UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ

RELATÓRIO – TRABALHO FINAL QUALIDADE DE SOFTWARE
SisCom

Equipe:

Francisco Tierry Barros Oliveira

Davi Teixeira Silva

Professora:

Carla Ilane Moreira Bezerra

QUIXADÁ

Agosto, 2021

1 DESCRIÇÃO DO PROJETO

O SisCom é um sistema comercial de compra e venda, desenvolvido em Java OO para desktop, em que basicamente temos subdividido os clientes, fornecedores, vendedores e produtos, onde cada um possui um crud básico de cadastro, listagem e remoção em tempo de execução, assim o administrador do sistema pode cadastrar um produto, após isso o fornecedor pode adquirir esse produto, repassando-o ao vendedor, e por fim o vendedor realizar a venda do determinado produto a um usuário cadastrado no sistema. É um projeto aberto o qual foi desenvolvido por um estudante de Belo Horizonte, mas ainda se encontra incompleto, e seu código fonte está disponível no link abaixo.

Link do projeto: https://github.com/TierryBr/Qualidade-SmellsAndRefactoring (Origem: https://github.com/breitembach/SisCom-Java)

Assim o projeto tem algumas características e informações como:

Cadastro:

Nome; Telefone; E-mail; CNPJ; Limite de Crédito.

Consulta:

Pesquisa pelo CPF; Funcionalidade de deletar um cliente; Funcionalidade de listar todos.

Na aba de Fornecedor, temos quatro funcionalidades, o cadastro de fornecedor, a consulta de fornecedor, uma aba de compras e uma aba de estatísticas, assim cada uma delas tem as seguintes informações a serem cadastradas:

Cadastro:

Nome; Telefone; E-mail; CNPJ; Contato

Consulta:

Pesquisa pelo CNPJ; Funcionalidade de deletar um cliente; Funcionalidade de listar todos.

Compra:

Aqui temos a funcionalidade de pesquisar um produto pelo cnpj do fornecedor, exibindo assim uma tabela com todos os produtos cadastrados naquele cnpj; Temos a opção de pesquisar pelo nome do produto; Uma área que mostra a lista de compras daquele fornecedor.

Na aba de Vendedor, temos quatro funcionalidades, o cadastro de vendedor, a consulta de vendedor, uma aba de vendas e uma aba de estatísticas de vendas, assim cada uma delas tem as seguintes informações a serem cadastradas:

Cadastro:

Nome; Telefone; E-mail; CPF; Meta Mensal

Consulta:

Pesquisa pelo CPF; Funcionalidade de deletar um vendedor; Funcionalidade de listar todos.

Venda:

Aqui temos a funcionalidade de pesquisar um vendedor pelo cpf, exibindo assim suas informações; Temos a opção de pesquisar pelo nome do produto; Uma área que mostra a lista de vendas daquele vendedor.

Por fim, na aba de Produtos, temos duas funcionalidades, o cadastro de produtos e a consulta de produtos, tendo as seguintes informações a serem cadastradas:

Cadastro:

Nome; Preço unitário; Estoque; Estoque Mínimo

Consulta:

Pesquisa pelo nome do produto; Funcionalidade de listar todos, tanto por ordem alfabética ou pelo preço mínimo.

Também temos algo que é padrão para todos, que é a:

Estatística de Compras:

Uma funcionalidade que mostra a lista de compras de todos os clientes, por um período estabelecido; Uma funcionalidade que mostra todas as estatísticas por um período fornecido.

A seguir, está uma tabela que mostra algumas propriedades do projeto:

Tabela 1 – Características do Projeto

Projeto	LOC	# de classes	# de releases
SisCom	5676	93	1

2 AVALIAÇÃO DO PROJETO

2.1 Medição 1 – Antes de refatorar o projeto

A tabela a seguir (Tabela 2) mostra a medição dos atributos de qualidade na ferramenta Understand antes de refatorar o projeto, o qual destaca a coesão, complexidade, herança, acoplamento e tamanho em cada classe diferente do código, e por fim mostra o somatório dessas métricas, como pode ser observada mais em detalhes na tabela 3.

Tabela 2 – Medição dos atributos antes de refatorar o projeto.

Pacotes/classe	COESÃO	COMPLEXIDADE				Herança			ACOPLAMENTO	TAMANHO			
"controller"	LCOM	ACC	SCC	EVG	Nesting	DIT	NOC	Base Clas	CBO	LOC	CLOC	NIM	CDL
Comercial	89	2	87	11	4	1	0	2	12	5676	1123	358	93
"error"													
SisComException	75	1	8	1	0	2	0	1	0				
"model"													
Cliente	70	1	11	1	1	2	0	2	2				
Compra	72	1	11	1	0	1	0	2	2				
Estatistica	71	1	12	1	0	1	0	2	0				
Fornecedor	70	1	11	1	1	2	0	2	2				
ItemCompra	62	1	8	1	0	1	0	2	1				
ItemVenda	66	1	9	1	0	1	0	2	1				
Pessoa	75	1	14	1	0	1	3	3	2				
Produto	77	1	17	1	0	1	0	3	0				
Venda	78	1	15	1	0	1	0	2	4				
Vendedor	73	1	13	1	1	2	0	2	3				
"utilitarios"													
Console	0	2	13	3	3	1	0	1	0				
CustomTransferHandler	100	3	16	5	3	2	0	1	3				
InterfaceUtil	0	1	7	1	0	1	0	1	0				
LtpLib	99	3	145	11	4	1	0	1	1				
"view"													
Colors	0	0	0	0	0	1	0	1	0				
InterfaceBase	75	1	3	1	1	2	0	1	6				
InterfaceComercial	76	3	94	1	2	1	0	1	12				
InternalFrameVendaCliente	58	3	10	1	4	2	0	1	11				
PanelClienteCenter	81	3	15	1	2	2	0	1	7				
PanelFornecedorCenter	89	3	33	1	5	2	0	1	11				
PanelProdutosCenter	56	2	4	1	1	2	0	1	5				
PanelVendedorCenter	88	3	26	1	2	2	0	1	9				
Total	1600	40	582	49	34	35	3	37	94				
Total All Metrics	1600		705				75						

Tabela 3 – Resultado da medição antes das refatorações.

Atributos Internos de Qualidade	Métricas	Valor da Métrica	Total do Atributo
Coesão	LCOM	1600	1600
Complexidade	ACC	40	705
	SCC	582	

	EVG	49	
	Nesting	34	
Herança	DIT	35	75
	NOC	3	
	Base Classes	37	
Acoplamento	CBO	94	94
Tamanho	LOC	5676	7250
	CLOC	1123	
	NIM	358	
	CDL	93	

2.2 Detecção dos Code Smells

Ao todo foram encontrados 55 Code Smells o qual foram detectados pela ferramenta JSPirit. Como vamos refatorar 40 smells, descartamos o Brain Method, Brain Class e Data Class, restando 5 tipos para se trabalhar e dentro da margem dos 40 smells. A tabela 4 abaixo mostra o total de ocorrências para cada tipo de smells detectados no código.

Tabela 4 – Code smells do projeto.

Nome do Code Smell	**Quantidade**
Feature Envy	24
Dispersed Coupling	13
Shotgun Surgery	7
Intensive Coupling	5
God Class	3
Brain Method	1

Brain Class	1
Data Class	1

2.3 Medição 2 – Após Refatorar Code Smell Feature Envy

Como visto, o nosso código possui em um maior número o Feature Envy, então dos 24 encontrados, foram refatorados 20 deles, assim usamos a técnica do Move Method e Extract Method para aplicar a refatoração, já que são as soluções mais adequadas para esse tipo de smell, que consiste em basicamente mover o método para outra classe que faça sentido, ou fazer um split no método até que ele seja excluído totalmente durante a detecção do JSPirit, embora tenha gerado com isso 2 Refused Parent Bequest, onde conseguimos resolver apenas um deles. Logo após concluir essas 20 refatorações, medimos o projeto novamente na ferramenta Understand e chegamos aos resultados mostrados na tabela 5.

Como observado na tabela 5, é possível perceber que houve um aumento em quase todos os atributos de qualidade, embora isso não signifique uma piora na qualidade geral do projeto. Como notado, tivemos aumento em todas as métricas que envolvem Coesão e Complexidade, como também o Acoplamento, isso se deve ao fato de que usamos o Move Method e Extract Method, que foram os métodos de refatorações mais recomendados para esse caso, em basicamente movemos alguns métodos e outros separamos em dois para uma melhor compreensão, enquanto isso, a Herança permaneceu estável. Por outro lado, podemos notar uma certa melhora no Tamanho, algo que chega a ser um pouco surpreendente, já que teve melhora em basicamente todas as métricas, acreditamos que tenha sido devido a uma melhor distribuição do código ao refatorar-lo.

Tabela 5 – Resultado após refatorar feature envy.

Atributos Internos de Qualidade	**Métricas**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	1640	1640
Complexidade	ACC	41	714
	SCC	583	
	EVG	53	

	Nesting	37	
Herança	DIT	35	75
	NOC	3	
	Base Classes	37	
Acoplamento	CBO	100	100
Tamanho	LOC	5663	7229
	CLOC	1123	
	NIM	352	
	CDL	91	

Legenda: maior, igual, menor

2.4 Medição 3 – Após refatorar Code Smell Dispersed Coupling

Dispersed Coupling era o segundo smell mais detectado no programa, assim resolvemos refatorar, embora contasse com 13, a nossa equipe conseguiu resolver 5 deles completamente, pois o restante era em classes de interface, que envolvia chamadas como JOptionPane por exemplo, as quais não fazia sentido refatorar, pois "quebraria" a execução do projeto completamente.

Basicamente nossa estratégia de refatoração foi o Extract Method e Move Method, seja criando métodos que faziam mais sentido, ou movendo eles para respectivas classes que faziam sentido. Com isso tivemos o resultado que pode ser visto na tabela 6, em que nela é possível observar que houve uma maior estabilidade das métricas, ou seja, boa parte delas se manteve igual, como é o caso da Herança e Acoplamento (DIT, NOC, BC, CBO), e também uma pequena melhoria na Complexidade, como nas métricas ACC e Nesting, essa melhoria pode ser explicado pelo motivo de termos criados métodos específicos e extraindo para suas respectivas classes, assim evitando métodos muitos longos e de alta complexidade, o restante teve pequenos aumentos, cerca de 30,8%, contando métricas como LCOM, SCC, LOC e NIM.

Tabela 6 – Resultado após refatorar dispersed coupling.

Atributos Internos de Qualidade	Métricas	Valor da Métrica	Total do Atributo
Coesão	LCOM	1655	1655
Complexidade	ACC	37	720
	SCC	594	
	EVG	53	
	Nesting	36	
Herança	DIT	35	75
	NOC	3	
	Base Classes	37	
Acoplamento	CBO	100	100
Tamanho	LOC	5698	7266
	CLOC	1115	
	NIM	362	
	CDL	91	

Legenda: maior, igual, menor

2.5 Medição 4 – Após refatorar Code Smell Intensive Coupling

Também sendo um dos que mais afetava o projeto, em que continha 5 smells inicialmente, onde conseguimos refatorar 4 deles completamente, restando apenas 1 que era relacionado a interface Swing, a qual decidimos não modificar para não atrapalhar a execução do programa.

A estratégia utilizada para solucionar essas ocorrências foi a mesma utilizada no Dispersed Coupling, com isso usamos o Extract Method e Move Method também. Com isso tivemos o resultado que é mostrado na tabela 7 abaixo, onde foi notado no geral uma melhora em nada, todos com poucas variações ou nenhuma, dando destaque apenas para a métrica de

Nesting da Complexidade e Coesão, que aumentou em relação à tabela anterior, acreditamos que isso tenha ocorrido devido a simplicidade e quantidade de métodos com esse problema, pois dois deles eram na main e foram bastante simples de resolver, assim não afetando os atributos internos de qualidade.

Tabela 7 – Resultado após refatorar intensive coupling.

Atributos Internos de Qualidade	**Métricas**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	1670	1670
Complexidade	ACC	36	733
	SCC	607	
	EVG	53	
	Nesting	37	
Herança	DIT	35	75
	NOC	3	
	Base Classes	37	
Acoplamento	CBO	100	100
Tamanho	LOC	5744	7323
	CLOC	1114	
	NIM	374	
	CDL	91	

Legenda: maior, igual, menor

2.6 Medição 5 – Após refatorar Code Smell Brain Method / Brain Class

Inicialmente ficamos confuso com o Brain Method em apenas um método específico, porém notamos que ele estava meio que incompleto levando em consideração a implementação, assim foi muito difícil achar uma solução, pois era uma parte de código muito complexo e não existe um método de refatoração adequado para esse tipo de smell, com isso nossa solução proposta foi comentar completamente parte desse método já que ele não está sendo usado e muito menos afetando o funcionamento do código, após realizar essa ação notamos que tanto o Brain Method com Brain Class foi solucionado.

Como podemos observar na tabela 8, tivemos cerca de 77% de igualdade em relação à tabela anterior, isso ocorre porque mudamos apenas um único método, em que ele afetou apenas a Complexidade (SCC) e Tamanho (LOC), tendo uma melhora significativa como um todo, chegando a ser também cerca de 15,5% em relação a tabela 7.

Tabela 8 – Resultado após refatorar Brain Method / Brain Class.

Atributos Internos de Qualidade	**Métricas**	**Valor da Métrica**	**Total do Atributo**
Coesão	LCOM	1670	1670
Complexidade	ACC	36	724
	SCC	598	
	EVG	53	
	Nesting	37	
Herança	DIT	35	75
	NOC	3	
	Base Classes	37	
Acoplamento	CBO	100	100
Tamanho	LOC	5720	7324
	CLOC	1139	
	NIM	374	
	CDL	91	

Legenda: maior, igual, menor

2.7 Medição 6 – Após a refatoração de todos os code smells do projeto

Após refatorar os 32 Smells, como citado acima, foi notado que também foi refatorado de forma automática 1 Shotgun Surgery, junto com a geração automática de Refused Parent Bequest, não temos um motivo exato do porque isso ocorreu, mas acreditamos que tenha sido devido ao Extract Method, que fez remover certas dependências que a classe Comercial tinha. Resumindo de forma geral como mostra a tabela 9, foi notado que todos os atributos de qualidade aumentaram de forma significante, como na Coesão e Complexidade, porém podemos notar algumas pequenas melhorias no atributo de Complexidade (ACC) e de Tamanho (CDL), que pode ter como justificativa o Extract Method também, pois com ele conseguimos gerar métodos menores e menos complexos, assim houve melhoria no número de classes.

Considerando as 13 métricas de qualidade, podemos perceber que houve uma diminuição em +/- 15% dos casos, em 23% dos casos as métricas ficaram iguais, e em 62% dos casos houve um aumento, sendo 31% desse aumento apenas na Coesão e Complexidade.

Tabela 9 – Resultado da medição após todas as refatorações.

Atributo Interno de Qualidade	Feature Envy	Dispersed Coupling	Shotgun Surgery	Intensive Coupling	God Class	Brain Method	Brain Class	Data Class	Mudança Final
Coesão	LCOM	LCOM	LCOM	LCOM	LCOM	LCOM	LCOM	LCOM	+70
Complexidade	ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	-4
	SCC	SCC	SCC	SCC	SCC	SCC	SCC	SCC	+16
	EVG	EVG	EVG	EVG	EVG	EVG	EVG	EVG	+4
	Nesting	Nesting	Nesting	Nesting	Nesting	Nesting	Nesting	Nesting	+3
Herança	DIT	DIT	DIT	DIT	DIT	DIT	DIT	DIT	0
	NOC	NOC	NOC	NOC	NOC	NOC	NOC	NOC	0
	Base Classes	Base Classes	Base Classes	Base Classes	Base Classes	Base Classes	Base Classes	Base Classes	0
Acoplamento	CBO	CBO	CBO	CBO	CBO	CBO	CBO	CBO	+6
Tamanho	LOC	LOC	LOC	LOC	LOC	LOC	LOC	LOC	+44

	CLOC	CLOC	CLOC	CLOC	CLOC	CLOC	CLOC	CLOC	+16
	NIM	NIM	NIM	NIM	NIM	NIM	NIM	NIM	+16
	CDL	CDL	CDL	CDL	CDL	CDL	CDL	CDL	-2