Seminário POO

Warley Ramires Gonçalves

Padrões de Projeto - State & Interpreter

1. Introdução

Os padrões de projeto são um conjunto de soluções típicas para problemas frequentemente encontrados durante o desenvolvimento de software. Eles podem ser aplicados independentemente da linguagem de programação utilizada, pois visam fornecer diretrizes gerais e abstratas para resolver questões recorrentes. Esses padrões surgiram como uma resposta à necessidade de boas práticas de programação e têm como objetivo auxiliar os desenvolvedores na criação de sistemas robustos, flexíveis e de fácil manutenção.

Ao utilizar padrões de projeto, os desenvolvedores podem se beneficiar de soluções testadas e comprovadas, evitando reinventar a roda a cada novo problema enfrentado. Esses padrões representam a experiência coletiva da comunidade de desenvolvedores e encapsulam soluções comprovadas para desafios comuns, como gerenciamento de dados, comunicação entre componentes, estruturação de código e tratamento de erros.

Além disso, os padrões de projeto promovem a reutilização de código e a modularidade, permitindo que partes específicas de um sistema sejam facilmente modificadas ou substituídas sem afetar o resto do código. Eles ajudam os desenvolvedores a criar sistemas mais flexíveis e adaptáveis, capazes de lidar com mudanças de requisitos e escalabilidade.

Em resumo, os padrões de projeto são ferramentas valiosas no desenvolvimento de software, pois fornecem soluções comprovadas para problemas comuns. Eles ajudam a promover boas práticas de programação, modularidade, reutilização de código e flexibilidade do sistema. Ao aplicar os padrões de projeto corretamente, os desenvolvedores podem economizar tempo, melhorar a qualidade do código e facilitar a manutenção do software ao longo do tempo.

2. Desenvolvimento

2.1.Surgimento

Os padrões de projeto surgiram como soluções eficientes e reutilizáveis para problemas recorrentes no desenvolvimento de software. A ideia de aplicar padrões a problemas de software ganhou destaque com a publicação do livro "Design Patterns: Elements of Reusable Object-Oriented Software" em 1994, escrito por Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, conhecidos como Gang of Four (GoF). O livro apresentou 23 padrões de projeto, que foram identificados a partir das experiências práticas e conhecimentos adquiridos pelos autores durante o desenvolvimento de sistemas complexos.

Desde então, os padrões de projeto se tornaram uma referência valiosa para criar sistemas robustos e flexíveis, e continuam evoluindo e sendo explorados pela comunidade de desenvolvedores.

A Gang of Four (GoF) desempenhou um papel significativo no estabelecimento dos padrões de projeto como um conceito no desenvolvimento de software. Seu livro "Design Patterns: Elements of Reusable Object-Oriented Software" se tornou um marco na área, apresentando 23 padrões de projeto comuns. Os autores, Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, documentaram soluções reutilizáveis para problemas frequentes e compartilharam seus conhecimentos adquiridos ao longo da carreira de ambos.

Esse trabalho pioneiro despertou um interesse renovado nos padrões de projeto e estimulou a comunidade de desenvolvedores a explorar e documentar mais padrões, levando a uma evolução contínua desse campo.

Os padrões de projeto propostos pela Gang of Four (GoF) são amplamente reconhecidos e utilizados na indústria de desenvolvimento de software. Eles fornecem um vocabulário comum e diretrizes para resolver problemas comuns, como a criação de objetos, a estruturação de classes e a definição de comportamentos.

Classificados em categorias como criação, estrutura e comportamento, são uma referência importante para a criação de sistemas flexíveis, modulares e de fácil manutenção. O trabalho da Gang of Four continua a influenciar o desenvolvimento de software e a estimular a busca por soluções elegantes e reutilizáveis para desafios recorrentes.

2.2. Tipos de Padrões de Projeto

Os padrões de projeto podem ser classificados em três categorias principais: criacionais, estruturais e comportamentais. Essas categorias representam diferentes aspectos e abordagens para resolver problemas específicos no desenvolvimento de software.

Os padrões criacionais lidam com o processo de criação de objetos, que pode ser complexo e apresentar desafios. Esses padrões visam fornecer soluções para evitar problemas de instanciação e para permitir um controle mais preciso no processo de criação de objetos. Eles se concentram em separar o processo de criação, conclusão e representação de um objeto, tornando-o mais flexível e fácil de gerenciar. Exemplos de padrões criacionais incluem o Singleton, o Builder e o Factory Method.

Os padrões estruturais estão relacionados à organização e estruturação das classes e dos relacionamentos entre elas. Eles ajudam a definir relações claras e flexíveis entre as entidades do sistema, facilitando a compreensão e a manutenção do código. Esses padrões podem ser usados para melhorar a composição de objetos, definir interfaces mais flexíveis ou separar responsabilidades entre as classes. Exemplos de padrões estruturais incluem o Adapter, o Decorator e o Composite.

Os padrões comportamentais estão centrados no comportamento e na comunicação entre objetos, sem que eles precisem ter conhecimento uns dos outros. Esses padrões se concentram em definir interações flexíveis e encapsuladas entre objetos, permitindo que eles cooperem de maneira eficiente e desacoplada. Eles abordam aspectos como a definição de algoritmos e comportamentos dinâmicos, tratamento de eventos e gerenciamento de fluxo de dados. Exemplos de padrões comportamentais incluem o Observer, o Strategy e o Command.

Cada categoria de padrões de projeto aborda uma área específica de preocupação no desenvolvimento de software e oferece soluções testadas e comprovadas para problemas recorrentes. Ao aplicar esses padrões, os desenvolvedores podem melhorar a estrutura, a flexibilidade e o comportamento de seus sistemas, resultando em um código mais organizado, de fácil manutenção e com maior reutilização de componentes.

2.3 Vantagens

O uso de padrões de projeto oferece várias vantagens significativas no desenvolvimento de software. Primeiramente, os padrões de projeto permitem a reutilização de soluções comprovadas para problemas recorrentes. Ao adotar esses padrões, os desenvolvedores podem aproveitar o conhecimento e a experiência acumulados pela comunidade, evitando a necessidade de reinventar a roda a cada novo problema enfrentado.

Além disso, o uso de padrões de projeto melhora a manutenção e a extensibilidade do código. Ao seguir padrões reconhecidos, os desenvolvedores podem estruturar o código de forma clara e modular, separando preocupações e responsabilidades. Isso facilita a compreensão e a manutenção do sistema, além de permitir que o código seja facilmente adaptado e modificado para atender a novos requisitos ou cenários.

Outra vantagem dos padrões de projeto é a padronização e a comunicação eficaz entre os membros da equipe de desenvolvimento. Ao utilizar padrões documentados e bem estabelecidos, os desenvolvedores podem se comunicar de maneira clara e precisa sobre a estrutura e o comportamento do sistema. Isso melhora a colaboração, o compartilhamento de conhecimento e a compreensão mútua entre os membros da equipe.

Os padrões de projeto também proporcionam flexibilidade e adaptabilidade ao sistema. Eles são projetados para fornecer soluções flexíveis, que possam ser facilmente modificadas e estendidas sem afetar outras partes do código. Isso é particularmente útil em situações em que os requisitos podem mudar ao longo do tempo ou quando é necessário lidar com a evolução contínua do software.

Também podemos ressaltar a melhoria da qualidade do código. Ao seguir padrões reconhecidos, os desenvolvedores são incentivados a adotar boas práticas de programação, como coesão, baixo acoplamento e princípios de design sólidos. Isso resulta em um código de melhor qualidade, mais legível, mais organizado e mais fácil de dar manutenção. O uso de padrões também ajuda a identificar e evitar anti-padrões, que são soluções problemáticas ou padrões mal aplicados.

Por fim, o uso de padrões de projeto contribui para a eficiência no desenvolvimento de software. Os padrões fornecem soluções pré-definidas e documentadas para problemas comuns, reduzindo o tempo e o esforço necessários para resolvê-los. Isso permite que os desenvolvedores se concentrem em aspectos mais complexos e exclusivos do sistema, acelerando o processo de desenvolvimento.

3. State e Interpreter
    3.1. State

O padrão de projeto State é um padrão comportamental que permite a um objeto alterar seu comportamento interno quando seu estado interno muda. Ele faz isso encapsulando estados individuais em classes separadas e permitindo que o objeto mude de uma classe de estado para outra à medida que seu estado interno muda. O objetivo principal do padrão State é permitir que um objeto altere seu comportamento de forma dinâmica, dependendo de seu estado interno, sem que seja necessário alterar seu código. Isso promove um design flexível e extensível, onde novos estados podem ser adicionados facilmente sem afetar o código existente.

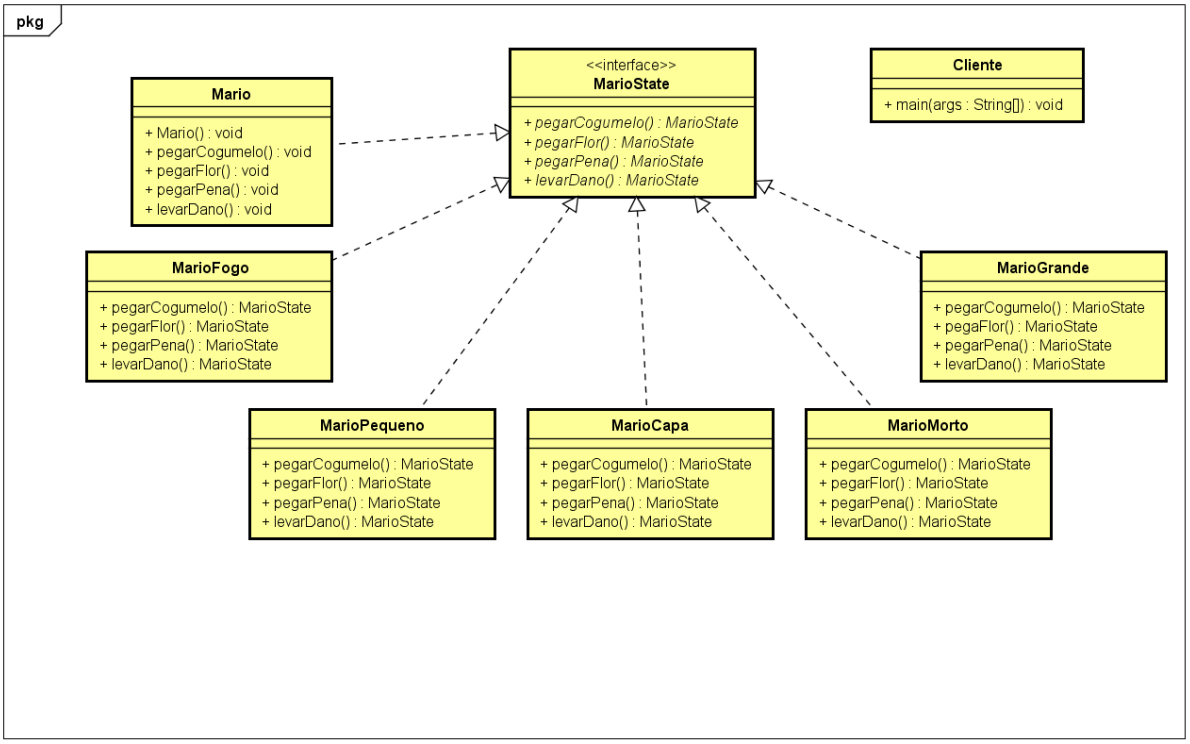
O padrão de projeto State oferece uma série de benefícios ao lidar com comportamentos condicionais complexos. Ao movê-los para classes de estado separadas, o código é simplificado e organizado de forma mais clara. Isso torna mais fácil entender e manter a lógica do objeto, evitando a necessidade de múltiplos condicionais aninhados.

Além disso, o State facilita a implementação de transições de estado complexas, fornecendo uma estrutura para controlar e gerenciar essas transições. Isso é especialmente útil quando as regras de transição dependem do estado atual do objeto, permitindo uma abordagem mais estruturada para lidar com as mudanças de comportamento.

Ajuda na manutenção do estado interno de um objeto, permitindo que ele responda a eventos ou ações de maneira diferente, dependendo do estado atual. Isso é alcançado através da encapsulação dos comportamentos específicos de cada estado em classes separadas, tornando o código mais modular e de fácil compreensão.

Uma das grandes vantagens do padrão State é sua capacidade de facilitar a adição de novos comportamentos ou estados ao objeto, sem a necessidade de modificar o código existente. Isso promove a extensibilidade do sistema, permitindo que novas classes de estado sejam criadas e incorporadas ao objeto conforme necessário.

O uso do padrão State ajuda a evitar a proliferação de condicionais excessivos no código. Em vez de usar múltiplos condicionais para verificar o estado atual do objeto, o padrão State substitui esses condicionais por implementações específicas de cada estado, reduzindo a complexidade e melhorando a legibilidade do código.



3.2. Interpreter

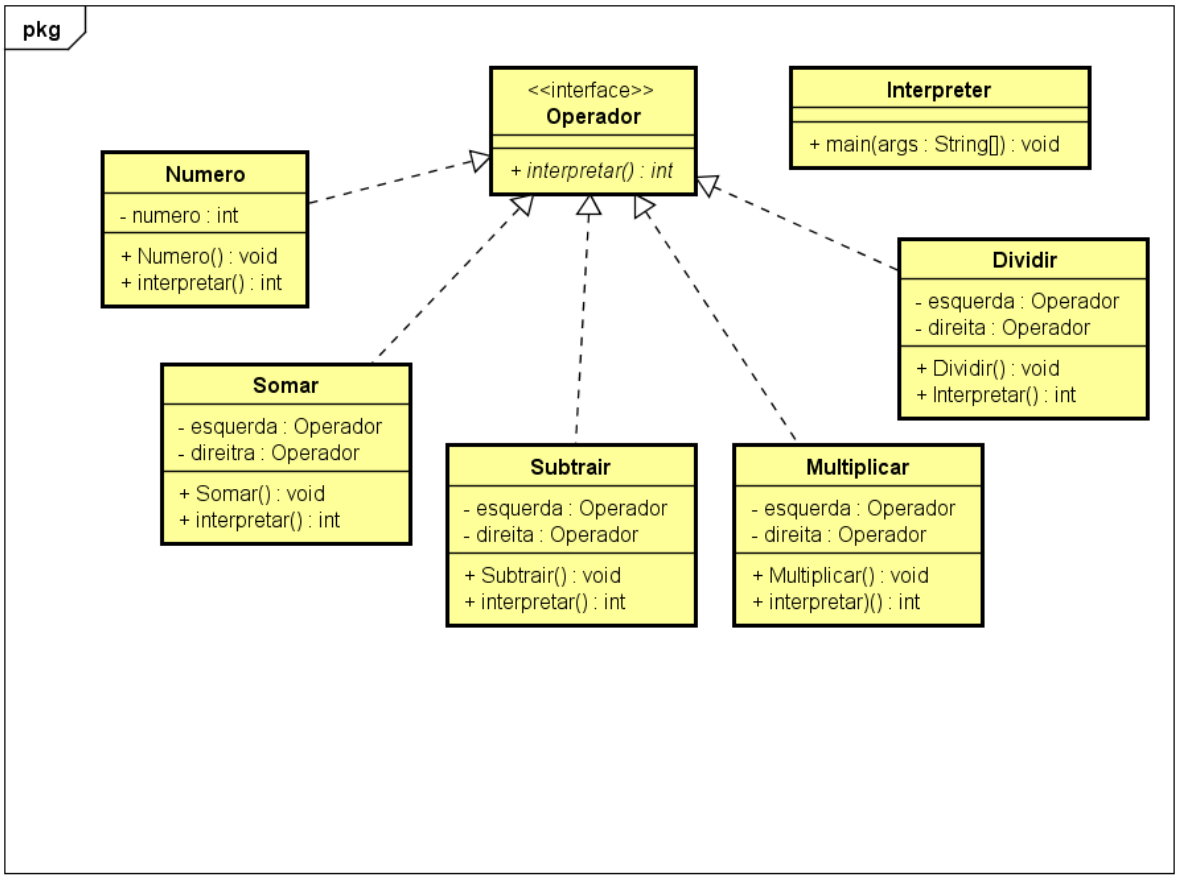
O padrão de projeto Interpreter envolve a implementação de uma interface de expressão que informa como interpretar um contexto específico. Assim como o State ele é um padrão comportamental. Ele é comumente utilizado em compiladores, analisadores ou expansões de macro. O objetivo principal desse padrão é fornecer uma maneira de avaliar expressões em uma linguagem específica, permitindo definir gramáticas e interpretar as expressões para resolver problemas particulares.

O padrão Interpreter é útil para resolver problemas em que é necessário interpretar e avaliar expressões em uma linguagem específica. Ele é aplicável quando há a necessidade de criar uma linguagem própria para solucionar um problema específico, permitindo definir gramáticas e interpretar expressões nessa linguagem. Ao separar a definição da gramática da lógica de interpretação, o padrão facilita a criação de linguagens específicas, oferecendo flexibilidade na interpretação, possibilidade de adição de novas expressões à gramática e facilidade na manutenção da linguagem interpretada.

Existem diversas aplicações para o padrão Interpreter. Ele é comumente utilizado no processamento de linguagens de programação personalizadas ou domínios específicos, onde é necessário criar uma linguagem de programação para resolver um problema

particular. O padrão também é útil em sistemas de filtragem, onde é necessário transformar dados em um padrão ou formato específico. Além disso, o padrão Interpreter pode ser empregado em sistemas de consulta e busca, permitindo interpretar e avaliar expressões de consulta de maneira personalizada.

Em resumo, o padrão de projeto Interpreter é usado para interpretar e avaliar expressões em uma linguagem específica, permitindo definir gramáticas e resolver problemas particulares. Ele é aplicável em diversas áreas, como processamento de linguagens de programação personalizadas, filtragem de dados e sistemas de consulta e busca. O padrão proporciona flexibilidade, modularidade e facilidade de manutenção ao separar a definição da gramática da lógica de interpretação.



4. Referências

https://www.treinaweb.com.br/blog/padroes-de-projeto-o-que-sao-e-o-que-resolvem

https://www.alura.com.br/artigos/design-patterns-introducao-padroes-projeto?gclid=CjwKCAjwvpCkBhB4EiwAujULMnmmOe1TmWXvN0gIkaCdp0eY-yc0mfg3NUzVC3nKYzy5J5FB9OkTTxoCUjgQAvD_BwE

https://blog.xpeducacao.com.br/padroes-de-projeto/

https://www.devmedia.com.br/conheca-os-padroes-de-projeto/957

https://gbbigardi.medium.com/arquitetura-e-desenvolvimento-de-software-parte-16-state-aa998a86277f

https://blog.matheuscastiglioni.com.br/interpreter-padroes-de-projeto-em-java/

https://www.youtube.com/watch?v=Ekq5DTUDpaQ