Trabalho – Code Smell e Refatoração

Gabriel Lima de Moraes
Carlos Alberto Andrino Júnior

Sobre o projeto

Encontramos um sistema de livraria contendo as seguintes funcionalidades:

Cadastros

- Gerenciar Livros
- Gerenciar Funcionários
- Gerenciar Administradores
- Gerenciar Editoras
- Gerenciar Clientes

Operacional

- Efetuar Venda
- Efetuar Devolução
- Ver histórico de vendas

Painel

- Zerar Sistema
- Fazer logoff

Repositório: https://github.com/jeuchaves/pdv-livraria/tree/main

Code smells identificados

Optamos por efetuar esta análise de código manualmente de primeira instância e depois fomos jogando para o chat GPT identificar code smells. A diferença foi pouca porém alguns pontos significados não encontrados manualmente foram descobertos.

Segue aqui a lista de code smells encontrados:

Controladores
Login Controlador

Categoria	**Problema**	**Solução**
Método longo	fazerLogin	Podemos fragmentar em métodos menores
Duplicação	Duplicações ao exibir mensagem de erro podem ser evitadas	Criar método global para lidar com exibição de erro
Tratamento de erro genérico	catch (Exception e)	catch (NumberFormatException e)
Método sem efeito	-	-
Código complexo	-	-
Nome de variáveis confusos	-	-

Variável desnecessária	linha 26 -> int id;	poderia ser apenas int id = Integer.parseInt(idUsuario);

Livro Controlador

Categoria	**Problema**	**Solução**
Método longo	método save e update	fragmentar o método
Duplicação	duplicação de código no save e update na questão de validação principalmente duplicação de código ao exibir mensagem de erro	generalizar essa duplicação utilizar do método global para exibição de erro
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	-	-
Código complexo	Código de validação complexo nos métodos save e update	Simplificar esses códigos
Nome de variáveis confusos	-	-
Variável desnecessária	-	-

Pessoa Controlador

Categoria	**Problema**	**Solução**
Método longo	método save e update	fragmentar o método
Duplicação	duplicação de código no save e update na questão de validação principalmente duplicação de código ao exibir mensagem de erro	generalizar essa duplicação utilizar do método global para exibição de erro
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	-	-
Código complexo	método save e update com uma complexidade de código	minimizar ao máximo e generalizar alguns pontos em métodos privados

	Complexidade na objeto tabela	Utilizar de laços de repetições para o preenchimento do objeto
Nome de variáveis confusos	-	-
Variável desnecessária	-	-

Funcionário Controlador

Categoria	**Problema**	**Solução**
Método longo	método save e update	fragmentar o método
Duplicação	duplicação de código no save e update	generalizar essa duplicação colocando em métodos privados
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	-	-
Código complexo	Objeto tabela complexo ao manipulá-lo Complexidade na validação e regras dos campos	simplificar a utilização deste objeto e validações
Nome de variáveis confusos	-	-
Variável desnecessária	-	-

Editora Controlador

Categoria	**Problema**	**Solução**
Método longo	método save e update	fragmentar o método
Duplicação	duplicação de código no save e update	generalizar essa duplicação colocando em métodos privados
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	Conversões de número parseInt feitas sem tratamento adequado	Corrigir esses tratamento
Código complexo	Complexidade nos métodos delete e update Objeto tabela complexo ao	simplificar a utilização deste métodos, objeto e validações

	manipulá-lo Complexidade na validação e regras dos campos	
Nome de variáveis confusos	-	-
Variável desnecessária	-	-

Criamos um novo método na classe EditoraDAO.java para verificar se o CNPJ já consta na base de dados pois tratava-se de uma verificação complexa.

Devolução Controlador

Categoria	**Problema**	**Solução**
Método longo	método removerItem e abrirDevolucao	fragmentar o método
Duplicação	Duplicação no método para exibir mensagem de erro	Utilizar da Classe criada globalmente TratamentoErro
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	-	-
Código complexo	Complexidade no método removerItem Complexidade na validação e regras dos campos	simplificar a utilização deste métodos e validações
Nome de variáveis confusos	-	-
Separação de responsabilidade	método removerItem há várias responsabilidades (validação, criação de item, atualização de estoque, etc.)	Dividir essas responsabilidades em métodos separados

Venda Controlador

Categoria	**Problema**	**Solução**
Método longo	Método addItem, finalizarVenda	Fragmentar método
Duplicação	no método addItem(), há uma possível exceção ao tentar converter strings em inteiros.	Lidar com essa exceção para fornecer feedback adequado ao usuário. Utilizar da classe global

	Exibição de erros	TratamentoErro
Tratamento de erro genérico	-	-
Método sem efeito	-	-
Código complexo	-	-
Redundância	Em finalizarVenda(), você verifica se o valor é maior que o preço total e, em seguida, exibe uma mensagem	Simplificado usando um único fluxo condicional.
Separação de responsabilidade	O método finalizarVenda() possui várias responsabilidades, como verificar o valor pago, calcular o troco, exibir mensagens etc.	Dividir essas responsabilidades em métodos auxiliares para tornar o método principal mais conciso.

Data Access Object
Funcionário, Livro, Editora, Item, Pedido, Pessoa

Categoria	**Problema**	**Solução**
Método longo	-	-
Duplicação	duplicação de código quando vai fechar recursos	criação de um método closeResources em util
Tratamento de erro genérico	Existe bastante tratamento de erro genérico	Mapear alguns pontos e corrigi los
Método sem efeito	-	-
Código complexo	-	-
Nome de variáveis confusos	-	-
Variável desnecessária	-	-

Todos esses arquivos de Data Access Object receberam apenas modificações do método closeResources e algumas correções de tratamentos de erro genérico.

Relatório Final

a. Quais os code smells mais comuns encontrados?
Os code smells mais comuns encontrados nos códigos que você forneceu foram:

Método Longo: Métodos extensos que realizam muitas ações e possuem muitas linhas de código, tornando-os difíceis de entender e manter.

Código Duplicado: Trechos de código idênticos ou muito semelhantes que estão repetidos em várias partes do código, o que aumenta a probabilidade de erros e torna a manutenção mais difícil.

Obsessão Primitiva: Uso excessivo de tipos primitivos (como int, string) em vez de criar classes que encapsulam esses tipos, resultando em código difícil de entender e que não reflete adequadamente a semântica do domínio.

Tratamento de erro: Os tratamentos de erros estavam sendo feitos todos de forma genérica o que poderia ocasionar um problema futuro, além disso as mensagens de erros estavam sendo instanciadas sempre que necessário de forma manual, fazendo o código ficar grande e complexo.

Separação de responsabilidade: Por incrível que pareça o autor do código separou bem as responsabilidades, foram poucos os casos deste code smell.

b. Foi difícil a refatoração?

Não foi tão difícil quanto nós imaginamos. Alguns code smells são mais fáceis de corrigir do que outros. Por exemplo, remover código duplicado ou dividir um método longo em métodos menores pode ser relativamente fácil, enquanto problemas de responsabilidades e redundância exigem uma análise mais profunda e uma mudança mais significativa na estrutura do código.

c. Considera que houve melhoria no código, explique.

Sim, definitivamente houve melhoria no código após a refatoração. Os code smells identificados são indicadores de más práticas de programação que podem resultar em código difícil de entender, manter e modificar.

Ao corrigir esses code smells, o código se torna mais legível, modular, coeso e menos acoplado. A utilização de boas práticas, como a criação de métodos mais curtos e significativos, redução de duplicação, encapsulamento de tipos primitivos, entre outros, resulta em um código mais limpo e mais fácil de manter. Isso também pode levar a uma redução de erros e a um desenvolvimento mais eficiente no futuro.

d. Técnica e processo de refatoração utilizada.

Optamos por fazer em fases…

Primeiro, ambos os membros identificamos e listamos alguns codes smells manualmente.

Depois juntamos as ideias e fizemos mais uma leitura do código em grupo.

Para a solução dos problemas optamos por utilizar o chat GPT e nisso ele apontou alguns outros pontos de code smells interessantes.