**Artigo dirigido sobre**

**Template Method / Observer**

**Guilherme Fernandes, Caio Sousa, Breno Silva, Fernando Ramilo, Pedro Floriano**

, caiosousa.2906@aluno.una.br, brenorezende.4457@aluno.una.br, pedroaguiar.0814@aluno.una.br

Una Contagem – Centro Universitário Una (UNA) – Contagem – MG – Brasil

***Abstract.*** The following article aims to point out and exemplify the main characteristics of the themes of two Template Method / Observer programming methods (Standard Method / Observer or Event Assistant), which are methods or project patterns, which result, each one of them in certain results. Thus, each one has its final objective, which classifies each one of them for certain projects.

***Resumo.*** O artigo a seguir tem como objetivo pontuar e exemplificar as principais característica dos temas de dois métodos de programação Template Method / Observer (Método Padrão / Observador ou Assistente do Evento), que são métodos ou padrões de projetos, que resultam, cada um deles em determinados resultados. Sendo assim cada qual tem seu objetivo final, o que classifica cada um deles para determinados projetos.

**Introdução**

Através deste artigo, o grupo busca trazer com maior clareza as características mais ressaltantes sobre “Template Method” e “Observer”, que são padrões de projetos comportamentais. Sendo o Template Method ou Método Padrão, que nada mais é que padrão de projeto comportamental que define a estrutura do algoritmo na superclasse porem as subclasses, sobrescreverem algumas etapas do algoritmo, não alterando a sua estrutura. E o Observer ou Observador que também e um padrão de projeto comportamental que permite a definição do mecanismo de assinatura, podendo assim gerar múltiplos alertas com os objetos sobre qualquer evento que ocorra com o objeto observado.

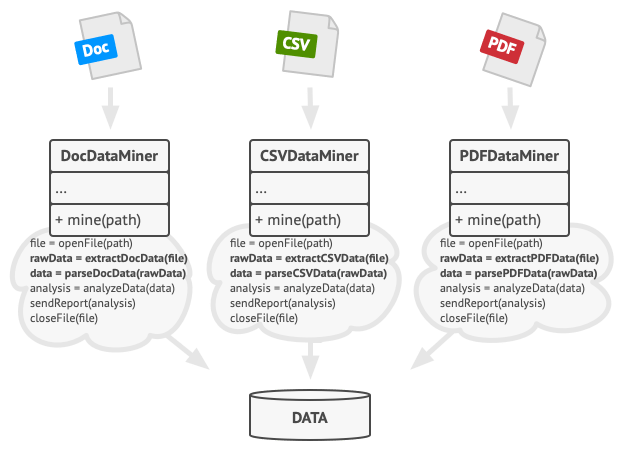
**Template Method ou Método Padrão**

Como introduzido, O **Template Method** é a forma comportamental de padronizar o projeto, definindo o esqueleto do algoritmo dentro da superclasse, porém deixando as subclasses sobrescreverem determinadas etapas do algoritmo, não permitindo que modifiquem sua estrutura.

Trazendo uma exemplificação com o mundo real, nesse tipo de abordagem com o Template Method, pode ser utilizada na construção em grande escala de casas, onde existe um padrão para as construções com diversos pontos que permitiriam o dono de alterar ou fazer alguns ajustes em alguns detalhes na moradia final. Sendo assim cada etapa da construção, poderia ser sutilmente alterado para resultar em uma moradia diferente das outras.

Com isso, o Template Method traz como sugestão a repartição do algoritmo, em diversas etapas, que cada etapa por sua vez, transforma em métodos. Dentro desses métodos, traz diversas series de chamadas para esses métodos dentro de um único método padrão. Tornando a utilização do algoritmo, repartida em etapas dentro da subclasse fornecida pelo cliente. Resumindo, O método padrão quebra o algoritmo em etapas, permitindo que subclasses sobrescrevam essas etapas, mas não o método atual.

Já para exemplificar um problema que esse método gera, imagine o cenário onde você cria uma aplicação com a finalidade de minerar diversos dados que são analisados. O usuário alimenta a aplicação com diversos tipos de documentos, e extrai dados de grande importância desses documentos de diversos formatos. A aplicação seria dividida por versões, onde a primeira versão da aplicação funciona somente com arquivos do primeiro tipo. Na versão seguinte suportaria arquivos de um segundo tipo. E na sequência um terceiro tipo de arquivo e utilizado. Como mostra na imagem abaixo:



Assim, pode se notar que todas as três classes possuem uma estrutura de código muito parecida, mesmo lidando com vários formatos de arquivos diferentes, onde seria mais interessante unificar os tipos de códigos, devido a proximidade deles. Além disso, o código do cliente tem o problema de ter muitas condições para processar o objeto, sendo assim, as três classes possuem uma interface comum, permitindo que elimine as condições no código do cliente e usa um sentido poliforme para chamar os métodos de um objeto processado.

Para melhor entendimento, utilize o padrão Template Method:

- Para deixar os clientes com apenas as etapas particulares do algoritmo, e não com todo o algoritmo e estrutura.

- Também permitindo que o algoritmo transforme em um algoritmo monolítico, deixando as etapas individuais, facilitando de ser compreendido pelas subclasses, deixando a superclasse sem alterações.

**Observer**

Outro método introduzido nesse artigo é o “Observer” ou “Observador”, que também é um padrão de projeto comportamental, aonde nele define um mecanismo de assinatura que notifica múltiplos objetos sobre qualquer tipo de evento com o objeto que está sendo observado.

Para exemplificar, com uma situação real, seria a assinatura de um jornal ou revista, onde exclui o fator de ir até a banca para atualizar as edições. Fazendo com que a publicadora mande sempre a nova versão ou até mesmo com antecedência.

A publicadora por sua vez possui uma lista de pessoas que são assinantes, e sabe quais são as revistas que desperta melhor interesse no público, sendo assim, os assinantes podem sair dessa listagem a qualquer momento, quando desejarem, que não recebam mais as atualizações.

Com isso, o objeto que tem uma situação de interesse, normalmente é denominado de sujeito, onde o mesmo pode notificar outros objetos sobre a mudança do estado que, por sua vez pode ser chamado de publicador. Os objetos que são interessados são chamados de assinantes.

Com todas essas informações, criasse o raciocínio que, o padrão Observer sugere a adição de um mecanismo de assinatura onde a classe publicadora, onde os objetos individuais possam assinar ou dessainar uma série de eventos que partem da publicadora.

Assim toda vez que um evento importante ocorre com a publicadora, ele se torna seu assinante e chama os métodos de notificação para o sujeito. Na prática, pode tornar diversos eventos diferentes com a classe assinantes, onde elas acompanham os eventos. Sendo assim, e crucial que os assinantes tragam a mesma interface e que a publicadora comunique com eles através da interface, que deve notificar através de um método.

O problema dessa metodologia é a utilização constante do interessa do assinante, pois assim, ele faz diversas validações antes de fato obter o produto assinado. Com isso os assinantes são, são notificados constantemente em ordem aleatória.

**Conclusão**

Como decorrido no artigo, ambos os métodos, buscam formas especificas de chegar a um ponto conclusivo de um determinado projeto. Percebendo assim, que cada um tem seu determinado objetivo com seus pontos positivos e negativos. Lembrando sempre a importância da utilização da maneira correta e coesa das regras dos métodos em questão.

**Referencias**

Blog sobre tecnologia: Interpretação feita baseada na leitura dos conteúdos abaixo listados. https://refactoring.guru/pt-br/design-patterns/behavioral-patterns / https://refactoring.guru/pt-br/design-patterns. Acesso em: 08 junho. 2022.