Código Limpo

Curso de Verão 2010 - IME/USP

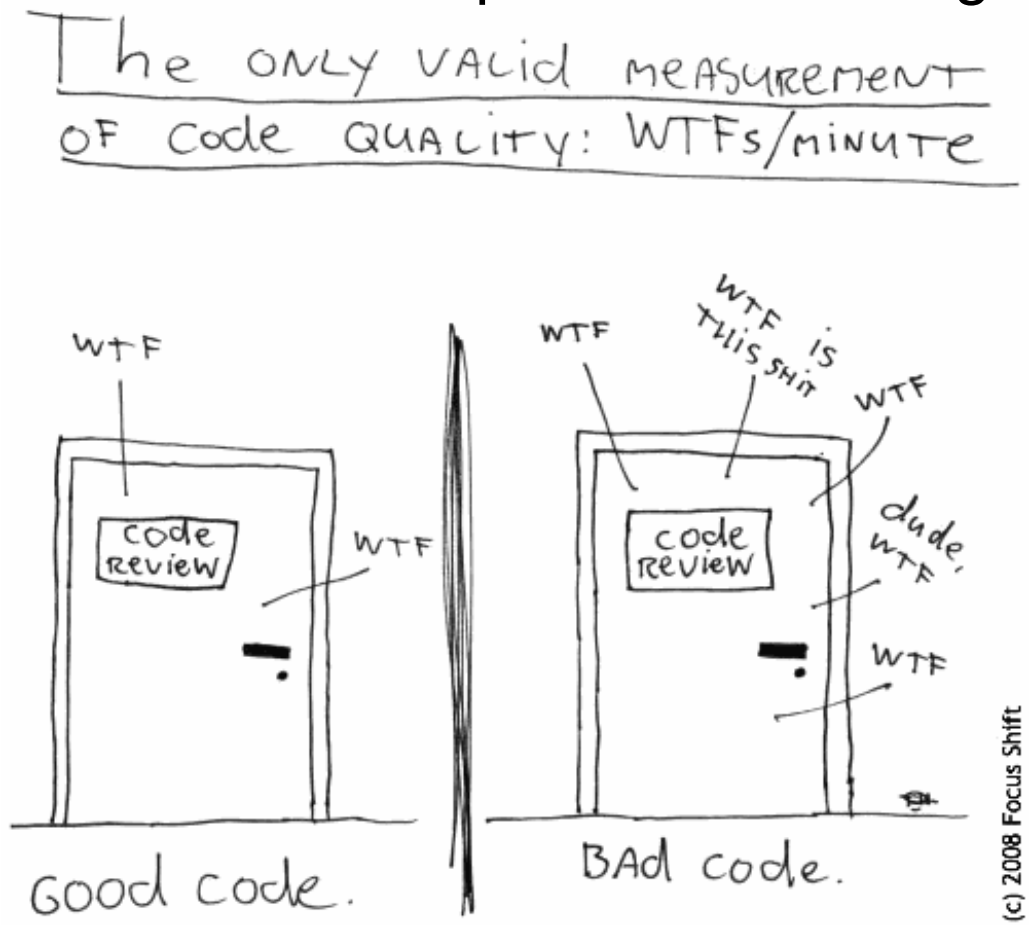
www.agilcoop.org.br

Hugo Corbucci

AgilCoop

Introdução

A única métrica válida de qualidade de código: VSFs/min



Código bom

Código ruim

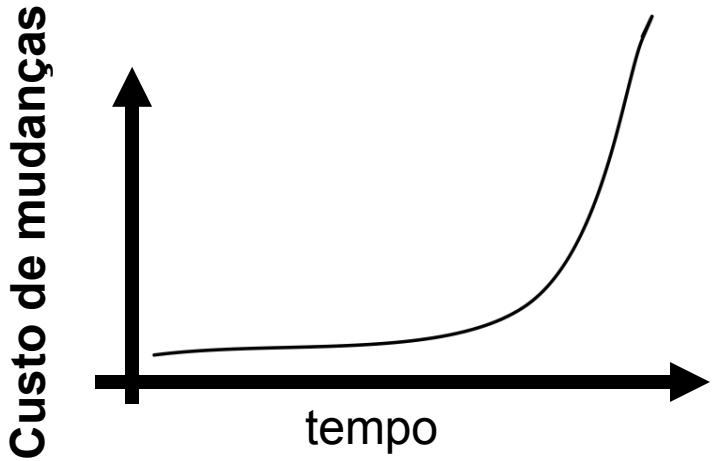
Introdução

"Você sabe que está trabalhando num código limpo quando cada rotina que você lê é aquilo que você esperava que ela fosse.
Pode chamá-lo de código bonito quando o código também faz parecer que a linguagem foi feita para o problema."

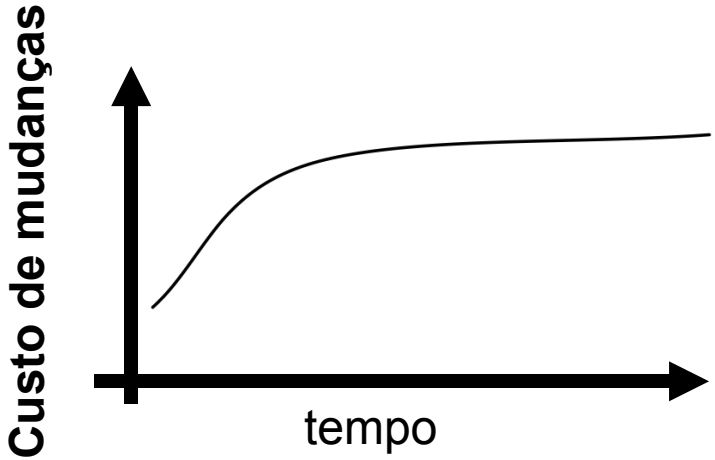
-- Ward Cunningham

Introdução

Como passar de

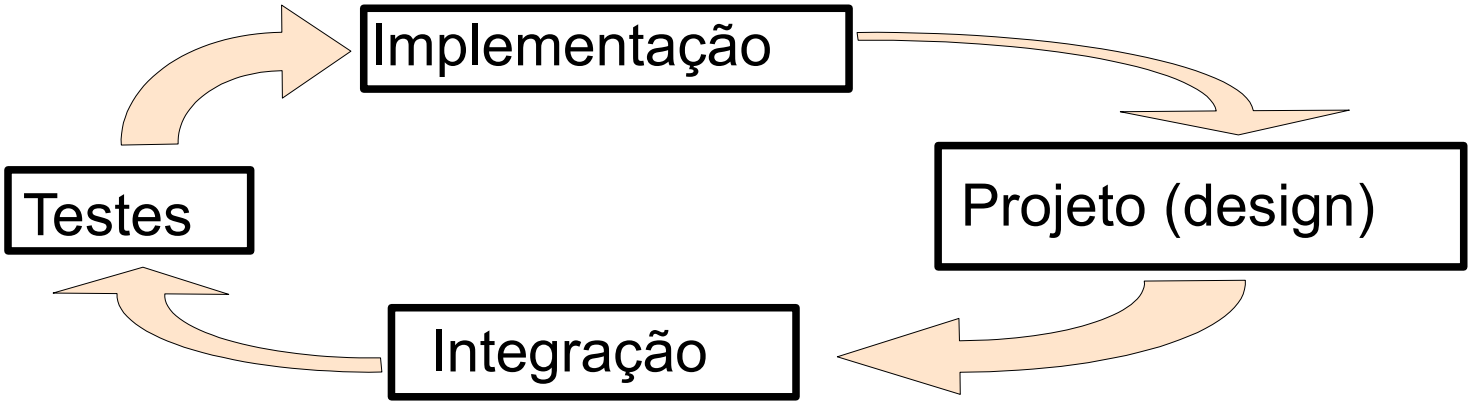


para



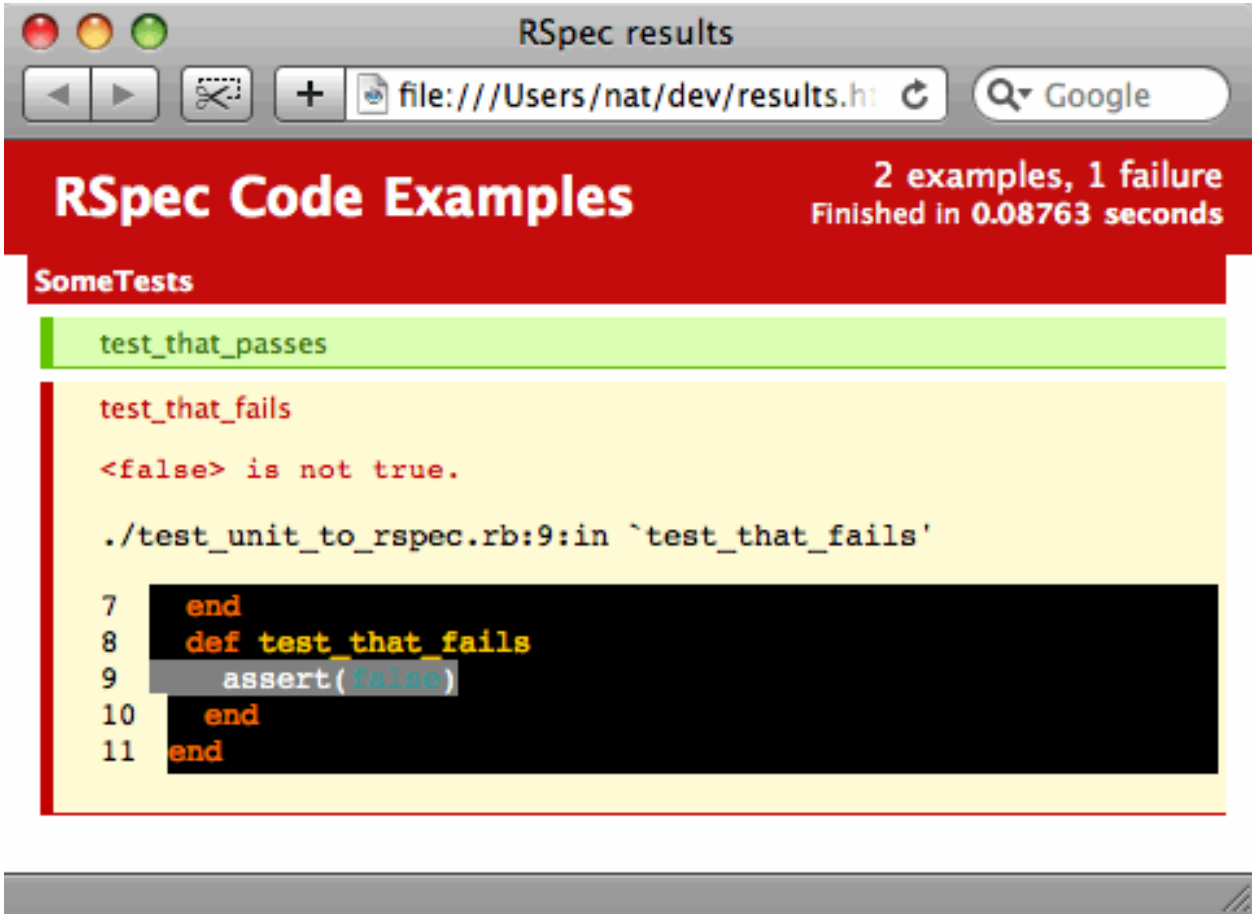
?

Introdução


Implementação
Projeto (design)
Integração
Testes


caps lock
tab

Introdução


RSpec results
file:///Users/nat/dev/results.h
Google
RSpec Code Examples
2 examples, 1 failure
Finished in 0.08763 seconds
SomeTests
test_that_passes
test_that_fails
<false> is not true.
./test_unit_to_rspec.rb:9:in `test_that_fails'
7 end
8 def test_that_fails
9 assert(false)
10 end
11 end

Introdução

(refatoração)

Regra dos escoteiros

"Deixe o acampamento mais limpo do que estava quando o encontrou."

Traduzindo para software:

Sempre que ler um trecho de código, refatore algo nele para melhorá-lo (desde que seus testes estejam todos passando).

Use nomes com sentido

```
public class C {
    private boolean[] p;

    public C(int m) {
        p = new boolean[m];
        p[0] = p[1] = false;
        for (int n = 2; n < m; n++)
            p[n] = true;

        for (int n = 2; n < m; n++)
            if (p[n])
                for (int f = 2; n * f < p.length; f++)
                    p[n * f] = false;
    }

    public boolean p(int n) {
        return n > 0 && n < p.length && p[n];
    }
}
```

Use nomes com sentido

```
public class CalculadoraDePrimos {
    private boolean[] ehPrimo;

    public CalculadoraDePrimos(int numeroMaximo) {
        ehPrimo = new boolean[numeroMaximo];
        ehPrimo[0] = ehPrimo[1] = false;
        for (int numero = 2; numero < numeroMaximo; numero++)
            ehPrimo[numero] = true;

        for (int numero = 2; numero < numeroMaximo; numero++)
            if (ehPrimo[numero])
                for (int fator = 2; numero * fator < ehPrimo.length; fator++)
                    ehPrimo[numero * fator] = false;
    }

    public boolean ehPrimo(int numero) {
        return numero > 0 && numero < ehPrimo.length && ehPrimo[numero];
    }
}
```

Use nomes com sentido

- Use nomes que revelem sua intenção
  - Ex:
    int tempoPassadoEmDias;
    ao invés de
    int t;

Use nomes com sentido

- Use nomes que revelem sua intenção
- Evite desentendimentos
  - Ex:
    int hipotenusa;
    ao invés de
    int hp;

Use nomes com sentido

- Use nomes que revelem sua intenção
- Evite desentendimentos
- Faça distinções com significado
  - Ex:

    int origem, destino;
    ao invés de:
    int a1, a2;

Use nomes com sentido

- Use nomes que revelem sua intenção
- Evite desentendimentos
- Faça distinções com significado
- Use nomes pronunciáveis
  - Ex:
    int somaDosQuadradosDosCatetos;
    ao invés de:
    int sdQdC;

Use nomes com sentido

- Use nomes que revelem sua intenção
- Evite desentendimentos
- Faça distinções com significado
- Use nomes pronunciáveis
- Use nomes buscáveis
  - Ex:
    boolean oxitona;
    ao invés de:
    boolean o;

Use nomes com sentido

- Use nomes que revelem sua intenção
- Evite desentendimentos
- Faça distinções com significado
- Use nomes pronunciáveis
- Use nomes buscáveis
- Evite mapas mentais
  - Ex: int fibonacci;
    ao invés de:
    int fbn;

Use nomes com sentido

- Uma palavra por conceito
  - Ex:

    Escolha entre apagar, remover, *deletar*, etc.

Use nomes com sentido

- Uma palavra por conceito
- Use nomes próximos do domínio da solução ou do problema

Use nomes com sentido

- Uma palavra por conceito
- Use nomes próximos do domínio da solução ou do problema
- Mantenha o contexto pequeno para manter o sentido

Funções/Métodos

```
private static void drawSmiley(final Shell shell, PaintEvent event) {
    Rectangle rect = shell.getClientArea();
    int diameter = Math.min(rect.width - 1, rect.height - 1);
    int x = (rect.width - diameter) / 2;
    int y = (rect.height - diameter) / 2;
    Rectangle smileySquare = new Rectangle(x, y, diameter, diameter);

    GC gc = event.gc;
    gc.setLineWidth(3);
    Display display = event.display;
    Color yellow = new Color(display, 255, 255, 0);
    gc.setBackground(yellow);
    gc.fillOval(smileySquare.x, smileySquare.y, smileySquare.width,
            smileySquare.height);
    gc.drawOval(smileySquare.x, smileySquare.y, smileySquare.width,
            smileySquare.height);

    Color black = new Color(display, 0, 0, 0);
    gc.setBackground(black);
    int horizontalCenter = smileySquare.x + smileySquare.height / 2;
    int leftEyeX = horizontalCenter - smileySquare.width / 8;
    int eyesStartY = smileySquare.y + smileySquare.height / 8;
    int eyesWidth = smileySquare.width / 16;
    int eyesHeight = smileySquare.height / 4;
    gc.fillOval(leftEyeX, eyesStartY, eyesWidth, eyesHeight);

    int rightEyeX = horizontalCenter + smileySquare.width / 8;
    gc.fillOval(rightEyeX, eyesStartY, eyesWidth, eyesHeight);

    gc.drawArc(smileySquare.x, smileySquare.y - smileySquare.height / 4,
            smileySquare.width, smileySquare.height, 225, 90);
}
```

Funções/Métodos

```
private static void drawCenteredSmileyInShell(Shell shell, PaintEvent event) {
    drawSmiley(event, getSmileyBoundaries(shell));
}
private static Rectangle getSmileyBoundaries(Shell shell) {
    Rectangle rectangle = shell.getClientArea();
    int diameter = Math.min(rectangle.width - 1, rectangle.height - 1);
    int horizontalStart = (rectangle.width - diameter) / 2;
    int verticalStart = (rectangle.height - diameter) / 2;
    return new Rectangle(horizontalStart, verticalStart, diameter, diameter);
}
private static void drawSmiley(PaintEvent event, Rectangle smileyBoundaries) {
    event.gc.setLineWidth(3);
    drawHead(smileyBoundaries, event.gc);
    drawEyes(smileyBoundaries, event.gc);
    drawMouth(smileyBoundaries, event.gc);
}
private static void drawHead(Rectangle headBoundaries, GC gc) {
    setHeadColor(gc);
    gc.fillOval(headBoundaries.x, headBoundaries.y, headBoundaries.width,
            headBoundaries.height);
    gc.drawOval(headBoundaries.x, headBoundaries.y, headBoundaries.width,
            headBoundaries.height);
}
private static void setHeadColor(GC gc) {
    gc.setBackground(new Color(Display.getCurrent(), 255, 255, 0));
}
private static void drawEyes(Rectangle headBoundaries, GC gc) {
    setEyesColor(gc);
    Rectangle eyeBoundaries = getEyeBoundaries(headBoundaries);
    drawLeftEye(gc, headBoundaries, eyeBoundaries);
    drawRightEye(gc, headBoundaries, eyeBoundaries);
}
```

Funções/Métodos

```
private static void setEyesColor(GC gc) {
    gc.setBackground(new Color(Display.getCurrent(), 0, 0, 0));
}
private static Rectangle getEyeBoundaries(Rectangle headBoundaries) {
    int headMiddle = headBoundaries.x + headBoundaries.height / 2;
    int y = headBoundaries.y + headBoundaries.height / 8;
    int width = headBoundaries.width / 16;
    int height = headBoundaries.height / 4;
    return new Rectangle(headMiddle, y, width, height);
}
private static void drawLeftEye(GC gc, Rectangle headBoundaries,
        Rectangle eyeBoundaries) {
    int leftEyeX = eyeBoundaries.x - headBoundaries.width / 8;
    gc.fillOval(leftEyeX, eyeBoundaries.y, eyeBoundaries.width,
            eyeBoundaries.height);
}
private static void drawRightEye(GC gc, Rectangle headBoundaries,
        Rectangle eyeBoundaries) {
    int rightEyeX = eyeBoundaries.x + headBoundaries.width / 8;
    gc.fillOval(rightEyeX, eyeBoundaries.y, eyeBoundaries.width,
            eyeBoundaries.height);
}
private static void drawMouth(Rectangle boundaries, GC gc) {
    gc.drawArc(boundaries.x, boundaries.y - boundaries.height / 4,
            boundaries.width, boundaries.height, 225, 90);
}
```

Funções/Métodos

- Pequenas/os
  - C/C++/C#/Java no máximo 15 linhas
    Ruby/Smalltalk/Python no máximo 5 linhas

Funções/Métodos

- Pequenas/os
- Faça uma coisa
  - Mau cheiro: nomes de métodos com E ou OU

Funções/Métodos

- Pequenas/os
- Faça uma coisa
- Um nível de abstração por função
  - Ex:
    Rectangle limites = obtemLimites(shell);
    gc.setLineDash(new int[]{3, 2, 5});
    gc.drawOval(limites.x, limites.y,
    limites.width, limites.height);
    desenhaOlhos(limites);
    Isso é muito estranho de ler!!!!

Funções/Métodos

- Pequenas/os
- Faça uma coisa
- Um nível de abstração por função
- Argumentos
  - 0, 1 ou 2. 3 raramente.
    Mais do que isso, provavelmente há algo errado!

Funções/Métodos

- Pequenas/os
- Faça uma coisa
- Um nível de abstração por função
- Argumentos
- Sem efeitos colaterais

  Ex: validaSenha não deveria inicializar uma sessão se a senha é válida!

Funções/Métodos

- Pequenas/os
- Faça uma coisa
- Um nível de abstração por função
- Argumentos
- Sem efeitos colaterais
- Prefira exceções a códigos de erros

Funções/Métodos

- Pequenas/os
- Faça uma coisa
- Um nível de abstração por função
- Argumentos
- Sem efeitos colaterais
- Prefira exceções a códigos de erros
- DRY – Don't Repeat Yourself

Comentários

```
StyledTextContent content = printerRenderer.content;
startLine = 0;
endLine = singleLine ? 0 : content.getLineCount() - 1;
PrinterData data = printer.getPrinterData();
if (data.scope == PrinterData.PAGE_RANGE) {
    int pageSize = clientArea.height / lineHeight;//WRONG
    startLine = (startPage - 1) * pageSize;
} else if (data.scope == PrinterData.SELECTION) {
    startLine = content.getLineAtOffset(selection.x);
    if (selection.y > 0) {
        endLine = content.getLineAtOffset(selection.x + selection.y - 1);
    } else {
        endLine = startLine - 1;
    }
}
```

WTF?!?

Comentários

- Não resolvem problemas no código

Comentários

- Não resolvem problemas no código
- Explique pelo código

```
//draw paragraph top in the current page and paragraph bottom in the next
int height = paragraphBottom - paintY;
gc.setClipping(clientArea.x, paintY, clientArea.width, height);
printLine(paintX, paintY, gc, foreground, lineBackground, layout, printLayout, i);
gc.setClipping((Rectangle)null);
printDecoration(page, false, printLayout);
printer.endPage();
page++;
if (page <= endPage) {
    printer.startPage();
    printDecoration(page, true, printLayout);
    paintY = clientArea.y - height;
    int layoutHeight = layout.getBounds().height;
    gc.setClipping(clientArea.x, clientArea.y, clientArea.width, layoutHeight - height);
    printLine(paintX, paintY, gc, foreground, lineBackground, layout, printLayout, i);
    gc.setClipping((Rectangle)null);
    paintY += layoutHeight;
}
```

ou

```
int height = drawParagraphTop(paragraphBottom, paintX, paintY, page);
if (page + 1 <= endPage)
    drawParagraphEnd(height, page + 1);
```

Comentários

- Não resolvem problemas no código
- Explique pelo código
- Bons comentários
  - Ex: Legais/obrigatórios
    Informativos
    Explicando a intenção
    Avisando de consequências
    Usados por pós processamento
    Amplificação
    Javadocs p/ APIs

Comentários

- Não resolvem problemas no código
- Explique pelo código
- Bons comentários
- Comentários ruins
  - Ex: Explica o que o código faz
    Que induzem ao erro
    Código comentado

    Javadocs em código interno

Formatação

```
public class CalculadoraDePrimos {
private boolean[] ehPrimo;
public CalculadoraDePrimos(int numeroMaximo) {
    ehPrimo = new boolean[numeroMaximo];
        ehPrimo[0] = ehPrimo[1] = false;
for (int numero = 2; numero < numeroMaximo; numero++)
ehPrimo[numero] = true;
        for (int numero = 2; numero < numeroMaximo; numero++)
if (ehPrimo[numero])
for (int fator = 2; numero * fator < ehPrimo.length; fator++)
                    ehPrimo[numero * fator] = false;}public boolean ehPrimo(int numero) {
                        return numero > 0 && numero < ehPrimo.length && ehPrimo[numero];
}}
```

Formatação

- O intuito da formatação
  - Facilitar a leitura
  - Evidenciar fluxos
  - Agrupar códigos “próximos”

Formatação

- O intuito da formatação
- Formatação vertical
  - Código deveria ser lido como jornal
  - Espaços entre conceitos
  - Densidade para unir
  - Distância vertical

Formatação

- O intuito da formatação
- Formatação vertical
- Formatação horizontal
  - Código deve caber na sua tela
  - Indentação!
  - Alinhamentos dão destaque. Pense no que destacar antes de alinhar coisas.

Formatação

- O intuito da formatação
- Formatação vertical
- Formatação horizontal
- Regras de formatação da equipe

Tratamento de erros

```
DeviceHandle handler = getHandle(DEV1);
if(handler != DeviceHandle.INVALID)
    saveStatus(handler);
else
    logError(handler);
```

```
try {
    Method method = getClass().getMethod(methodName);
    method.invoke(this, methodName);
} catch (SecurityException e) {
    e.printStackTrace();
} catch (NoSuchMethodException e) {
    e.printStackTrace();
} catch (IllegalArgumentException e) {
    e.printStackTrace();
} catch (IllegalAccessException e) {
    e.printStackTrace();
} catch (InvocationTargetException e) {
    e.printStackTrace();
}
```

Tratamento de erros

- Exceções ao invés de códigos de erro
  - Código de erro => if logo após chamada

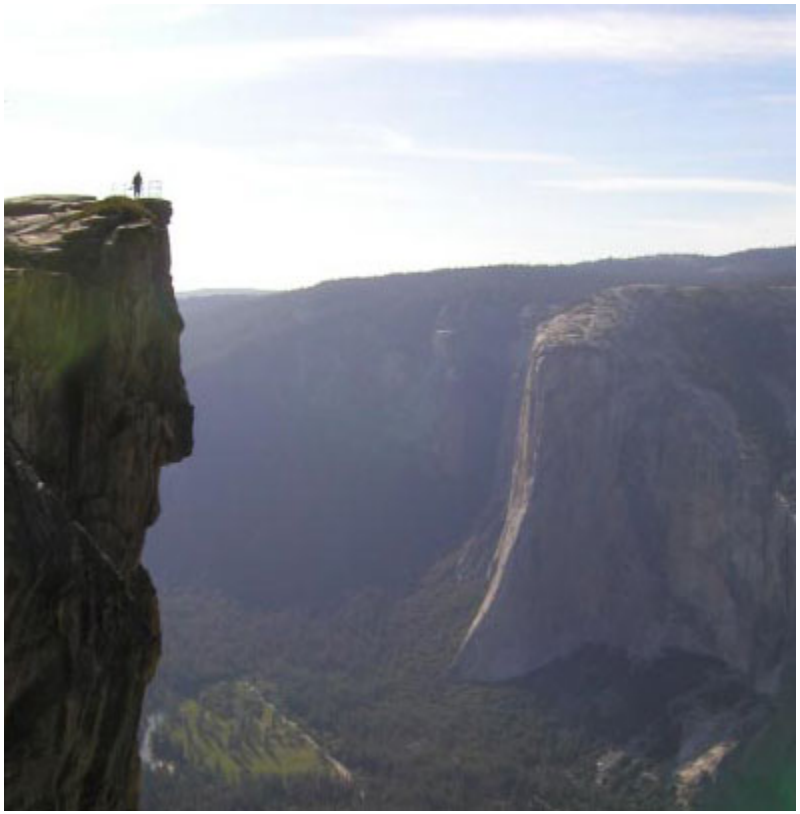
Tratamento de erros

- Exceções ao invés de códigos de erro
- Use exceções não-verificadas (*unchecked*)
  - Exceções verificadas poluem até serem tratadas

Tratamento de erros

- Exceções ao invés de códigos de erro
- Use exceções não verificadas
- Não devolva/passe null
  - Padrão NullObject

No limite!



No limite!

- Usando código de outros
  - Encapsule seu uso externo
  - Escreva testes para garantir comportamento

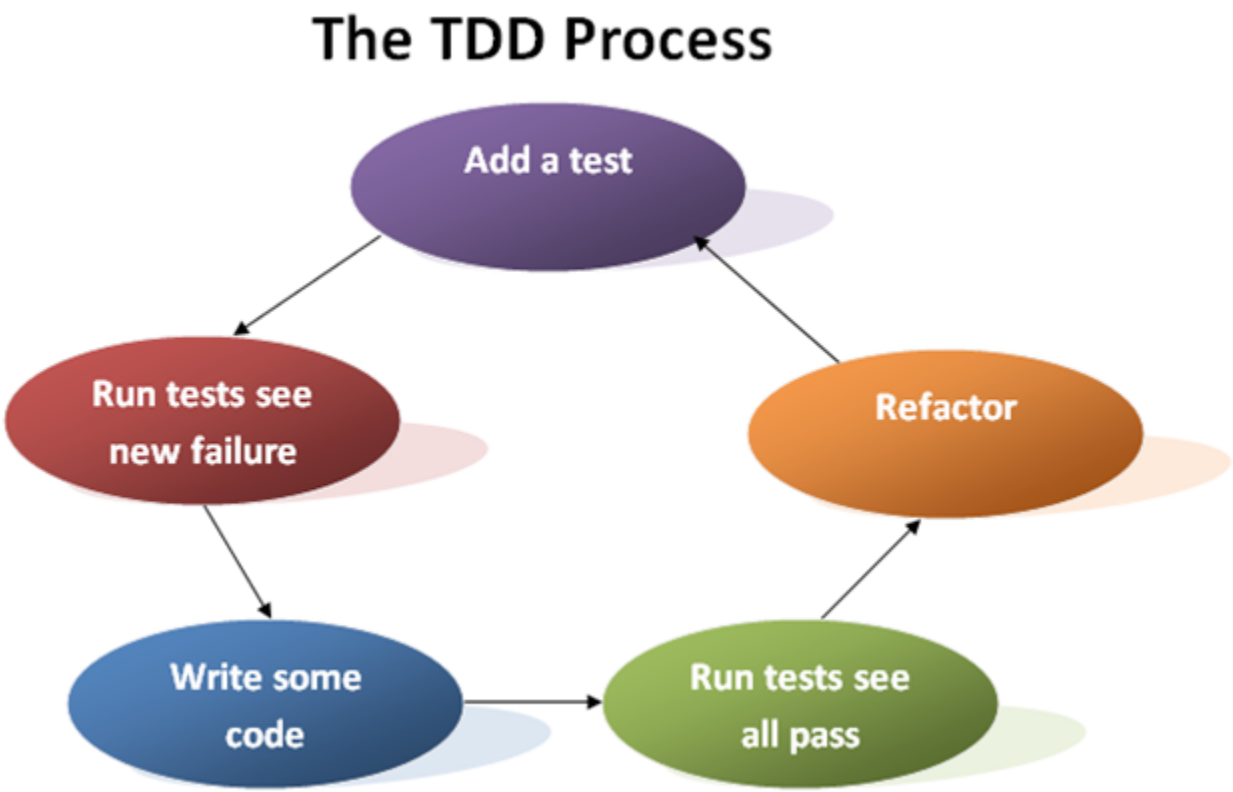
No limite!

- Usando código de outros
- Usando código que ainda não existe
  - Use interfaces
  - Use testes (que falham por enquanto)

No limite!

- Usando código de outros
- Usando código que ainda não existe
- Mantendo os limites limpos
  - Atualize suas fachadas
  - Quando o código existir, acerte os testes

Testes e TDD


The TDD Process
Add a test
Run tests see new failure
Write some code
Run tests see all pass
Refactor

Testes e TDD

- 3 regras de TDD
  - Você não pode escrever código de produção até escrever um teste que falha.
  - Você não pode escrever mais de um teste necessário para falhar, e não compilar é falhar.
  - Você não pode escrever mais código de produção do que é suficiente para passar o teste que está falhando.

Testes e TDD

- 3 regras de TDD
- Mantenha seus testes muito limpos
  - Refatore
  - Use padrões
  - Tenha fábricas, classes para asserção, etc.
  - Quebre em funções
  - Mocks, Stubs, Spies, etc.

Testes e TDD

- 3 regras de TDD
- Mantenha seus testes muito limpos
- Uma asserção por teste
  - Se tiver mais, você não sabe o que está passando e o que não está.

Testes e TDD

- 3 regras de TDD
- Mantenha seus testes muito limpos
- Uma asserção por teste
- F.I.R.S.T.
  - Fast/Rápido
    Independente
    Repetível
    Self-Validating/Auto-validate
    Timely/Em tempo

Classes

- Encapsuladas/Coesas
  - Deveriam funcionar sem depender da atitude de outros

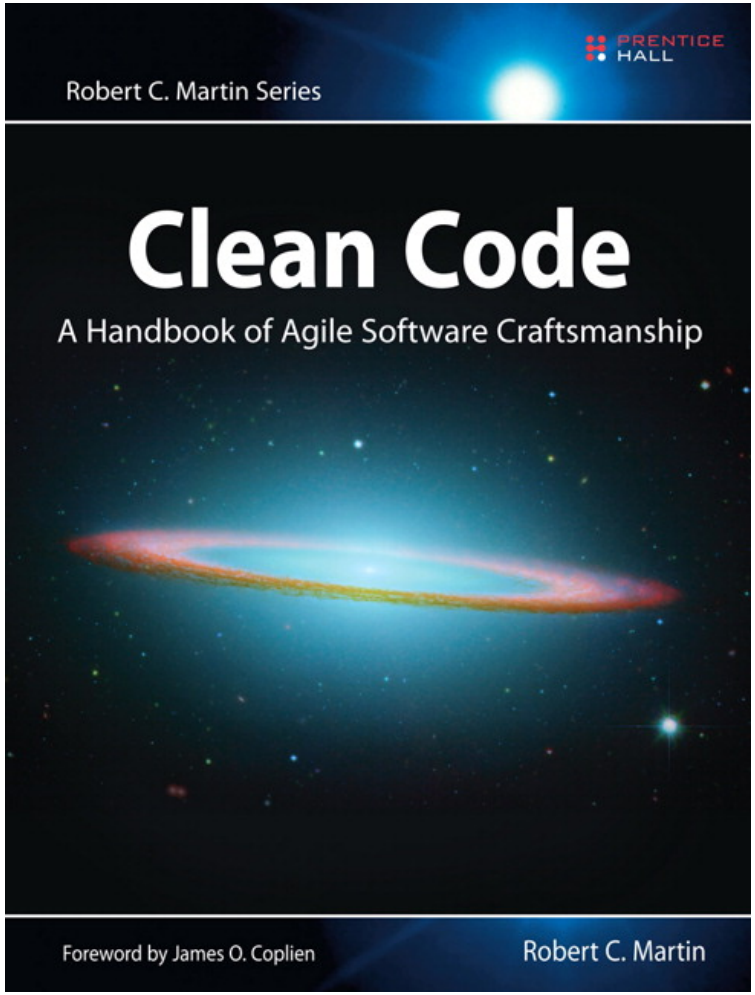
Classes

- Encapsuladas/Coesas
- Pequenas
  - Ex: Classes grandes são sinal de muita responsabilidade

Perguntas

Hugo Corbucci
hugo@agilcoop.org.br

Referência


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin