Modelado y programación 2022-1

Práctica 2: Implementación de los patrones State, Template e Iterator

Bernal Márquez Erick. 317042522
Deloya Andrade Ana Valeria. 317277582

- ***Parte teórica:***

1. *Menciona los principios de diseño esenciales del patrón State, Template e Iterator.*

State:
El patrón permite a un objeto cambiar su comportamiento cuando su estado interno cambia a partir de transiciones diseñadas por el programador en un diagrama de estados. El objeto al cambiar su estado, aparenta haber cambiado de clase.

Los principios que sigue este patrón son:

- ★ Responsabilidad única. Los estados están separados en clases únicas.
- ★ Principio de Abierto Cerrado. Las clases están cerradas para modificar pero abiertas para expandir.

Template:
Define el esqueleto de un algoritmo en una superclase permitiendo que las subclases difieran en algunos pasos. Esto hace que las clases hijas sigan *casi* el mismo método.

Los principios que sigue este patrón son:

- ★ No nos llames, nosotros te llamamos. Las super clases dirigen el show, estas mandaran a llamar las subclases cuando sea necesario.

Iterator:
Provee una forma de acceder a los elementos de un objeto que los contiene de manera secuencial, sin exponer su representación interna.

Los principios que sigue este patrón son:

- ★ Responsabilidad única. El iterable solo se encarga de existir y el Iterador solo de iterar
- ★ Principio de abierto cerrado. Podemos agregar más colecciones e iteradores sin descomponer la estructura principal.

2. *Menciona una desventaja de cada patrón.*

   State:
   - ❖ Implementarlo puede ser excesivo si la clase tiene pocos estados a los cuales cambiar o cambia raramente

   Template:
   - ❖ Puede que se viole el *principio de sustitución de Liskov*, al suprimir una implementación por defecto de la clase padre.
   - ❖ Suele ser difícil de mantener si se agregan más pasos.

   Iterator:
   - ❖ Algunas veces puedes ser menos eficiente que recorrer directamente los elementos de la colección

- ***Sobre la práctica:***
  El patrón state lo implementa el robot, pues este tiene diferentes comportamientos dependiendo en qué estado se encuentre.

  Iterator al momento de recorrer cualquier estructura que tenga a hamburguesas como elementos.

  Y template en la preparación de cada hamburguesa.

  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 robot inicia en un estado suspendido y lo que único que puede hacer es ponerse a caminar con el método *activar()*
  Una vez que esté caminando el robot puede volver a suspenderse o atender al cliente, si atiende el robot le mostrará el menú de Hamburguesas al cliente.
  Ya que el robot esté atendiendo éste debe recibir la orden del cliente con *atender()*, pues de esta manera guardará la orden del cliente, posteriormente puede ponerse a cocinar (algo así como mandarlo a la cocina) una vez teniendo la orden del cliente.

Ahora sí, el robot cocina la hamburguesa, después vuelve al estado de suspendido