THE DEVELOPER'S CONFERENCE

Design de Código

Code Smell, como evitar e manter a qualidade no seu código.

Lorena Dutra

Mestre em Ciência da Computação

Sobre o que vamos conversar...

THE DEVELOPER'S CONFERENCE

- Introdução
- O que é code smell?
- Refatoração
- Conhecendo e identificando os smells
- Soluções
- Ferramenta para métricas de qualidade de código.
- Finalizando

Projetos de desenvolvimento de software

THE DEVELOPER'S CONFERENCE

Custo

Manutenção

Facilitar

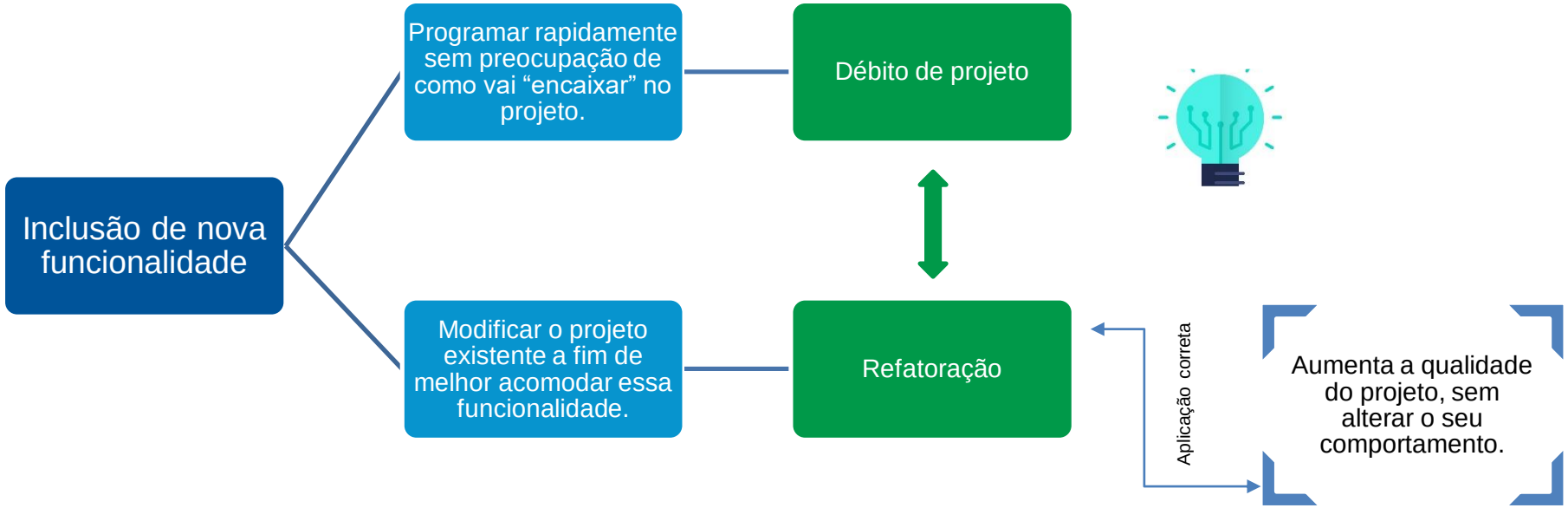
Manutenibilidade e legibilidade, melhorando a qualidade interna.

Boas práticas

Fundamentos de código limpo e refatoração

Software mais fácil de ser mantido e evoluído.

Projetos de desenvolvimento de software



Código limpo

THE DEVELOPER'S CONFERENCE

Prática que visa a escrita de código legível, funcional, pequeno, simples, fácil de entender e manter.

Facilmente acessível a outros, com uma clara intenção, sem ambiguidades.

Se um código está difícil de ser entendido, ele deve ser refatorado.

Normalmente comentados

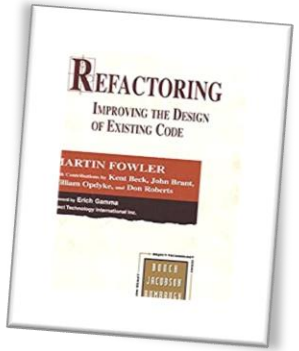
Facilita a adição de novos trechos de código.

O que é Code Smell ?

THE DEVELOPER'S CONFERENCE

- É qualquer característica no código que possivelmente indica um problema mais crítico.

- O termo foi popularizado no final da década de 1990, e o uso do termo aumentou depois que ele foi apresentado no livro Refatoração: Melhorando o Design do Código Existente, de Martin Fowler.

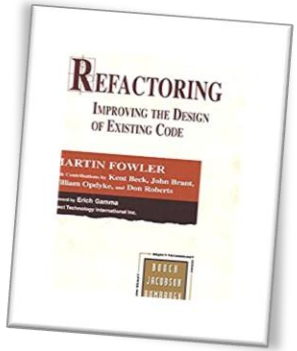


O que é Code Smell ?

THE DEVELOPER'S CONFERENCE

- É qualquer característica no código que possivelmente indica um problema mais crítico.
- O termo foi po ularizado e o uso do termo aumentou depois que atoração: Melhorando o Design do Fowler.

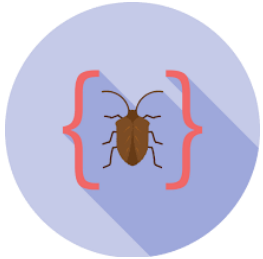
São um indicador de um problema, e não o próprio problema.



O que é Code Smell ?

THE DEVELOPER'S CONFERENCE

- Não são tecnicamente incorretos e não impedem o funcionamento do programa.
- Em vez disso, indicam fraquezas que podem estar atrasando o desenvolvimento ou aumentando o risco de bugs ou falhas no futuro.



O que é Code Smell ?

THE DEVELOPER'S CONFERENCE

- Os melhores smells são fáceis de detectar e na maioria das vezes levam a problemas realmente significativos.
- Frequentemente, refatorações simples podem resolver o problema.
- É fácil para pessoas sem muita experiência identificá-los, mesmo que não tenham conhecimento suficiente para avaliar se há um problema real ou corrigi-los.

Refatoração

THE DEVELOPER'S CONFERENCE

- É o processo de alterar um código de tal forma que não altere o comportamento do código, mas melhora sua estrutura interna.

Fazer refatoração sem testes é como pular de um avião sem paraquedas.

Bons testes são essenciais para um boa refatoração, pois eles lhe dão segurança ao fazer alguma mudança mais tarde.

Conhecendo e identificando os smells

- Método Longo
- Classe Grande
- Obsessão Primitiva
- Lista longa de Parâmetros
- Agrupamentos de dados

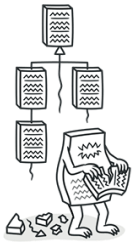
Infladores

Acopladores

- Feature Invejosa
- Intimidade inapropriada
- Cadeias de mensagens
- Middle Man

Code Smell

Abusadores de orientação a objetos



- Alternar declarações
- Campo temporário
- Requisito recusado
- Classes alternativas com diferentes interfaces

Preventores de mudança

- Mudança divergente
- Cirurgia de Shotgun
- Hierarquias de herança paralela

Dispensáveis

- Classe preguiçosa
- Classe de Dados
- Código Duplicado
- Código morto
- Generalidade Especulativa

Infladores

- São códigos, métodos e classes que aumentaram para proporções gigantescas com as quais são difíceis de trabalhar.

- Não surgem imediatamente, mas acumulam-se ao longo do tempo à medida que o código evolui (e especialmente quando ninguém se esforça para erradicá-los).

Infladores - Métodos longos

THE DEVELOPER'S CONFERENCE

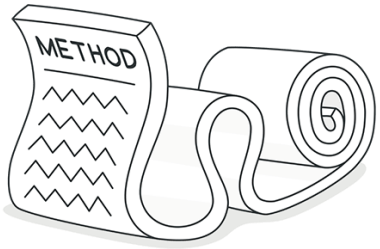
- Como é mais fácil escrever código do que lê-lo, esse smell permanece despercebido até que o método se transforme em algo gigante.

Tratamento

Se sentir a necessidade de comentar algo dentro de um método, você deve pegar esse código e colocá-lo em um novo método. Mesmo uma única linha pode e deve ser dividida em um método separado.

E se o método tiver um nome descritivo, ninguém precisará examinar o código para ver o que ele faz.

- *Oferecem o esconderijo perfeito para códigos duplicados indesejados.*



Infladores - Classes Grandes

THE DEVELOPER'S CONFERENCE

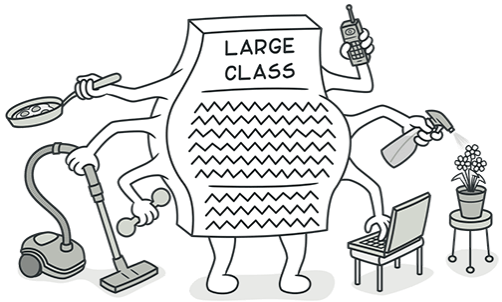
- Uma classe contém muitos campos / métodos / linhas de código.

Tratamento

Extrair classes em componentes separados para atribuir responsabilidades objetivas a cada uma.

A divisão de grandes classes em partes evita a duplicação de código e funcionalidade.

- *Os desenvolvedores acham menos exigente colocar um novo recurso em uma classe existente do que criar uma nova classe para o recurso.*



Infladores - Obsessão Primitiva

THE DEVELOPER'S CONFERENCE

- É a tendência por usar sempre tipos primitivos ao invés de classes.

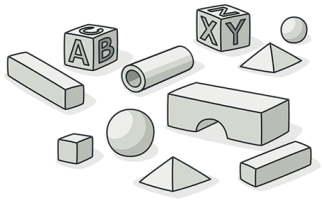
Tratamento

Agrupar logicamente alguns deles em sua própria classe. Melhor ainda, mova o comportamento associado a esses dados para a classe

```
1 public function salvar($valor,$nomeProduto,$nomeCliente,
2     $valorDesconto,$valorComissao,$telCliente,
3     $enderecoCliente,$nomeVendedor,$idProduto,$quantidade){
4
5 }
```

```
1 public function salvar(Venda $Venda,Cliente $Cliente){
2
3 }
```

- *O código se torna mais flexível graças ao uso de objetos em vez de primitivos.*
- *Melhor compreensão e organização do código.*



Infladores - Lista longa de parâmetros

THE DEVELOPER'S CONFERENCE

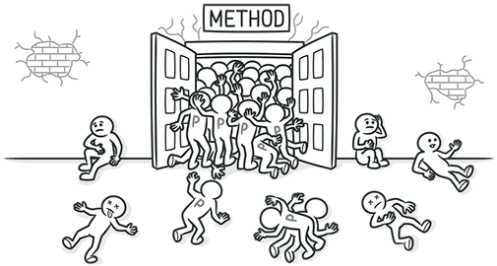
- Mais de três ou quatro parâmetros para um método.

Tratamento

Em vez de uma longa lista de parâmetros, um método pode usar os dados de seu próprio objeto.

Se o objeto atual não contiver todos os dados necessários, outro objeto (que obterá os dados necessários) poderá ser transmitido como parâmetro de método.

- *Código mais legível e mais curto.*
- *Quando ignorar - Não se desfaça de parâmetros se isso causar dependência indesejada entre classes.*



Infladores - Agrupamento de dados

THE DEVELOPER'S CONFERENCE

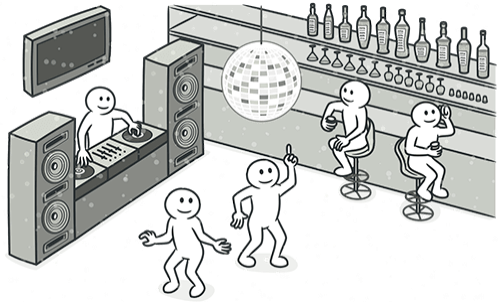
- Às vezes, partes diferentes do código contêm grupos idênticos de variáveis (como parâmetros para conectar-se a um banco de dados).

Tratamento

As operações com dados específicos são reunidas em um único local, em vez de aleatoriamente em todo o código.

Esses grupos devem ser transformados em suas próprias classes.

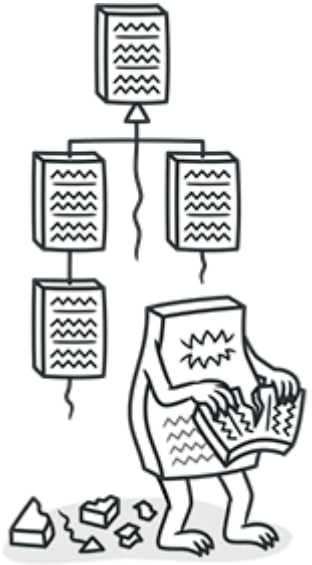
- *Melhora a compreensão e organização do código.*
- *Reduz o tamanho do código.*



Abusadores de orientação a objetos

THE DEVELOPER'S CONFERENCE

- São aplicações incompletas ou incorretas dos princípios de programação orientada a objetos.



Abusadores de orientação a objetos - Alternar declarações

THE DEVELOPER'S CONFERENCE

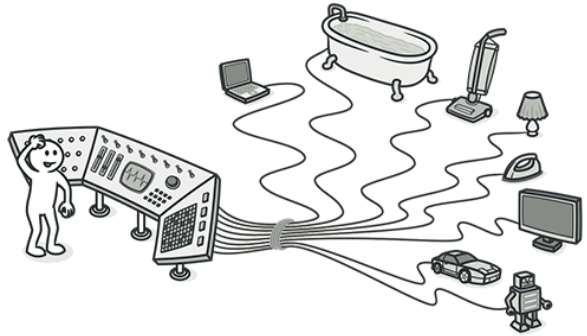
- Um switch complexo ou uma sequencia de instruções if

Tratamento

Geralmente, o código de um único switch pode ser espalhado em diferentes locais do programa.

Isolar e colocar em uma classe específica.

- *Organização de código aprimorada.*



Abusadores de orientação a objetos - Campo temporário

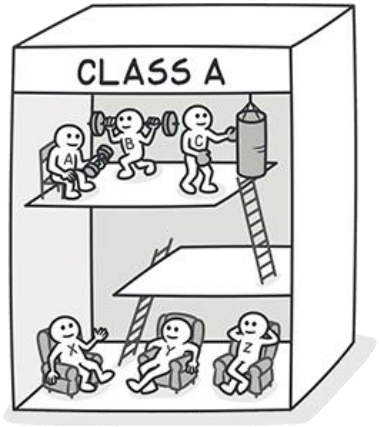
THE DEVELOPER'S CONFERENCE

- Os campos temporários obtêm seus valores somente sob determinadas circunstâncias. Fora dessas circunstâncias, eles estão vazios.

Tratamento

Os campos temporários e todo o código operando neles podem ser colocados em uma classe separada.

- *Melhor clareza e organização do código.*



Abusadores de orientação a objetos – Requisito recusado

THE DEVELOPER'S CONFERENCE

- Se uma subclasse usa apenas alguns dos métodos e propriedades herdados de seus pais, a hierarquia é desequilibrada.

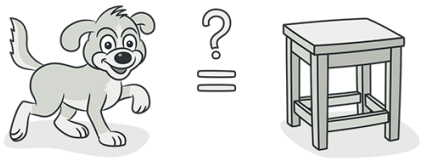
Tratamento

Livre-se dos campos e métodos desnecessários da subclasse.

Extraia todos os campos e métodos necessários pela subclasse da classe pai.

Coloque-os em uma nova subclasse e defina as duas classes para herdar

- *Melhora a clareza e organização do código.*
- *Você não precisará mais se perguntar por que a classe "cachorro" é herdada da classe "cadeira"*



Abusadores de orientação a objetos - Classes alternativas com diferentes interfaces

THE DEVELOPER'S CONFERENCE

- O desenvolvedor que criou uma das classes provavelmente não sabia que uma classe funcionalmente equivalente já existia.

Tratamento

Deixar as interfaces iguais.

Depois de determinar qual método de tratamento usar e implementá-lo, você poderá excluir uma das classes.

- *Você se livra de códigos duplicados desnecessários, tornando o código resultante menos volumoso.*
- *O código se torna mais legível e compreensível.*

Preventores de mudança

THE DEVELOPER'S CONFERENCE

- Se você precisar alterar algo em um local do seu código, também precisará fazer muitas alterações em outros locais.
- O desenvolvimento se torna muito mais complicado e caro.

Preventores de mudança - Mudança divergente

THE DEVELOPER'S CONFERENCE

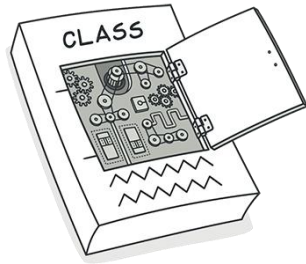
- Quando muitas mudanças são feitas em uma única classe.
- Ter que alterar muitos métodos não relacionados ao fazer alterações em uma classe. Por exemplo, ao adicionar um novo tipo de produto, é necessário alterar os métodos para localizar, exibir e solicitar produtos.

Tratamento

Dividir o comportamento da classe.

Se diferentes classes tiverem o mesmo comportamento, convém combinar as classes por herança.

- *Melhora a organização do código.*
- *Reduz a duplicação de código.*



Preventores de mudança – Cirurgia de Shotgan

THE DEVELOPER'S CONFERENCE

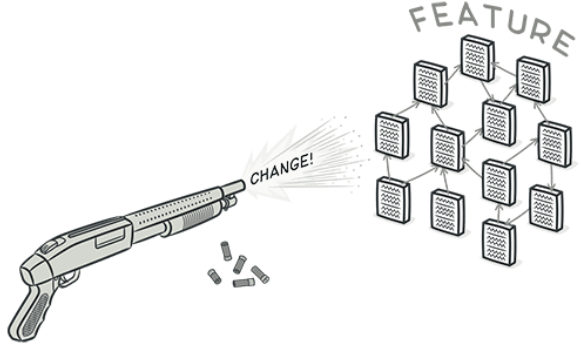
- Quando uma única alteração é feita em várias classes simultaneamente.

Tratamento

Mover comportamentos de classe existentes em uma única classe.

Se não houver uma classe apropriada para isso, crie uma nova.

- *Melhor organização.*
- *Menos duplicação de código.*
- *Manutenção mais fácil.*



Preventores de mudança - Hierarquias de herança paralela

THE DEVELOPER'S CONFERENCE

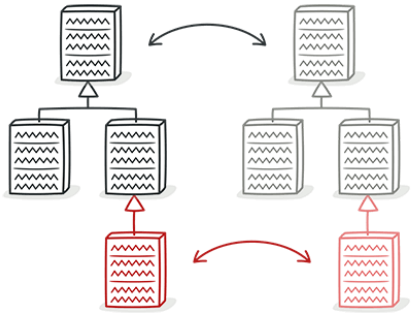
- Sempre que criar uma subclasse para uma classe, vai precisar criar uma subclasse para outra classe.

Tratamento

Desduplicar hierarquias de classes paralelas em duas etapas.

Primeiro, faça com que instâncias de uma hierarquia se refiram a instâncias de outra hierarquia. Em seguida, remova a hierarquia na classe referida.

- *Reduz a duplicação de código.*
- *Pode melhorar a organização do código.*



Dispensáveis

THE DEVELOPER'S CONFERENCE

- Algo desnecessário cuja ausência tornaria o código mais limpo, mais eficiente e mais fácil de entender.

Dispensáveis - Comentários

THE DEVELOPER'S CONFERENCE

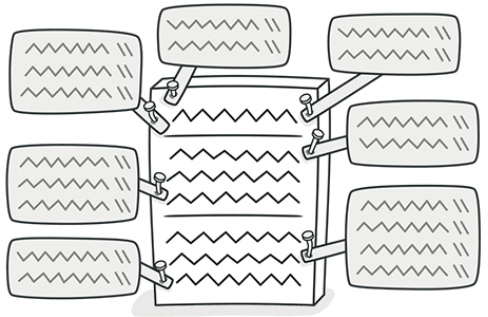
- São criados com a melhor das intenções, quando o autor percebe que seu código não é intuitivo ou óbvio.

Tratamento

Se um comentário pretende explicar uma expressão complexa, a expressão deve ser dividida em subexpressões compreensíveis.

Se um comentário explica uma seção do código, esta seção pode ser transformada em um método. O nome do novo método pode ser obtido do próprio texto do comentário.

- O código se torna mais intuitivo e óbvio.



Dispensáveis - Código duplicado

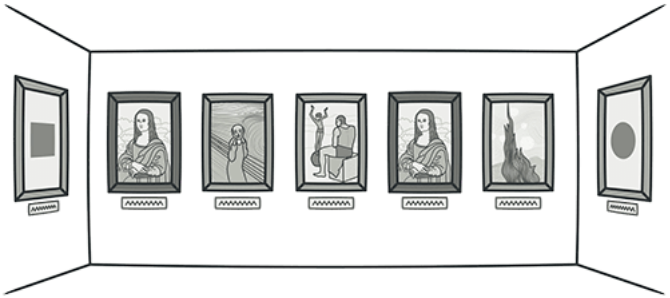
THE DEVELOPER'S CONFERENCE

- Quando partes específicas do código parecem diferentes, mas na verdade realizam o mesmo trabalho.

Tratamento

Extraia o método e faça chamadas para o novo método nos locais de duplicação de código.

- *A mesclagem de código duplicado simplifica a estrutura do seu código e o reduz.*



Dispensáveis - Classe preguiçosa

THE DEVELOPER'S CONFERENCE

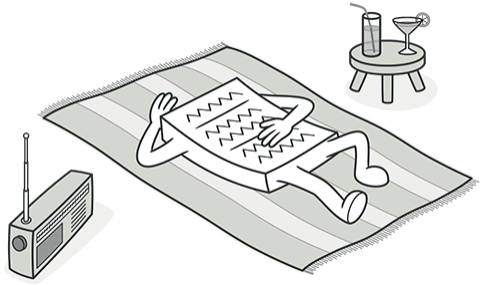
- Classe sem responsabilidade relevante para o código.

Tratamento

Os componentes que são quase inúteis devem ser removidos.

Para subclasses com poucas funções, deve-se remover a hierarquia.

- *Tamanho de código reduzido.*
- *Manutenção mais fácil.*



Dispensáveis - Classe de dados

THE DEVELOPER'S CONFERENCE

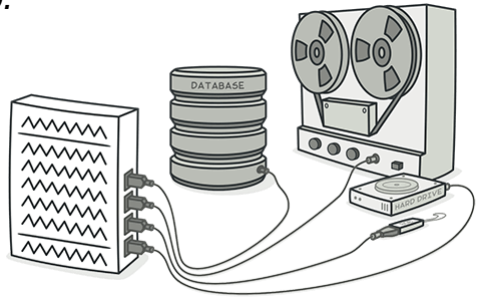
- Uma classe que contém apenas campos e métodos brutos.
- São containers para dados usados por outras classes.
- Essas classes não contêm nenhuma funcionalidade adicional e não podem operar independentemente.

Tratamento

Revisar o código que usa a classe.

Nele, você pode encontrar funcionalidades que seriam melhor localizadas em sua própria classe de dados.

- *Melhora a compreensão e organização do código.*
- *As operações reunidas em um único local, em vez de aleatoriamente em todo o código.*



Dispensáveis – Código morto

THE DEVELOPER'S CONFERENCE

- Uma variável, parâmetro, campo, método ou classe não é mais usada.
- Quando os requisitos para o software foram alterados ou as correções foram feitas, ninguém teve tempo de limpar o código antigo.

Tratamento

Exclua o código não utilizado e os arquivos desnecessários.

- *Tamanho de código reduzido.*
- *Suporte mais simples.*

Dispensáveis – Generalidade especulativa

THE DEVELOPER'S CONFERENCE

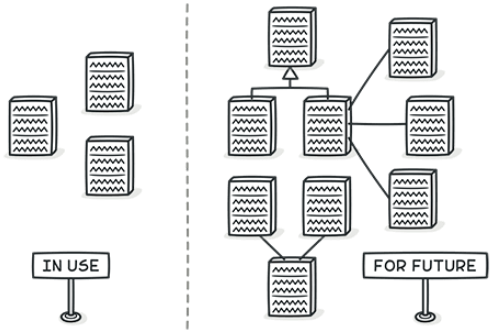
- Há uma classe, método, campo ou parâmetro não utilizado.
- Criado "por precaução" para oferecer suporte a recursos futuros previstos que nunca são implementados

Tratamento

↓

Livre-se deles!

- *Código mais fino.*
- *Suporte mais fácil.*



Acopladores

THE DEVELOPER'S CONFERENCE

- Todos os membros deste grupo contribuem para o acoplamento excessivo entre as classes ou mostram o que acontece se o acoplamento for substituído por delegação excessiva.

Acopladores – Feature invejosa

THE DEVELOPER'S CONFERENCE

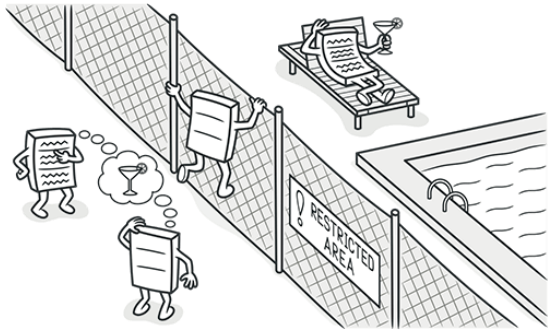
- Um método acessa os dados de outro objeto mais do que seus próprios dados.

Tratamento

Determinar qual classe contém a maioria dos dados usados.

Coloque o método nesta classe junto com os outros dados.

- *Menos duplicação de código.*
- *Melhor organização do código.*



Acopladores - Intimidade inapropriada

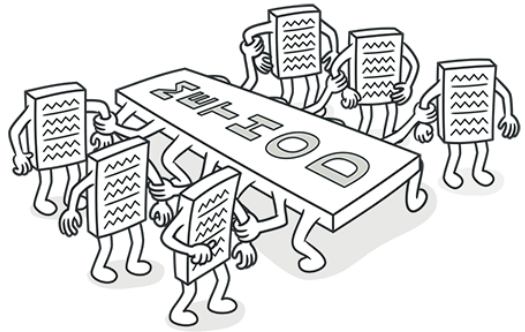
THE DEVELOPER'S CONFERENCE

- Uma classe usa os campos e métodos internos de outra classe.
- Boas classes devem saber o mínimo possível uma sobre a outra.

Tratamento

Mover partes de uma classe para a classe na qual essas partes são usadas.

- *Melhora a organização do código.*
- *Simplifica o suporte e a reutilização de código.*



Acopladores - Cadeias de mensagens

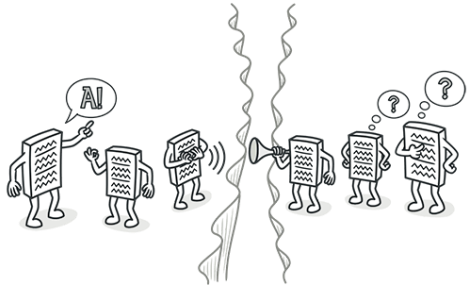
THE DEVELOPER'S CONFERENCE

- Uma série de chamadas semelhantes.
- Quando um método solicita um objeto, esse objeto solicita outro, e assim por diante. Essas cadeias significam que o método depende da navegação ao longo da estrutura da classe.
- Quaisquer mudanças nesses relacionamentos requerem a modificação do método.

Tratamento

Crie um novo método na classe A que delegue a chamada para o objeto B. Agora o método não conhece ou depende da classe B.

- *Reduz as dependências entre as classes de uma cadeia.*



Acopladores - Homem do meio

THE DEVELOPER'S CONFERENCE

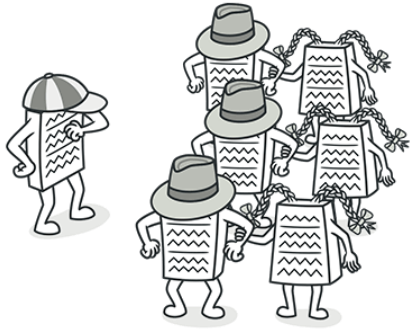
- Uma classe que executa apenas uma ação, delegando trabalho a outra classe.

Tratamento

↓

Livre-se dele!

- *Código menos volumoso.*
- *Cuidado! Um intermediário pode ter sido adicionado para evitar dependências entre classes.*



Ferramentas para métricas de qualidade de código.

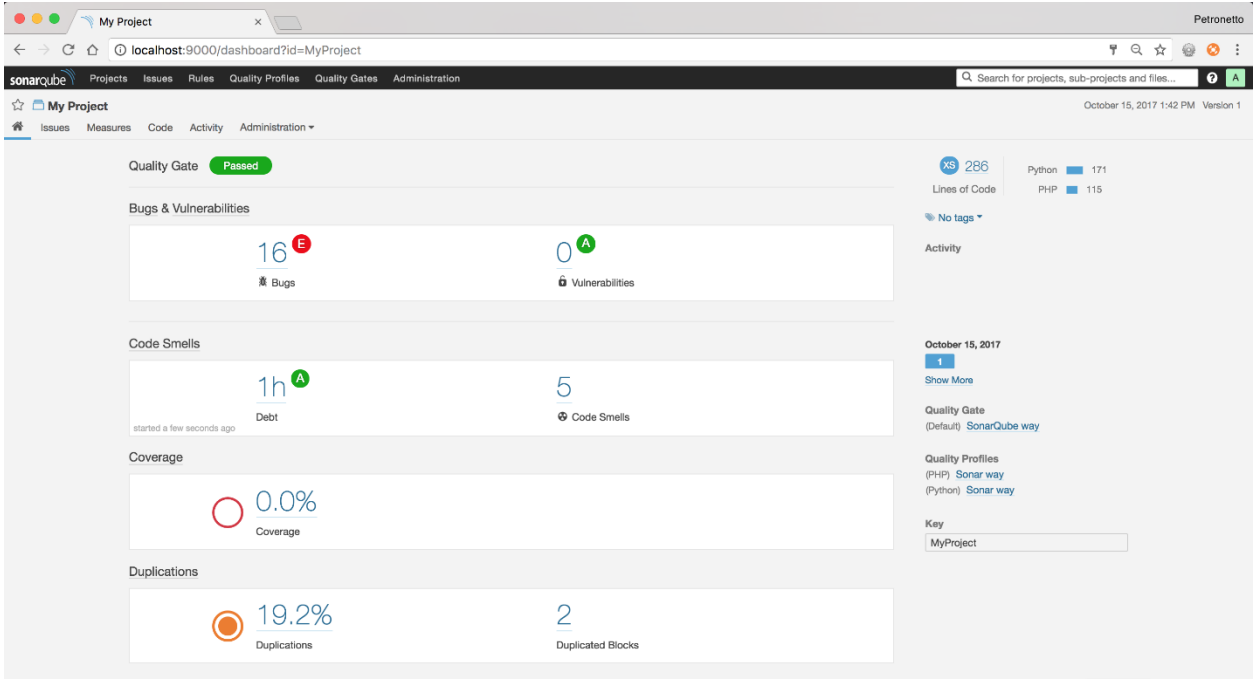
THE DEVELOPER'S CONFERENCE

- SonarQube
  - Plataforma de código aberto para inspeção contínua da qualidade do código, para executar revisões automáticas com análise estática do código para detectar bugs, code smell e vulnerabilidades de segurança.
  - Abrange uma ampla área de pontos de verificação de qualidade de código que incluem: arquitetura e design, complexidade, Duplicações, regras de codificação, potenciais bugs, testes unitários, etc.
  - Um conjunto de recursos que ajudam a manter a qualidade do seu código.

sonarqube

Ferramentas para métricas de qualidade de código.

THE DEVELOPER'S CONFERENCE



Ferramentas para métricas de qualidade de código.

THE DEVELOPER'S CONFERENCE

- biteGarden
  - Plug-in para gerenciar sua instâncias do SonarQube ™ para obter uma visão geral de todas as métricas de qualidade agregadas de três maneiras: todos os projetos, grupos personalizados definidos nas configurações ou usando tags de projetos.

- Sonnar Scanner
  - Um executável que irá realizar o scan do nosso código e enviá-lo para o sua aplicação do SonarQube.

Finalizando...

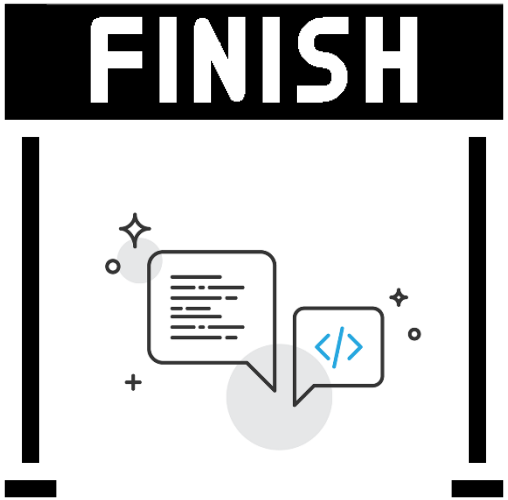
THE DEVELOPER'S CONFERENCE

- Seguir uma abordagem de desenvolvimento corretiva e evolutiva
- Aprender sobre code smells torna mais fácil identificar problemas em códigos.
  - Se conhecemos os problemas e sabemos identificar, saberemos apontar quais os tipos de abordagens pra corrigir, até mesmo sem conhecer a base de código inteira.
- Sempre que codificar, ao terminar limpe seu código, tente melhora-lo.
- Se pergunte se está legível, se qualquer pessoa consegue entender o que foi escrito.

Finalizando...

THE DEVELOPER'S CONFERENCE

- Pague o débito!!!



- Qualidade de código

≠

Qualidade de software

Links

THE DEVELOPER'S CONFERENCE

- https://www.industriallogic.com/wp-content/uploads/2005/09/smellstorefactorings.pdf
- https://refactoring.guru/
- http://martinfowler.com/bliki/CodeSmell.html

THE DEVELOPER'S CONFERENCE

Obrigada!

ldutra.info@gmail.com