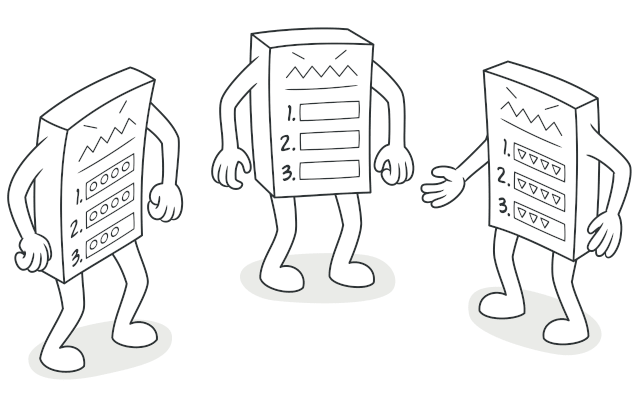
**Padrão de Projeto:** **Template Method**

O Template Method é um padrão de projeto comportamental que define o esqueleto de um algoritmo na superclasse mas deixa as subclasses sobrescreverem etapas específicas do algoritmo sem modificar sua estrutura.



Esse padrão de projeto com o objetivo de resolver esse problema:

Imagine que você está criando uma aplicação de mineração de dados que analisa documentos corporativos. Os usuários alimentam a aplicação com documentos em vários formatos (PDF, DOC, CSV), e ela tenta extrair dados significativos desses documentos para um formato uniforme.

A primeira versão da aplicação podia funcionar somente com arquivos DOC. Na versão seguinte, ela era capaz de suportar arquivos CSV. Um mês depois, você a “ensinou” a extrair dados de arquivos PDF.

Em algum momento você percebeu que todas as três classes tem muito código parecido. Embora o código para lidar com vários formatos seja inteiramente diferente em todas as classes, o código para processamento de dados e análise é quase idêntico. Não seria bacana se livrar da duplicação de código, deixando a estrutura do algoritmo intacta?

Havia outro problema relacionado com o código cliente que usou essas classes. Ele tinha muitas condicionais que pegavam um curso de ação apropriado dependendo da classe do objeto processador. Se todas as três classes processantes tiverem uma interface comum ou uma classe base, você poderia eliminar as condicionais no código cliente e usar polimorfismo quando chamar métodos em um objeto sendo processado.

**Estrutura:**

****

**Componentes:**

Os componentes desse padrão de projeto são as classes abstratas e as classes concretas

A **Classe Abstrata** declara métodos que agem como etapas de um algoritmo, bem como o próprio método padrão que chama esses métodos em uma ordem específica. Os passos podem ser declarados como abstratos ou ter alguma implementação padrão.

As **Classes Concretas** podem sobrescrever todas as etapas, mas não o próprio método padrão.