Práctica 2, State, Template y Iterator.

Rojas Reyes Saúl Adrián.

114006224.

1. Menciona los principios de diseño esenciales del patrón State, Template e Iterator. Menciona una desventaja de cada patrón.

El patrón State se basa en alterar el comportamiento de un objeto específico en base a un cambio de estado interno. Se define un objeto, y una interfaz de estado, este objeto guarda en él una instancia de cada estado disponible, además de una asignación a un estado actual, estos deben implementar a la interfaz. El objeto manda a llamar los métodos de su estado actual, y dependiendo de cual sea este, el comportamiento cambiará. Conforme van aumentando la cantidad de estados disponibles aumenta en gran manera la cantidad de código el cual mantener y puede llegar a dar problemas como incongruencias.

El patrón Template se basa en diseñar el esqueleto de un algoritmo en un método, las clases que extiendan de esta redefinirá partes de este algoritmo sin alterar su estructura original. Se define una clase abstracta con un algoritmo central, este manda a llamar a más métodos de la clase, los métodos que queden como abstractos deben ser sobrecargados por las clases extensoras. La desventaja de este patrón es que se puede crear ambigüedad debido a el uso de tantos métodos abstractos.

El patrón Iterador es bastante fácil, consiste en darle a todas tus estructuras de datos, o clases que tengan alguna estructura interna, una manera de recorrerlas sin alterar su estructura interna, al implementar iterador se pueden recorrer todas con el mismo procedimiento, incluso desde los mismos métodos. Realmente no veo desventajas con este patrón, casi todas las estructuras de datos tienen iteradores por esa razón.

Sobre la práctica. Esta se maneja desde la consola al ingresar órdenes al robot, estas órdenes son:

suspenderse
caminarACliente -argumento-
atender
cocinar -argumento-
salir

Siempre se pueden ingresar -argumento-, pero solo se tomarán en cuenta al cocinar o al caminar, caminar puede o no recibir un nombre de cliente, cocinar debe de recibir forzosamente una id de un platillo, solo es necesario separarlo con un espacio. Se puede ingresar y salir en cualquier estado para terminar la simulación. El sistema es permisivo con acentuación, mayúsculas, minúsculas y puntuación, pero los errores de ortografía hacen fallar la orden, si la orden no se reconoce se pide otra.

La práctica se ejecuta con la clase Practica2, la simulación se encuentra en su main, yo usé el método javac Practica2.java para compilarla, luego java Practica2, debería funcionar correctamente.