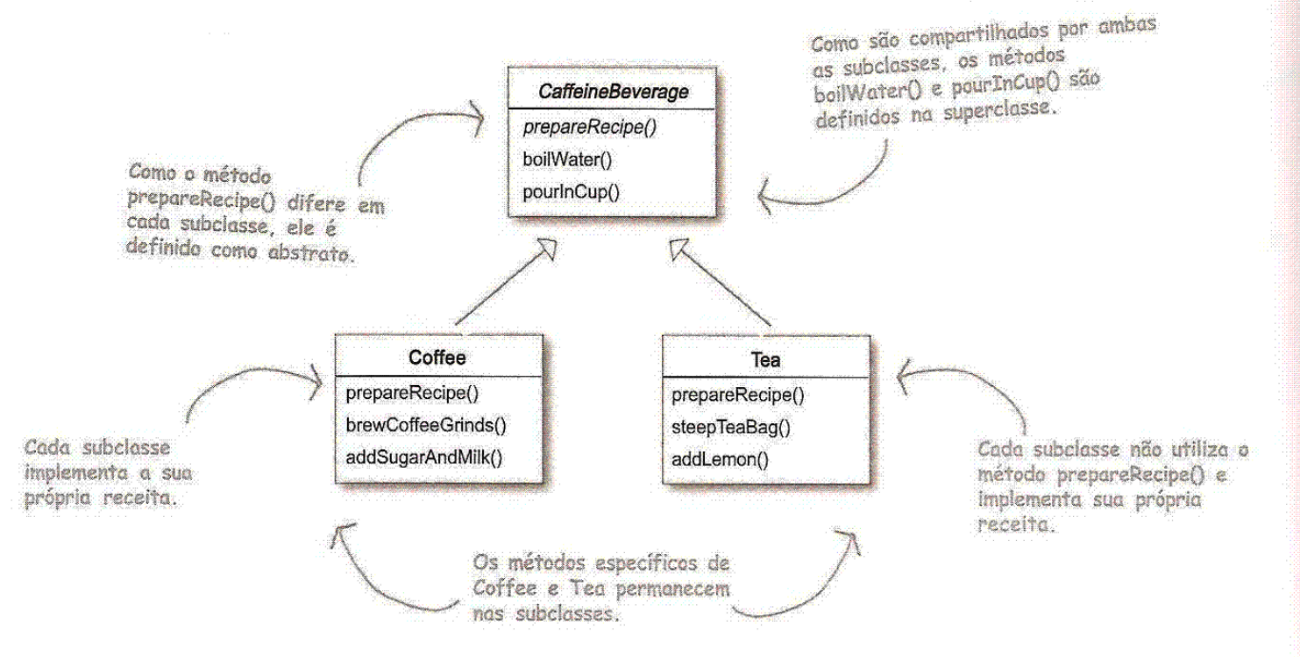
Resumo:

Um **Template Method** auxilia na definição de um algoritmo com partes do mesmo definidos por métodos abstratos. As subclasses devem se responsabilizar por estas partes abstratas, deste algoritmo, que serão implementadas, possivelmente de várias formas, ou seja, cada subclasse irá implementar à sua necessidade e oferecer um comportamento concreto construindo todo o algoritmo.

O Template Method fornece uma estrutura fixa, de um algoritmo, esta parte fixa deve estar presente na superclasse, sendo obrigatório uma classe abstrata que possa conter um método concreto, pois em uma interface só é possível conter métodos abstratos que definem um comportamento, esta é a vantagem de ser uma Classe Abstrata porque também irá fornecer métodos abstratos às suas subclasses, que por sua vez herdam este método, por herança, e devem implementar os métodos abstratos fornecendo um comportamento concreto aos métodos que foram definidos como abstratos. Com isso certas partes do algoritmo serão preenchidos por implementações que irão variar, ou seja, implementar um algoritmo em um método, postergando a definição de alguns passos do algoritmo, para que outras classes possam redefini-los

Diagrama uml:



Usando como exemple para demonstrar o template method podemos usar a produção de café e cha, como vemos no diagrama ambas a receitas compartilham as atepas de ferver a agua(boilWater()) e colocar a bebida em uma chicara(pourInCup()) divergindo nas outras etapas.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

Abstraindo as recitas temos quem ambos os processos brewCoffe e steeptea podem ser resumido em um único objeto brew() que faz a função de uma infusão do material em agua assim se tornado um template que sera alterado pelas subclasses acomtece o mesmo com os condimentos

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Virando assim um um template method