**El patrón de diseño State** es reconocido por que modifica su comportamiento dependiendo del estado en el que se encuentra la aplicación. Para lograr esto se necesita crear una serie de clases que simbolizaran los distintos estados por los que puede pasar la aplicación; dicho de otro modo , es necesaria una clase por cada uno de los estados por los que la aplicación puede pasar.

A veces, cuando el entorno en el que se está desarrollando necesita que un determinado objeto tenga diferentes comportamientos según el estado en el que se encuentre, resulta complejo poder manejar ese cambio de comportamiento y los estados de este, todo en el mismo bloque de código.

El patrón State ofrece una solución a esta dificultad, creando un objeto por cada uno de los estados posible del objeto en el que se implementa este patron.



Ahora explicare los elementos que forman el patrón State:

* Context: Es la clase que va ha cambiar de estado, la cual tiene entre sus propiedades el estado actual. En el ejemplo que he realizado este seria la mascota como tal.
* AbstractState: Clase base para la generación de los distintos estados.
* ConcreteState: Cada una de estas clases representa un posible estado a los que el Contex(Mascota) puede pasar, por lo que tendremos un ConcreteState por cada estado posible. Esta clase debe de heredar de AbstractState.

Otros lenguajes

* En C+: El patrón State se utiliza habitualmente en C# para convertir las enormes máquinas de estados basadas en **switch**, en objetos. (Enlace a un ejemplo en la bibliografía)
* En PHP : El patrón State se utiliza ocasionalmente en PHP para convertir grandes y voluminosas máquinas de estados basadas en operadores **switch**, en objetos. (Enlace a un ejemplo en la bibliografía)

En resumen el patrón State se utiliza en los distintos lenguajes para convertir

máquinas de estados en objetos.

Propósito

Permite a un objeto realizar de manera distinta unas actiones segun el estado en le que se encuentre.

El ejemplo que he utilizado es una mascota que cambia entre: Hambriento, aburrido y cansado de manera que se van realizando ciertas acciones.

* Al inicio del programa he creado un objeto de tipo **Mascota** ( Que por defecto su estado está inicializado en **Aburrido**) y mostramos por consola las distintas acciones que se puden realizar segun que número se introduzca, aunque solo respondera de manera afirmativa en el caso de jugar ya que esta **aburrido**. Posteriormente cambiara a el estado de cansado por lo que querra dormir.Por ultimo antes de volver al estado de aburrido pasara por el de hambriento por lo que habra que realizar la acción de '**Alimentar**'

Bibliografía:

* <https://es.wikipedia.org/wiki/State_(patr%C3%B3n_de_dise%C3%B1o)>
* <https://reactiveprogramming.io/blog/es/patrones-de-diseno/state>
* Ejemplo C++: <https://refactoring.guru/es/design-patterns/state/csharp/example#:~:text=State%20es%20un%20patr%C3%B3n%20de,cuando%20cambia%20su%20estado%20interno>.
* Ejemplo PHP: <https://refactoring.guru/es/design-patterns/state/php/example#:~:text=State%20es%20un%20patr%C3%B3n%20de,cuando%20cambia%20su%20estado%20interno>.