibilidad de una sobrecarga adicional debido a la gestión de los estados, lo que puede afectar ligeramente el rendimiento en comparación con una implementación más directa. Además, puede requerir una curva de aprendizaje adicional para los desarrolladores menos familiarizados con el patrón, lo que implica un tiempo y esfuerzo extra para comprender y aplicar correctamente el patrón State en el contexto de la máquina de café.

- **¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?**
  La implementación del patrón de diseño State en un programa que imita el comportamiento de una máquina de café puede presentar ciertos desafíos. En este contexto, la aplicación puede volverse compleja debido a la necesidad de representar múltiples estados y gestionar las transiciones entre ellos. Además, si la máquina de café imitada es simple, la estructura y lógica del patrón State pueden parecer excesivas y no proporcionar una solución eficiente. Por tanto, es importante considerar alternativas y simplificar el diseño para mantener el código comprensible y adecuado a las necesidades específicas del programa de la máquina de café.

  1. Simplificar la estructura: Si la implementación actual del patrón State resulta demasiado compleja, se puede revisar y simplificar la estructura. Esto implica evaluar si todos los estados y transiciones son realmente necesarios y eliminar aquellos que no aporten valor. También se puede considerar agrupar comportamientos similares en un único estado para reducir la cantidad de clases involucradas.

  2. Optimizar el rendimiento: Si la sobrecarga asociada con el patrón State es un problema, se pueden implementar optimizaciones para mejorar el rendimiento. Esto podría incluir el uso de técnicas como la caché de resultados o la minimización de operaciones costosas en cada transición de estado. Además, se pueden aplicar técnicas de diseño y algoritmos eficientespara reducir la complejidad temporal y espacial.

  3. Simplificar la transición de estados: Si la lógica de transición entre los estados se vuelve confusa o complicada, se puede introducir un enfoque más

estructurado para manejar las transiciones. Por ejemplo, se podría utilizar una máquina de estados finitos (FSM) o una tabla de transiciones para definir claramente las condiciones y acciones asociadas con cada transición.

4. Proporcionar documentación y ejemplos claros: Si los desarrolladores tienen dificultades para comprender y aplicar correctamente el patrón State, se puede proporcionar documentación clara y ejemplos prácticos. Esto incluye explicar los conceptos clave del patrón y proporcionar casos de uso específicos relacionados con una máquina de café para mostrar cómo se aplica en la práctica.