**Trabajo en clase**

**Santiago Martínez Martínez**

**Programación Orientada a Objetos**

**Patrón estado**

**PATRON DE DISEÑO STATE**

**Nombre:** Patrón de diseño state o de estado.

**¿Qué es?**

Este patrón resulta útil cuando necesitamos que un objeto se comporte de forma diferente dependiendo del estado interno en el que se encuentre en cada momento.

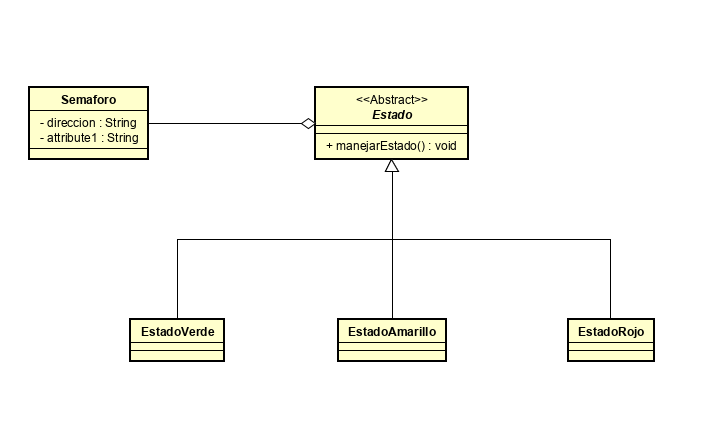
Es decir que el objeto puede de tener un estado diferente dependiendo del contexto en donde se encuentre.

**¿Cuál es el problema que busca resolver?**

Este patrón de diseño busca mejorar la implementación de soluciones en donde el objeto cambia de estado en el tiempo, es decir, busca implementar la solución de una mejor manera para este caso en especial.

**Solución**

En este caso lo que se busca es definir el objeto y que este tenga un atributo de tipo Estado, con esto buscamos que este estado puede ser cambiante dependiendo de la situación. En este caso hablamos de un semáforo, el cual cambia de estado dependiendo de su contador.



**Ejemplo:**

El ejemplo más común que nos encontramos es el semáforo, el cual cambia su estado dependiendo de su cronometro, es decir, por ejemplo, el estado del semáforo después de 40 segundos, cambia de estado **EstadoRojo** a **EstadoAmarillo**, lo que sucede aquí es que al cambiar el estado su comportamiento debe de cambiar de la misma forma.

**Patrón del curso**

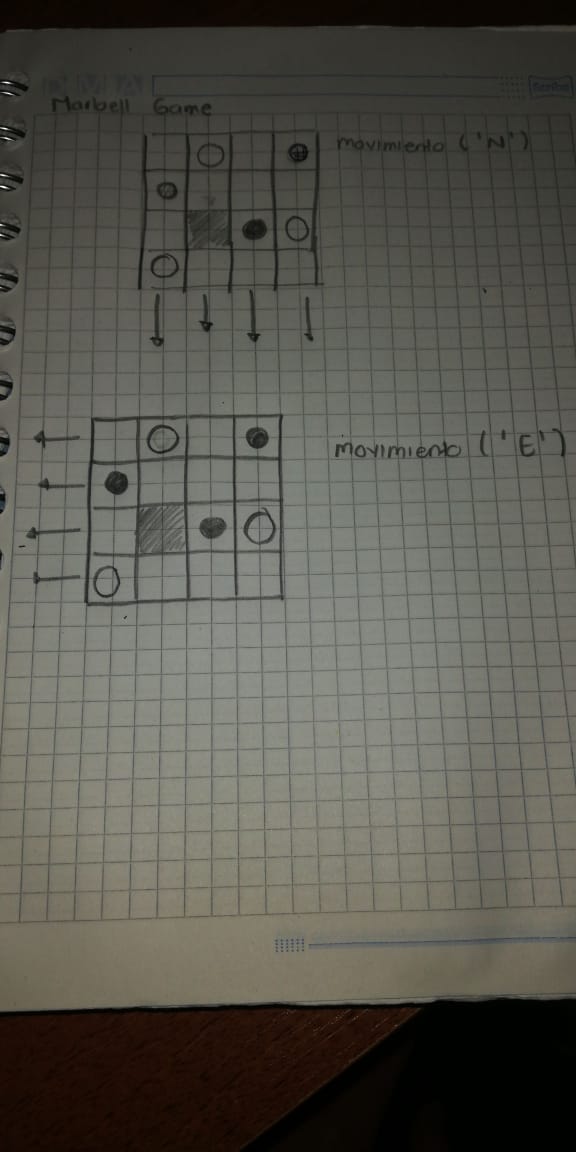
Considerando sus experiencias como desarrolladores en POOB, propongan un patrón de diseño.

**Nombre:** Patrón de diseño basado en ejemplos e intentos.

**¿Qué es?**

Este patrón de diseño lo que busca es que antes de comenzar a programar nuestra solución, hagamos varios ejemplos y plasmarlos en una hoja, con esto lo que buscamos es que al tratar de resolver el problema en la hoja tengamos más claro cada paso para llegar a la solución, al tener ya la experiencia de solucionarlo en la hoja, será más fácil plasmar esto en código.

Al terminar de codificar y ver que todo funcione, haremos un repaso del código y miraremos que líneas se pueden acortar o que métodos no son tan necesarios o están mal implementados. Con esto no solo queremos tener un código que funcione, sino que sea óptimo.

**Solución**

**Ejemplo:**

Tomaremos como ejemplo el laboratorio número 5 del curso, en este laboratorio lo que nos pedían era implementar un juego el cual hiciera referencia al juego Marbell Game. Para que este juego funcionara adecuadamente teníamos que implementar de manera correcta el movimiento de los objetos, que en estos casos serían las canicas.

Al realizar una serie de ejemplos en una hoja de papel, me di cuenta que al realizar los movimientos teníamos que comenzar a mover las fichas de lado contrario y comenzando desde el lado opuesto. Por ejemplo: Si el usuario quiere levantar el tablero del **NORTE**, nos damos cuenta que las canicas se mueven hacia el sur, pero no podemos mover primero las canicas de arriba hacia abajo, sino que tenemos que mover primero las que estén abajo para que todas las fichas si se muevan correctamente.

Toda la lógica de este caso de uso, fue descubierta después de haber hecho muchos ejemplos y casos.