Authors:
Delgado Díaz Itzel Azucena
Martínez Mendoza Miguel Ángel
Rojas Espinoza Luis Ángel
6 / Septiembre / 2018
Método Template y Patrón State

1. Aplicabilidad de Template

Es factible cuando se quiere hacer una factorización del comportamiento que tienen en común varias subclases.
Implementamos las partes fijas de un algoritmo una sola vez y dejamos que las subclases implementen las partes que pueden ser modificadas. Es decir, controlamos las ampliaciones de las subclases, convirtiendo en métodos plantilla aquellos que pueden ser redefinidos.
Favorece la reutilización del código. Este método es muy útil para construir bibliotecas.

2. ¿Para qué tipos de problemas nos son útil para poderlos usar en nuestra solución?

El método template no es útil usarlo cuando tienes métodos específicos que nunca van a cambiar. Es mejor utilizarlo cuando se construyen jerarquías de clases complejas para una aplicación, a menudo se duplican distintas partes de código. Esa situación no es deseable, porque la intención es reutilizar tanto código como sea posible.
Este patrón se vuelve de especial utilidad cuando es necesario realizar un algoritmo que sea común para muchas clases, pero con pequeñas variaciones entre una y otras. Cuando no ocurre esto es mejor crear algoritmos distintos para cada problema.

El patrón State se utiliza En determinadas ocasiones cuando se requiere que un objeto tenga diferentes comportamientos según el estado en que se encuentra. Esto resulta complicado de manejar, sobre todo cuando se debe tener en cuenta el cambio de comportamientos y estados de dicho objeto, todos dentro del mismo bloque de código. El patrón State propone una solución a esta complicación, creando un objeto por cada estado posible. Cuando sabemos el estado del objeto pero no lo vamos a utilizar entonces no es necesario usar State .

3. ¿Cuáles son los inconvenientes de usar estos patrones de diseño?

En el método template se puede producir ambigüedad si no escribimos bien los métodos. Si el método template llama demasiados métodos abstractos, se cansara pronto de usar AbstractClass como súper clase.

El único inconveniente en el patrón State es que incrementamos demasiado el número de subclases, tanto como sea necesario.

PD. Para compilar la práctica solo es necesario utilizar ant.