Relatório Técnico-Científico OAT1.2 - Classe Builder em Java

Tiago F. de Novais, Renato G. Souza, Haniel P. A. da Silva, Carlos E. S. Santos, Ênio F. T. Soares

Curso de Sistemas da Informação – Faculdade UNEX
Vitória da Conquista – Bahia – Brasil

Abstract: The Builder pattern is a creational design pattern that provides a flexible solution to various object construction problems in Java. This paper discusses the implementation and benefits of the Builder pattern, demonstrating how it improves code readability and maintains immutability in complex objects. The pattern separates the construction of a complex object from its representation, allowing the same construction process to create different representations.

Resumo: O padrão Builder é um padrão de projeto criacional que fornece uma solução flexível para vários problemas de construção de objetos em Java. Este artigo discute a implementação e benefícios do padrão Builder, demonstrando como ele melhora a legibilidade do código em objetos complexos. O padrão separa a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção crie diferentes representações.

1. Introdução

O padrão Builder é particularmente útil quando se precisa criar objetos com muitos parâmetros opcionais ou quando o processo de construção envolve múltiplos passos. Diferentemente dos construtores telescópicos, que se tornam difíceis de manter à medida que o número de parâmetros cresce, o Builder oferece uma abordagem mais legível e flexível.

2. Implementação do Padrão Builder

A implementação básica do padrão Builder em Java envolve a criação de uma classe estática interna que gerencia a construção do objeto. Esta abordagem garante que o objeto principal permaneça imutável após sua construção.

2.1. Uso do Builder

O uso do padrão Builder torna a criação de objetos mais clara e legível. Exemplo simples do Builder:

Order pedido = OrderBuilder.builder()

```
        .withCustomer(cliente)
        .addItem(menuItem1, 2)
        .addItem(menuItem2, 1)
        .build();
```

2.2. Estrutura

Considerando o contexto desse projeto, a estrutura do Builder final do projeto fica como segue (os comentários estão em negrito para a facilitar a leitura):

```
public class OrderBuilder {
   private Customer customer;
   private final List<OrderItem> items = new ArrayList<>();

   // Construtor público para facilitar o uso por iniciantes
   public OrderBuilder() {}

   // Define o cliente do pedido
   public OrderBuilder setCustomer(Customer customer) {
      if (customer == null) {
         throw new IllegalArgumentException("Cliente não pode ser nulo!");
      }
      this.customer = customer;
      return this;
   }

   // Adiciona um item ao pedido
   public OrderBuilder addItem(MenuItem menuItem, int quantidade) {
      if (menuItem == null) {
         throw new IllegalArgumentException("Item não pode ser nulo!");
      }
      if (quantidade <= 0) {
         throw new IllegalArgumentException("Quantidade deve ser positiva!");
      }
      items.add(new OrderItem(null, menuItem, quantidade));
      return this;
   }

   // Cria o pedido
   public Order build() {
      if (customer == null) {
         throw new IllegalStateException("Cliente deve ser definido!");
      }
```

```
        if (items.isEmpty()) {
            throw new IllegalStateException("Adicione pelo menos um item!");
        }
        Order order = new Order(customer);
        for (OrderItem item : items) {
            order.addItem(item.getMenuItem(), item.getAmount());
        }
        order.setStatus(OrderStatus.ACCEPTED);
        return order;
    }
}
```

3. Vantagens do Padrão Builder

3.1. Legibilidade do Código

O padrão Builder melhora significativamente a legibilidade do código, especialmente quando se lida com objetos que possuem muitos parâmetros. Cada método do Builder fornece um contexto claro para o parâmetro que está sendo definido.

3.2. Imutabilidade

Ao usar o padrão Builder, é possível criar objetos imutáveis, o que é uma prática recomendada em programação orientada a objetos. Objetos imutáveis são mais seguros em ambientes concorrentes e mais fáceis de raciocinar.

3.3. Validação Centralizada

O método build() pode conter lógica de validação para garantir que o objeto seja construído em um estado consistente.

4. Considerações Finais

O padrão Builder é uma ferramenta valiosa para um desenvolvedor Java, especialmente quando se trabalha com objetos complexos que requerem flexibilidade na construção. Embora introduza um pouco de complexidade adicional na forma de classes extras, os benefícios em termos de legibilidade, manutenção e segurança superam amplamente este custo.

Referências

Gamma et al. (1994) Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994) "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley.

Bloch (2008) Bloch, J. (2008) "Effective Java", Prentice Hall, 2nd edition.

Freeman et al. (2004) Freeman, E., Robson, E., Bates, B., and Sierra, K. (2004) "Head First Design Patterns", O'Reilly Media.