

Padrões e Desenho de Software

Professor:
José Luis Oliveira
Sérgio Matos

Padrões de desenho de Software:

Composite & Command

Hugo Paiva, 93195



DETI
Universidade de Aveiro
05-06-2020

Índice

1 Composite	2
1.1 Descrição	2
1.2 Problema	2
1.3 Solução	2
2 Command	3
2.1 Descrição	3
2.2 Problema	3
2.3 Solução	3
3 Bibliografia	4

1 Composite

1.1 Descrição

1.2 Problema

1.3 Solução

2 Command

2.1 Descrição

É expectável que um padrão de desenho de software comportamental, para além de fornecer boas práticas para resolver um determinado problema, foque as mesmas na forma como as classes e objetos interagem e distribuem responsabilidades. *Command*, sendo um destes tipos de padrões, foca-se exatamente nesse ponto.

O objetivo principal do padrão *Command* é encapsular um pedido/comando em um objeto, permitindo, desta forma, novas aproximações ao lidar com o mesmo. Uma destas novas aproximações é, sem dúvida, a reutilização do objeto criado. Considerando, por exemplo, um comando de televisão, este executa múltiplas ações sobre uma televisão, repetindo as ações ao longo da sua utilização. Aproximando este exemplo a uma linguagem de programação, não faria sentido, sempre que um determinado botão fosse clicado, criar um novo objeto com um determinado método ou até mesmo executar um código repetidamente, que apenas funcionaria para uma televisão em específico. Seguindo a lógica do *Command*, faz muito mais sentido instanciar uma classe que armazena a televisão recetora do pedido e que, daí em diante, apenas necessita que lhe chamem o seu método específico para executar a ação, reutilizando e uniformizando o código.

Outra grande vantagem das novas formas de lidar com pedido é a possibilidade de adicionar mais métodos ao objeto criado, complementares à simples ação pedida pelo cliente. A aproximação mais usual é a adição de um método que permite reverter o comando. Um dos exemplo mais banais desta situação é toda a gestão que um programa ou sistema operativo faz dos *inputs* de um utilizador. Ao utilizar, por exemplo, qualquer tipo de editor de texto, o sistema operativo permite ao utilizador inserir caracteres num campo mas, permite, também, reverter essas inserções (com a ajuda do atalho *ctrl+z*). Ora, isto é um exemplo visível onde faria sentido aplicar este padrão de desenho de software.

Sendo um utilizador casual de software de edição de imagem e vídeo, ao associar o *Command* à gestão que estes programas fazem das ações dos utilizadores, ficaram ainda mais claro estes conceitos. Dando o exemplo do *Adobe Photoshop*, todas as alterações à imagem podem ser revertidas múltiplas vezes, pela mesma ordem em que foram executadas, bem como voltar a executá-las, após a reversão, também na ordem correta. Ora, aproximando novamente a uma linguagem de programação, estas ações podem muito bem ser executadas com recurso ao padrão *Command*. Encapsulando os pedidos do utilizador em objetos, estes objetos podem também ser introduzidos numa lista onde constam todos os pedidos recentes. Ao utilizar os famosos atalhos para reverter e desreverter (*ctrl+z* e *ctrl+shift+z*), o programa apenas teria utilizar o método para reverter a ação e percorrer a lista na ordem inversa. O mesmo se aplica para reverter as reversões, sendo apenas necessário utilizar o método para executar a ação e percorrer novamente a lista já na ordem "normal".

Como em todos os padrões de desenho de software, a implementação mais correta é sempre discutível pois, apesar de uma solução se desviar da solução-base, esta pode ser justificada de acordo com o problema.

2.2 Problema

2.3 Solução

3 Bibliografia

- [1] https://sourcemaking.com/design_patterns/command
- [2] <https://refactoring.guru/design-patterns/command>
- [3] https://www.tutorialspoint.com/design_pattern/command_pattern.htm
- [4] <https://www.geeksforgeeks.org/command-pattern/>
- [5] <https://medium.com/better-programming/the-command-design-pattern-2313909122b5>
- [6] <https://www.nku.edu/~foxr/CSC360/Programs/MazeWithGraphics.java>