Método Template:

Na [programação orientada a objetos](https://en.wikipedia.org/wiki/Object-oriented_programming) , o método template é um dos padrões de [design](https://en.wikipedia.org/wiki/Software_design_pattern)[comportamental](https://en.wikipedia.org/wiki/Behavioral_pattern) identificados por Gamma et al. no livro [Design Patterns](https://en.wikipedia.org/wiki/Design_Patterns) . O método template é um método em uma superclasse, geralmente uma superclasse abstrata, e define o esqueleto de uma operação em termos de várias etapas de alto nível. Essas etapas são implementadas por métodos auxiliares adicionais na mesma classe que o método de modelo.

Os métodos auxiliares podem ser [métodos abstratos](https://en.wikipedia.org/wiki/Abstract_method) , caso em que as subclasses são necessárias para fornecer implementações concretas, ou [métodos de gancho](https://en.wikipedia.org/w/index.php?title=Hook_methods&action=edit&redlink=1) , que possuem corpos vazios na superclasse. [As subclasses](https://en.wikipedia.org/wiki/Subclass_(computer_science)) podem (mas não são obrigadas a) personalizar a operação [substituindo](https://en.wikipedia.org/wiki/Method_overriding) os métodos de gancho. A intenção do método template é definir a estrutura geral da operação, enquanto permite que as subclasses refinem ou redefinam certas etapas.

Este padrão tem duas partes principais:

* O "método modelo" é implementado como um método em uma [classe base](https://en.wikipedia.org/wiki/Base_class) (geralmente uma [classe abstrata](https://en.wikipedia.org/wiki/Abstract_class)). Esse método contém código para as partes do algoritmo geral que são invariáveis. O modelo garante que o algoritmo abrangente seja sempre seguido.  No método template, partes do algoritmo que podem variar são implementadas enviando mensagens self que solicitam a execução de métodos auxiliares adicionais. Na classe base, esses métodos auxiliares recebem uma implementação padrão, ou nenhuma (ou seja, podem ser métodos abstratos).
* As subclasses da classe base "preenche" as partes vazias ou "variantes" do "modelo" com algoritmos específicos que variam de uma subclasse para outra.  É importante que as subclasses não substituam o próprio método template.

Em tempo de execução, o algoritmo representado pelo método template é executado enviando a mensagem template para uma instância de uma das subclasses concretas. Por meio de herança, o método de modelo na classe base começa a ser executado. Quando o método template envia uma mensagem para self solicitando um dos métodos auxiliares, a mensagem será recebida pela sub-instância concreta. Se o método auxiliar foi substituído, a implementação de substituição na sub-instância será executada; se não tiver sido substituído, a implementação herdada na classe base será executada. Esse mecanismo garante que o algoritmo geral siga as mesmas etapas todas as vezes, permitindo que os detalhes de algumas etapas dependam de qual instância recebeu a solicitação original para executar o algoritmo.

Esse padrão é um exemplo de [inversão de controle](https://en.wikipedia.org/wiki/Inversion_of_control) porque o código de alto nível não determina mais quais algoritmos devem ser executados; um algoritmo de nível inferior é selecionado em tempo de execução.

Algumas das mensagens próprias enviadas pelo método de modelo podem ser para métodos de gancho. Esses métodos são implementados na mesma classe base que o método template, mas com corpos vazios (ou seja, eles não fazem nada). Os métodos de gancho existem para que as subclasses possam substituí-los e, assim, ajustar a ação do algoritmo sem a necessidade de substituir o próprio método de modelo. Em outras palavras, eles fornecem um "gancho" no qual "travar" as implementações de variantes.

Uso:

O método template é usado em frameworks, onde cada um implementa as partes invariáveis ​​da arquitetura de um domínio, enquanto fornece métodos de gancho para customização. Este é um exemplo de [inversão de controle](https://en.wikipedia.org/wiki/Inversion_of_control). O método de modelo é usado pelos seguintes motivos.

* Ele permite que as subclasses implementem comportamentos variados (por meio da [substituição](https://en.wikipedia.org/wiki/Method_overriding_(programming)) dos métodos de gancho).
* Evita a duplicação no código: o fluxo de trabalho geral do algoritmo é implementado uma vez no método template da classe abstrata, e as variações necessárias são implementadas nas subclasses.
* Ele controla o(s) ponto(s) em que a especialização é permitida. Se as subclasses simplesmente substituíssem o método do modelo, elas poderiam fazer alterações radicais e arbitrárias no fluxo de trabalho. Em contraste, substituindo apenas os métodos de gancho, apenas alguns detalhes específicos do fluxo de trabalho podem ser alterados, e o fluxo de trabalho geral permanece intacto.