**Patrón de comportamiento:**

**State**

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Busca que un objeto pueda reaccionar según su estado interno. Si bien muchas veces esto se puede solucionar con un boolean o utilizando constantes, esto suele terminar con una gran cantidad de if-else, código ilegible y dificultad en el mantenimiento. La intención del State es desacoplar el estado de la clase en cuestión.

Es complicado cuando se debe tener en cuenta el cambio de comportamientos y estados del objeto dentro del mismo bloque de código. El patrón State propone una solución a esta complicación, creando un objeto por cada estado posible.

*Este patrón debe ser utilizado cuando:*

* El comportamiento de un objeto depende de un estado, y debe cambiar en tiempo de ejecución según el comportamiento del estado.
* Cuando las operaciones tienen largas sentencias con múltiples ramas que depende del estado del objeto.

*Consecuencias*

* Se localizan fácilmente las responsabilidades de los estados concretos, dado que se encuentran en las clases que corresponden a cada estado.
* Mayor claridad en el desarrollo y el mantenimiento posterior.
* Cambios de estados explícitos en el código y en tiempo de ejecución.
* Facilita la ampliación de estados mediante herencia, sin afectar a la ventanilla.
* Se incrementa el número de subclases.