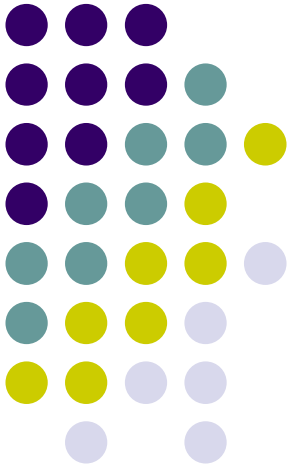
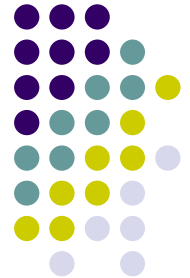
Refatoração

Fabrício Sousa



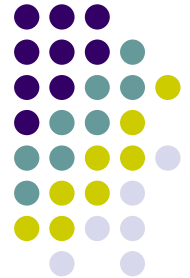
REDE FTC
FACULDADE DE TECNOLOGIA E CIÊNCIAS

Refatoração (Substantivo)



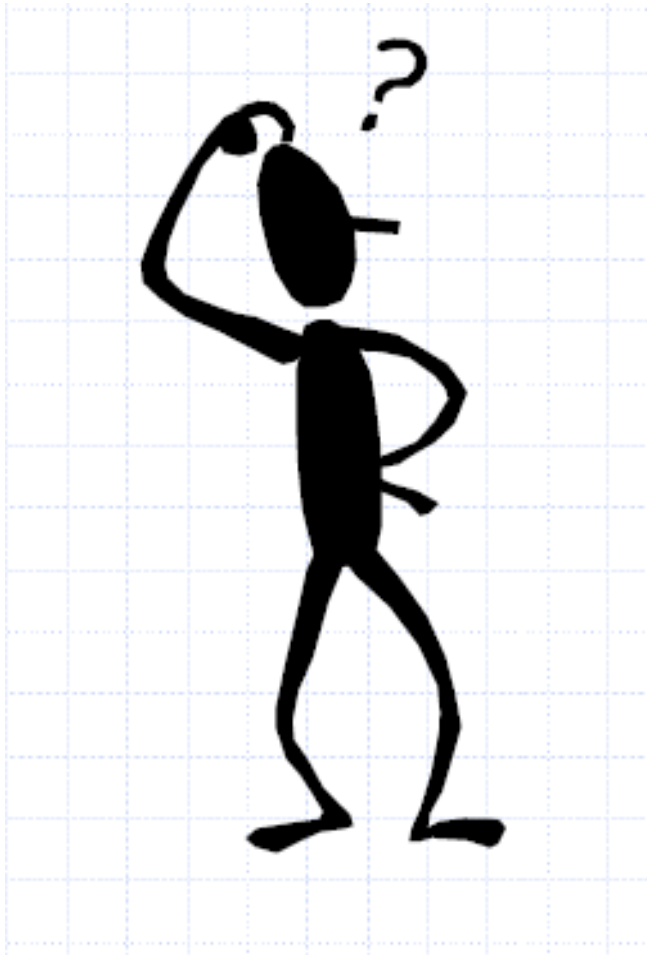
- Uma alteração feita na estrutura interna do software para torná-lo mais fácil de ser entendido e menos custoso de ser modificada sem alterar seu comportamento observável

Refatorar (verbo)

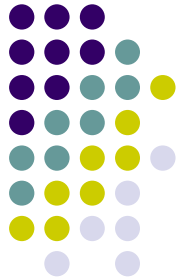


- Reestruturar software aplicando uma série de refatorações sem alterar seu comportamento observável

A refatoração é apenas limpeza de código?

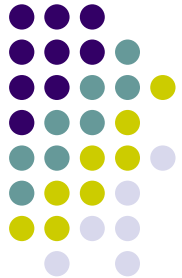


Resposta...



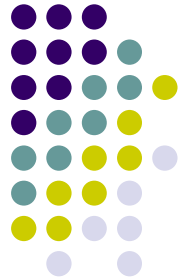
- De certo modo sim..mas vai muita além disso
  - Técnica para limpar código de uma maneira mais eficiente e controlada

Objetivo



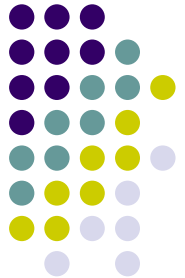
- 1º Tornar o software mais fácil de entender e modificar
- 2º Não altera o comportamento observável do software
  - Executa a mesma função de antes

Outras Definições:



- É o processo de mudar um software de forma que o comportamento externo do código não seja alterado mesmo que sua estrutura interna seja incrementada.
- É o processo de reorganizar o projeto de um sistema para torná-lo mais flexível e/ou reusável.

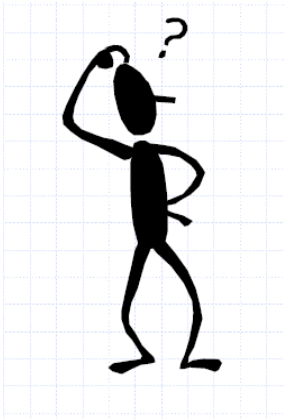
Os dois chapéus



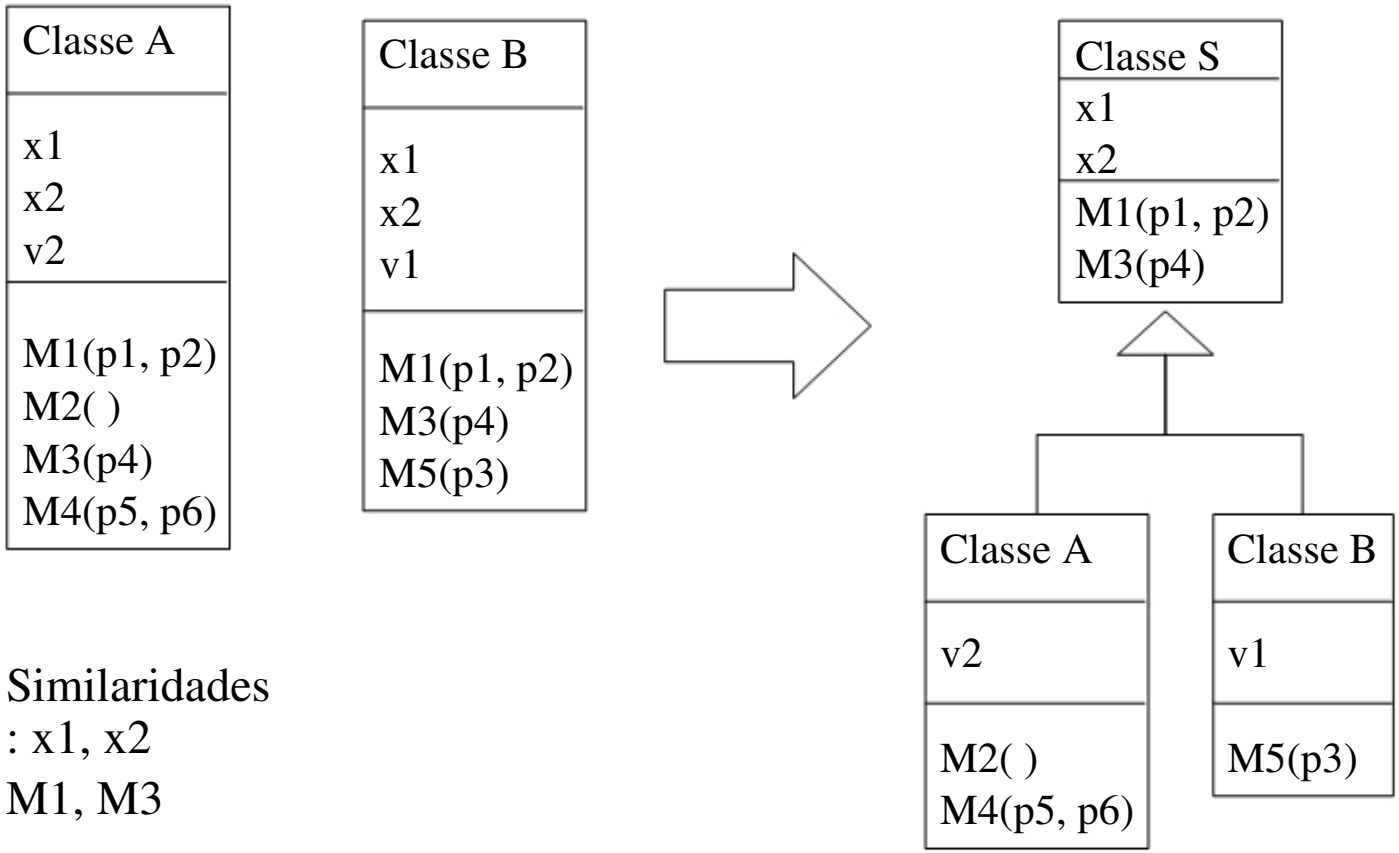
- Adicionar funcionalidades e refatorar
  - Hora assumimos um chapéu , hora assumimos o outro

Nova Funcionalidade

Refatorar

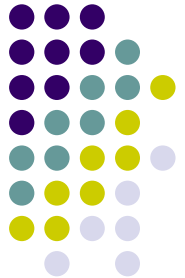


Exemplo 1



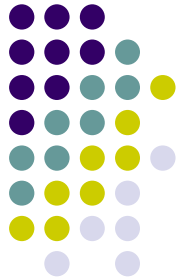
Similaridades
: x1, x2
M1, M3

Refatorar X Reescrever



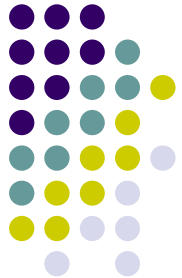
- Refatorar:
  - Não altera a funcionalidade ou o conteúdo do sistema
- Reescrever:
  - Altera seu conteúdo ou funcionalidade

Vantagens



- Eficiência
- Manutenibilidade:
  - É mais fácil manter um sistema quando o código está “limpo”

Vantagens



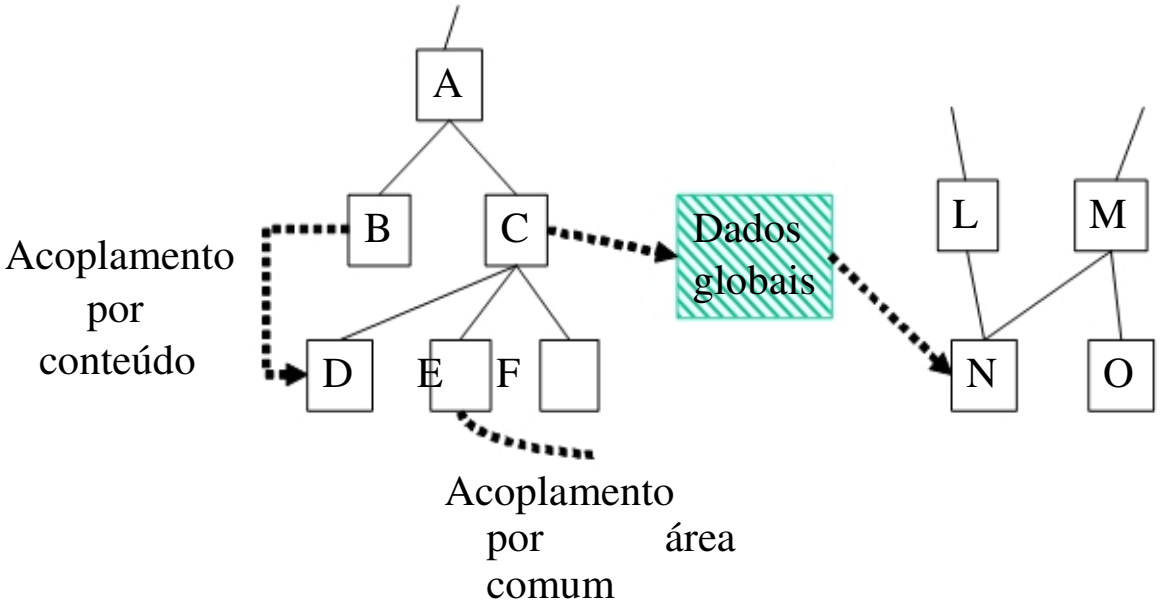
- Eficiência
  - Exemplo:

Seja a expressão: (x-1) * (x+1) = x^2 -1

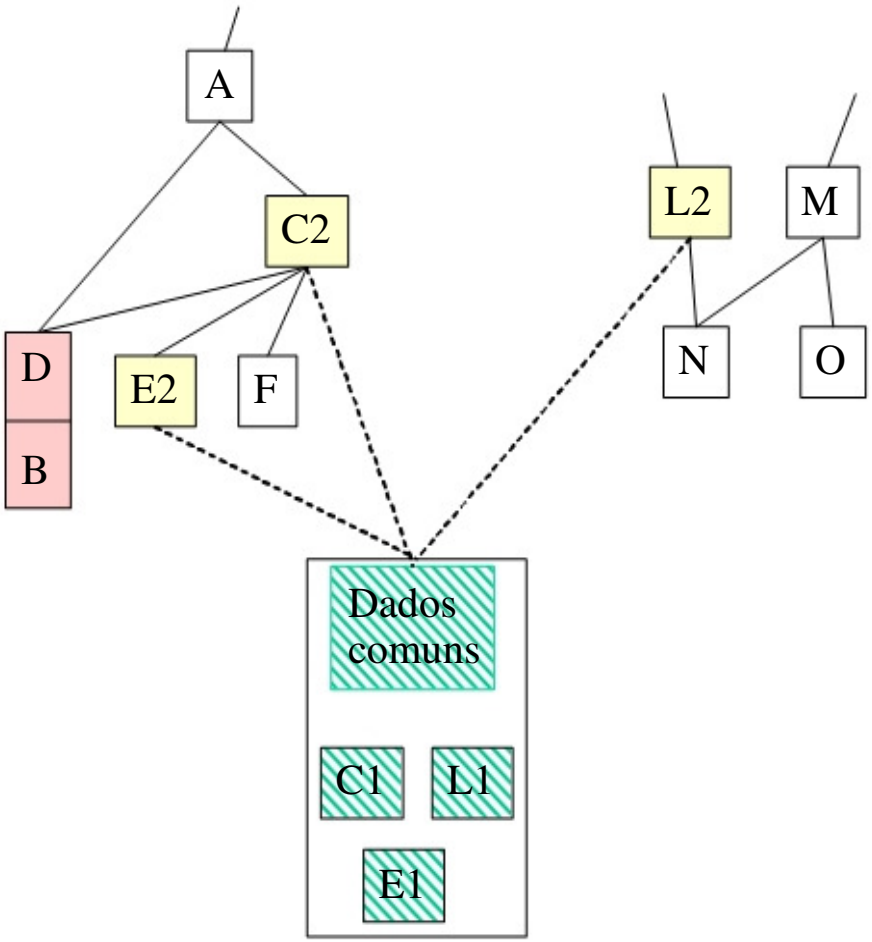
- Qual é mais eficiente?
- Lado esquerdo Por que?
- Usa operações mais simples

apesar de possuir mais termos e operadores

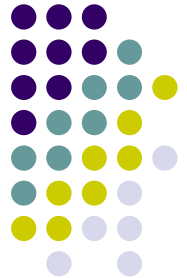
Exemplo 2


A
B
C
D
E
F
L
M
N
O
Dados globais
Acoplamento por conteúdo
Acoplamento por área comum

Exemplo 2



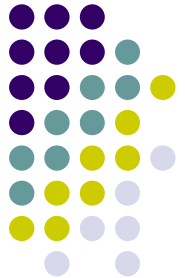
Fatorar X Refatorar



- Fatorar: descobrir os componentes do sistema

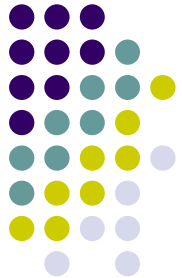
- Refatorar: achar um conjunto diferente de componentes que faça o mesmo trabalho inicial

Objetivos



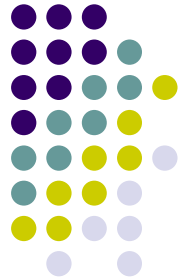
- Tornar o código mais claro, limpo, simples e elegante
- Permitir que componentes mais simples ou expressões mais eficientes sejam usadas
- Exemplos:
  - trocar o nome de uma variável
  - acrescentar um novo método a uma classe

Quando usar



- Quando o código não estiver claro
- Para eliminar código duplicado
- Melhorar legibilidade
- Separar a lógica existente de uma alteração necessária
- Substituir um algoritmo por outro
- Melhorar o desempenho

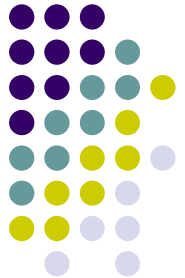
Benefícios



- Adiciona qualidade ao sistema pois:
  - ✍ reduz o código duplicado
  - ✍ Aumenta a simplicidade
  - ✍ Facilita a leitura
  - ✍ Melhora a performance

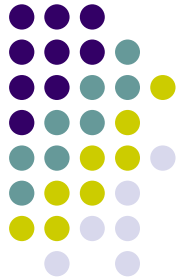

Por que devo refatorar?

Por que devo refatorar?



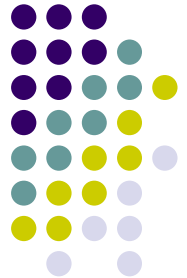
- Não é a cura de todos os problemas
- Ferramenta valiosa, um alicate de prata que lhe ajuda a manter seu código seguro
- Ferramenta que deve e pode ser usada por diversos propósitos

Por que devo refatorar?



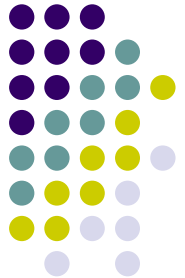
- Melhora o projeto de Software
- Torna o software mais fácil de entender
- Refatorar ajuda a encontrar falhas
- Ajuda a programar mais rápidamente

Quando devo refatorar?



- Devo alocar duas semanas a cada dois meses para refatorar?
  - Não devemos separar tempo e sim devemos fazer todo o tempo em pequenas rajadas
  - Você não decide refatorar, você refatora porque quer fazer alguma coisa qualquer e refatorar ajuda a fazê-lo.

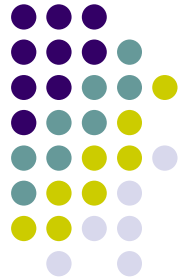
A regra do Três



- Diretriz de Don Roberts
  - “Na primeira vez que faz algo, você apenas faz. Na segunda vez que faz algo parecido, você estremece diante da duplicação, mas faz de qualquer forma. Na terceira vez em que faz algo parecido, você refatora.

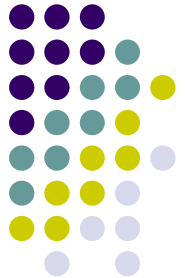
Três vezes e você refatora

Refatore quando acrescentar funções



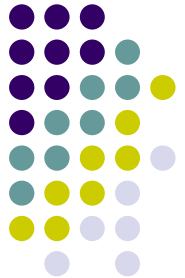
- Compreende algum código que preciso melhorar
  - Pode ter sido escrito por outra pessoa ou por mim
  - Tenho que entende o que o código faz

Refatore quando precisar consertar uma falha



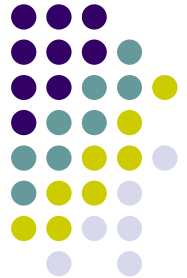
- Tornar o código mais compreensível
- Ajuda a encontrar a falha

Refatore enquanto revisa o código



- Algumas organizações fazem revisão de código com freqüência
- Ajuda a difundir o conhecimento
- O código pode estar claro para mim, mas para a equipe não
- Revisão
  - Oportunidade para surgir novas idéias

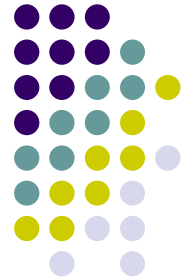
O que torna um programa difícil de trabalhar?



- Programas que são difíceis de ler, são difíceis de modificar
- Programs que têm lógica duplicada, são difícies de modificar
- Programas qe, para inclusão de novas funcionalidades, requerem alteração de código existente, são difícies de modificar
- Programas com lógicas condicionais complexas são difíceis de modificar

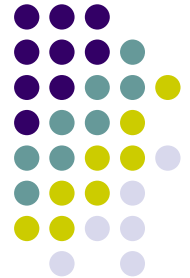
Quero refatorar, o que digo ao meu Gerente?

Quero refatorar, o que digo ao meu Gerente?



- Se ele for experiente, introduzir o assunto não é difícil
- Enfatizar aspectos relacionados a qualidade
  - Revisão, Inspeções
    - Maneira importante de reduzir falhas
- Mas se ele for orientado a cronograma, não diga nada

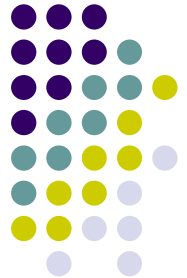
Quero refatorar, o que digo ao meu Gerente?



- Se eu for adicionar uma nova função
  - Projeto não apropriado a mudança
    - Descubro que é mais rápido fatorar primeiro e depois acrescentar a função
- Se eu preciso consertar uma falha
  - Preciso entender como o programa funciona
    - Descubro que refatorar é a maneira mais fácil para fazer isso
- Gerente orientado a cronograma quer que o programador faça as coisas da maneira mais rápida
  - A maneira mais rápida é refatorar, portanto eu refatoro.

Indireção e refatoração

Indireção e refatoração

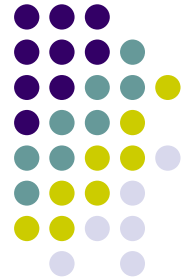


- Tende a dividir objetos grandes em diversos objetos.
- Maior parte da refatoração produz indireção
- Cada vez que você divide algo em duas partes, você tem mais coisas para gerenciar.
  - Um objeto delega a um objeto que delega a outro...

Problemas com a Refatoração

- Quando aprendemos uma técnica é dificil vermos onde ela não se aplica
- Refatoração
  - Sabemos benefícios
  - Não conhecemos as limitações
    - Existem

Quando não devo refatorar?



- Quando você deve escrever da estaca zero
  - Mais recomeçar do princípio do que refatorar
- Sinal claro da necessidade de reescrever
  - Quando o código simplesmente não funciona
    - Descobrimos testando...
- A outra vez que devemos evitar refatoração quando estiver próximo de um prazo final
  - Deixe para o futuro a “dívida”

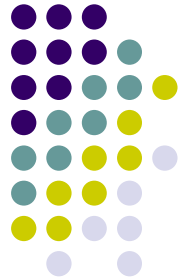
De onde veio a Refatoração?

- Bons programadores certamente têm gasto pelo menos algum tempo limpando seu código.
  - Eles o fazem porque aprenderam que código limpo é mais fácil de ser alterado do que o código complexo e confuso
  - Bons programadores sabem que raramente escrevem código limpo na primeira vez
- Adotado na metodologia XP (E**X**treme **P**rogramming)

“Maus Cheiros” no Código

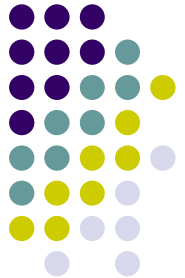
- “Se cheirar mal, troque-o”
  - Vovó Beck, discutindo filosofia de traseiro de bebê

“Maus Cheiros” no Código



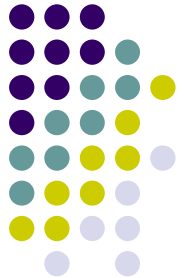
- O que iremos fazer é dar uma indicação de que há problemas que podem ser resolvidos por uma refatoração
- Você terá que desenvolver sua própria percepção sobre quantas variáveis de instância são demais e quantas linhas de código em um método são excessivas

1 Código Duplicado



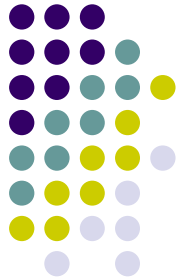
- Número um no ranking dos cheiros é o código duplicado
- Se você vir o mesmo código em mais de um lugar, pode ter certeza de que seu programa ficará melhor se você encontrar um meio de unificá-los.
- Problema mais simples
  - A mesma expressão em dois métodos da mesma classe
    - Utilizar o *Extrair Método* e chamar o código de ambos os lugares

1 Código Duplicado



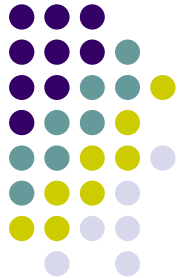
- Outro problema comum é quando você tem a mesma expressão em duas subclasses irmãs.
  - Eliminação: *Extrair Método* em ambas as classes e então *Subir Método na Hierarquia*

2 Método longo



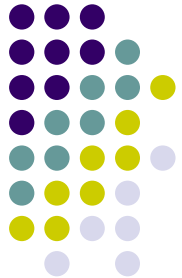
- Os programas OO que vivem melhor e por mais tempo são aqueles com métodos curtos.
- Quanto maior for o método, mais difícil é entendê-lo
- Boa dica
  - Bom nome para um método, não precisa olhar o corpo do mesmo

2 Método longo



- Boa heurística
  - Sempre que sentimos a necessidade de comentar , em vez disso escreva um método
    - Tal método contém o código que seria comentado
- 99% dos casos, tudo que precisamos para encurtar um método aplicar: *Extrair Método*

3 Classe Grande

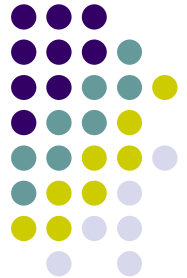


- Quando uma classe tenta fazer muita coisa, ela tem muitas variáveis de instâncias
  - Extrair classe
- Classe com código demais é solo fértil para código duplicado, caos e morte
  - Solução mais simples
    - Eliminar a redundância da própria classe
    - *Extrair Classe* ou *Extrair Subclasse*

4 Lista de Parâmetros longa

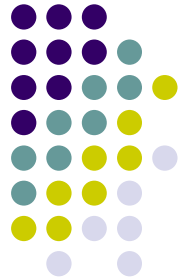
- Quando éramos programadores iniciantes, fomos ensinados a passar como parâmetro tudo o que uma rotina precisasse
- Maneira correta
  - Passar o suficiente para que o método possa obter tudo que precisa
- Lista de parâmetros longas são difíceis de entender, porque se torna difícil de usar
  - Substituir Parâmetro por método

5 Alteração Divergente



- Estruturamos nosso software para tornar as alterações mais fáceis, afinal software deve ser *soft*(maleável)
- Alteração
  - Queremos ser capazes de ir a um único ponto no sistema, claramente definido, e fazer essa alteração

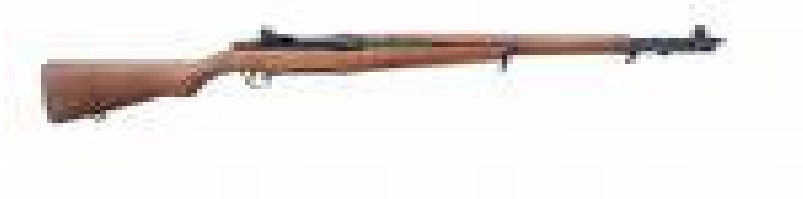
5 Alteração Divergente



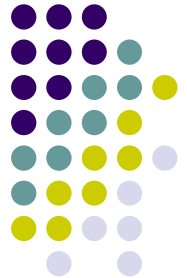
- Ocorre quando é uma classe é alterada de diferentes maneiras por diferentes razões
  - Extrair Classe

6 Cirurgia com Rifle

- Parecida com a alteração divergente, mas é o oposto
- Cada vez que executar uma mudança , tem que fazer muitas alterações pequenas em muitas classes diferentes
- Quando elas estão por toda a parte são difíceis de encontrar e é fácil deixar de fazer alguma alteração importante

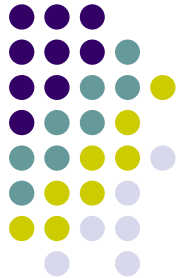


7 Inveja dos dados



- A essência de objetos é que eles são uma técnica para empacotar dados com os processos usados nesses dados
- Indício clássico do problema
  - É um método que se mostra mais interessado em uma classe diferente daquela na qual se encontra
  - O foco mais comum da inveja
    - Dados
    - *Mover Métodos*

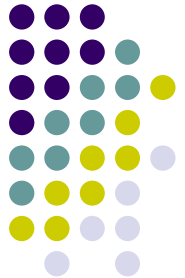
8 Grupos de Dados



- Os dados tendem a ser como crianças
  - Gostam de ficar em grupos
  - Você verá muitas vezes os mesmos três ou quatros dados juntos em muitos lugares

Dúvidas?

Bibliografia



- FOWLER, Martin. *Refatoração: aperfeiçoando o projeto de código existente.*Porto Alegre: Bookmn, 2004.

Vida de Programador

ANATOMIA DO PROGRAMADOR

Informaticus programatorum