# El patrón de método de plantilla

El patrón de método de plantilla se puede utilizar en situaciones en las que existe un algoritmo, algunos pasos de los cuales podrían implementarse de múltiples formas diferentes. El patrón del método de plantilla sugiere mantener el esquema del algoritmo en un método separado denominado **método de plantilla dentro de una clase, que puede denominarse clase de plantilla,** dejando de lado las implementaciones específicas de las porciones variantes.

La clase Plantilla no necesariamente tiene que dejar la implementación a las subclases en su totalidad. En cambio, como parte de proporcionar el esquema del algoritmo, la clase Plantilla también puede proporcionar cierta cantidad de implementación que puede considerarse invariante en diferentes implementaciones. Incluso puede proporcionar una implementación predeterminada para las partes variantes, si corresponde. Solo se implementarán detalles específicos dentro de diferentes subclases. Este tipo de implementación elimina la necesidad de código duplicado, lo que significa que se debe escribir una cantidad mínima de código. Con el lenguaje de programación Java, la clase Plantilla se puede diseñar de una de las dos formas siguientes.

**Clase abstracta**

Este diseño es más adecuado cuando la clase Plantilla proporciona solo el esquema del algoritmo sin ninguna implementación predeterminada para sus partes variantes. Suponiendo que los diferentes pasos del algoritmo se pueden convertir en métodos individuales:

* El método Template puede ser un método concreto y no abstracto con llamadas a otros métodos que representan diferentes pasos del algoritmo.
* La clase Template puede implementar partes invariantes del algoritmo como un conjunto de métodos no abstractos.
* El conjunto de pasos variantes se puede diseñar como métodos abstractos. Se pueden proporcionar implementaciones específicas para estos métodos abstractos dentro de un conjunto de subclases concretas de la clase Plantilla abstracta.

En este diseño, la clase Abstract declara métodos y cada una de las subclases implementa estos métodos de una manera específica sin alterar el esquema del algoritmo.

**Clase Concreta**

Este diseño es más adecuado cuando la clase Plantilla proporciona, además del esquema del algoritmo, la implementación predeterminada para sus partes variantes. Suponiendo que los diferentes pasos del algoritmo se pueden convertir en métodos individuales:

* El método Template puede ser un método concreto y no abstracto con llamadas a otros métodos que representan diferentes pasos del algoritmo.
* La clase Template puede implementar partes invariantes del algoritmo como un conjunto de métodos no abstractos.
* El conjunto de pasos variantes se puede diseñar como métodos no abstractos con la implementación predeterminada. Las subclases de la clase Template pueden sobrescribir estos métodos para proporcionar implementaciones específicas sin alterar el esquema del algoritmo.

De ambas estrategias de diseño, se puede ver que la implementación del patrón de plantilla se basa en gran medida en la herencia y la sobrescritura de funciones. Por lo tanto, siempre que se usa la herencia para implementar los detalles, se puede decir que el patrón del Método de plantilla se usa en su forma más simple.