# Clean Code e Clean Architecture

Rodrigo Branas

Porque estamos aqui?

# Porque o projeto que você trabalha tem o design e a arquitetura que tem?

1. Não sei, já estava assim quando eu cheguei
2. Copiamos de outro projeto que já existia pra seguir o padrão
3. Seguimos um tutorial que encontramos na internet
4. A documentação do framework disse que era pra fazer assim
5. Não sei exatamente, foi aleatório

# Clean Code

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)
- Era normal mexer em uma coisa e estragar outra

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)
- Era normal mexer em uma coisa e estragar outra
- Tinha mais defeitos para corrigir do que funcionalidades novas pra implementar

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)
- Era normal mexer em uma coisa e estragar outra
- Tinha mais defeitos para corrigir do que funcionalidades novas pra implementar
- Tudo era urgente e tinha que parar uma coisa pra resolver outra o tempo todo

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)
- Era normal mexer em uma coisa e estragar outra
- Tinha mais defeitos para corrigir do que funcionalidades novas pra implementar
- Tudo era urgente e tinha que parar uma coisa pra resolver outra o tempo todo
- Ninguém tinha coragem de fazer deploy na sexta-feira ou em véspera de feriado

# Quem já trabalhou em um projeto onde:

- O código era tão bagunçado que você não sabia nem por onde começar
- Existiam partes do projeto que só uma pessoa sabia mexer (as vezes essa pessoa era você)
- Era normal mexer em uma coisa e estragar outra
- Tinha mais defeitos para corrigir do que funcionalidades novas pra implementar
- Tudo era urgente e tinha que parar uma coisa pra resolver outra o tempo todo
- Ninguém tinha coragem de fazer deploy na sexta-feira ou em véspera de feriado

Muita gente tem uma experiência tão ruim que acaba pedindo demissão...

Alguém aqui já pediu demissão porque não aguentava mais trabalhar no projeto?

A qualidade do código que a gente escreve impacta muito na produtividade

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerCharArray;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

O que você acha desse código? Porque as pessoas escrevem tanto código ruim?

Talvez um dos principais motivos seja a influência de como o código era escrito antigamente

```
READ INPUT TAPE A1, B1, C1;
501 FORMAT A1;
IF (A1) 777, 777, 777
IF (B1) 888, 888, 888
IF (C1) 999, 999, 999
STOP 1
799 S = FLOATF(A1 + B1 + C1) / 2.0
WRITE TO TAPE S
END PROCESS
```

As linguagens de programação e a cultura de desenvolvimento eram diferentes

```
Datei  Bearbeiten  Suchen  Optionen                                  Hilfe
                              JENKEY.DEF
DEFINITONSDATEI FÜR JENKEY-DOS-TASTATURBELEGUNG

Syntax: [<Text><Space><Code><Space><Returnflag>];<Remarks>
        Unterstrich ('_') steht für ein Leerzeichen
        Für Return-Flag : j,J,y,Y,t,T = True, alles andere False
        Code ist die Zahl von 0;xxx (ANSI-ASCII-Tastencode).
        Zeilen mit ';' als erstem Zeichen sind Kommentarzeilen.
        Hinter der gestrichelten Linie gehen die Definitionen los.
-------------------------------------------------------------------

; Gültige Definitionen:
;
A: 30 y                    ; ALT-A     -> A:<Ret>
; B: 48 y                    ; ALT-B     -> B:<Ret>
C: 46 Yo mama, do it.      ; ALT-C     -> C:<Ret>
D: 32 T                    ; ALT-D     -> D:<Ret>
E: 18 y                    ; ALT-E     -> E:<Ret>
MEM_/c_/p 101 Y            ; CTRL-F8   -> MEM /c /p<Ret>
DIR_/p 68 y                ; F10       -> DIR /p<Ret>
DIR_/ad 103 y              ; CTRL-F10  -> DIR /ad<Ret>
DIR_/ad 113 y              ; ALT-F10   -> DIR /ad<Ret>
MS-DOS-Editor   F1=Hilfe  ALT=Menü aktivieren          N 00018:056
```

O ambiente de desenvolvimento não tinha tantas ferramentas e automações

O poder de processamento e armazenamento eram muito mores do que hoje em dia

Deve sempre existir um equilíbrio entre
comportamento e estrutura

O comportamento é o que faz os clientes ganharem ou economizarem dinheiro

A estrutura é o que mantém o comportamento de pé, sem colapsar

Pense na estrutura como: métodos, classes, módulos, serviços e principalmente a relação entre cada uma delas

Quanto mais comportamento, mais estrutura será necessária para suportá-lo

Entre o comportamento e estrutura, qual você prioriza no seu dia a dia?

Existem dois perfis nos extremos, nenhum dos dois é bom, o ideal é o equilíbrio

Nós passamos a maior parte do tempo
lendo o código, não escrevendo

Quanto pior for o código, mais chato, braçal e desgastante será o nosso dia a dia

Quanto pior for o código, mais vamos nos sentir atrasado, pressionado e improdutivo

Quanto pior for o código, vai existir mais lugar para os defeitos existirem e se esconderem

Quanto pior for o código, mais lenta será a evolução do produto

Quanto pior for o código, mais os clientes serão impactados e menos satisfeitos vão ficar

Quanto pior for o código, menos competitivo no mercado vamos ficar

No fim, a empresa toda é impactada

Muitas empresas vão à falência com um código bem feito, mas poucas tem sucesso ao longo do tempo com um código mal escrito

Como medir a qualidade do código?

**Linhas de código?**

Número de métodos?

Número de classes?

Tamanho dos método?

Complexidade?

???

# The only valid measurement of code quality: WTFs/minute

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

WTF?

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

WTF?

WTF?

```
2445            if ((lowerChar == Character.ERROR) ||
2446                (lowerChar >= Character.MIN_SUPPLEMENTARY_CODE_POINT)) {
2447                if (lowerChar == Character.ERROR) {
2448                    lowerCharArray =
2449                        ConditionalSpecialCasing.toLowerCaseCharArray(this, i, locale);
2450                } else if (srcCount == 2) {
2451            resultOffset += Character.toChars(lowerChar, result, i + resultOffset) - srcCo
2452            continue;
2453                } else {
2454            lowerCharArray = Character.toChars(lowerChar);
2455        }
2456
2457                /* Grow result if needed */
2458                int mapLen = lowerCharArray.length;
2459        if (mapLen > srcCount) {
2460                    char[] result2 = new char[result.length + mapLen - srcCount];
2461                    System.arraycopy(result, 0, result2, 0,
2462                        i + resultOffset);
2463                    result = result2;
2464        }
2465                for (int x=0; x<mapLen; ++x) {
2466                    result[i+resultOffset+x] = lowerCharArray[x];
2467                }
2468                resultOffset += (mapLen - srcCount);
2469            } else {
2470                result[i+resultOffset] = (char)lowerChar;
2471            }
2472        }
2473        return new String(0, count+resultOffset, result);
2474    }
2475
```

WTF?

WTF?

WTF?

WTF?

Alguém aqui xinga o código com frequência?

Nenhum projeto começa dessa forma...

As vezes, a única solução é reescrever

Muitas vezes, existe um ponto de não retorno, evite chegar lá, poderá ser muito caro e muito arriscado fazer qualquer mudança

Refatore antes que seja tarde demais

# Refactoring

"Alteração feita na estrutura interna do software para torná-lo mais fácil de ser entendido e menos custoso de ser modificado, sem alterar o seu comportamento observável"

Martin Fowler

Refactoring é um investimento, torna o software sustentável e competitivo

Dia após dia a falta de refactoring
consome o tempo da equipe

Refatore com um propósito, evite refatorar apenas por refatorar

Fique atento as oportunidades

Refatore na hora de adicionar novas funcionalidades

Refatore quando for corrigir um defeito
Foto: Divulg

Refatore quando precisar entender
uma parte do código

Muitas empresas precisa contrair algumas dívidas para crescer e escalar sua operação

Mas cuidado com o aumento do dívida técnico, os juros são bem altos

Como evitar que o problema aconteça?

# Code Smells e Técnicas de Refactoring

Um smell é um sintoma que ocorre dentro do código fonte e que pode ser um indicador de problemas

Nomes estranhos de variáveis, métodos, classes e arquivos

```
5  public class StringUtil {
6      public static String convertAttributeToProperty(String nomeOriginal) {
7          String ret = "";
8          String aux;
9          StringTokenizer token = new StringTokenizer(nomeOriginal, "_");
10         ret = token.nextToken();
11         while (token.hasMoreTokens()) {
12             aux = token.nextToken();
13             aux = String.valueOf(aux.charAt(0)).toUpperCase() + aux.substring(1).to
14             ret += aux;
15         }
16         return ret;
17     }
```

```java
130     */
131     public String getDv43(String numero) {
132
133         int total = 0;
134         int fator = 2;
135
136         int numeros, temp;
137
138         for (int i = numero.length(); i > 0; i--) {
139
140             numeros = Integer.parseInt( numero.substring(i-1,i) );
141
142             temp = numeros * fator;
143             if (temp > 9) temp=temp-9; // Regra do banco NossaCaixa
144
145             total += temp;
146
147             // valores assumidos: 212121...
148             fator = (fator % 2) + 1;
149         }
150
151         int resto = total % 10;
152
153         if (resto > 0)
154             resto = 10 - resto;
155
156         return String.valueOf( resto );
157
158     }
```

# Renomear Variável

```
function calculateDiscount(amount, p) {
    const disc = (amount * p)/100;
    return disc;
}
```

```
1.    function calculateDiscount(amount, p) {
2.        const disc = (amount * p)/100;
3.        return disc;
4.    }
```

```
function calculateDiscount(amount, ) {
    const disc = (amount * )/100;
    return disc;
}
```

```
function calculateDiscount(amount, percentage) {
    const disc = (amount * percentage)/100;
    return disc;
}
```

```
function calculateDiscount(amount, percentage) {
    const disc = (amount * percentage)/100;
    return disc;
}
```

```
function calculateDiscount(amount, percentage) {
    const  = (amount * percentage)/100;
    return ;
}
```

```
function calculateDiscount(amount, percentage) {
    const discount = (amount * percentage)/100;
    return discount;
}
```

# Números mágicos

# Substituir Números Mágicos por Constantes

```
function calculatePotencialEnergy(mass, height) {
    return mass * 9.81 * height;
}
```

```
function calculatePotencialEnergy(mass, height) {
    return mass * 9.81 * height;
}
```

```
function calculatePotencialEnergy(mass, height) {
    return mass *  * height;
}
```

```
function calculatePotencialEnergy(mass, height) {
    return mass * GRAVITY * height;
}
```

```
1.    const GRAVITY = 9.81;
2.
3.    function calculatePotencialEnergy(mass, height) {
4.        return mass * GRAVITY * height;
5.    }
6.
```

# Comentários

```
58          if (nmCache != null && username != null) {
59              // Descriptografa as variáveis
60              nmEntidade = FuncoesUtil.easyDecripto(nmEntidade);
61
62              // Verifica de qual entidade, se deseja recuperar a figura.
63              if (nmEntidade.equalsIgnoreCase("pessoa")) {
64                  // Recupera os dados do request, referentes a entidade em questão.
65                  String idPessoa     = request.getParameter("idPessoa");
66                  String icFuncao     = request.getParameter("icFuncao");
67                  String icTipoPessoa = request.getParameter("icTipoPessoa");
68
69                  // Verifica se existem os parâmetros necessários, no request, referentes a entidade em questão.
70                  if (StringSvc.hasValue(idPessoa) && StringSvc.hasValue(icFuncao)) {
71                      // Descriptografa as variáveis
72                      idPessoa = FuncoesUtil.easyDecripto(idPessoa);
73                      icFuncao = FuncoesUtil.easyDecripto(icFuncao);
74
75                      // Verifica se foi passado o tipo da pessoa, caso tenha, o valor do idPessoa corresponde a u
76                      // professor, aluno, etc. e portando deverá primeiramente obter o código da pessoa correspon
77                      if (StringSvc.hasValue(icTipoPessoa)) {
78                          // Descriptografa as variáveis
79                          icTipoPessoa = FuncoesUtil.easyDecripto(icTipoPessoa);
80
81                          // Recupera o código da pessoa, correspondente a entidade passada (aluno, professor, etc
82                          idPessoa = String.valueOf(getCodigoPessoa(nmCache, username, idPessoa, icTipoPessoa));
83                      }
84
85                      // Verifica se existe o código da pessoa.
86                      if (StringSvc.hasValue(idPessoa)) {
87                          to = new ASPTO();
88                          // Cria um TO, contendo os dados necessários para a recuperação da entidade Pessoa
89                          to.addColumn("idPessoa", "Integer");
90                          // O campo icFuncao indica Tipo da figura a ser recuperada (1 - Assinatura, 2 - Foto)
91                          to.addColumn("icFuncao", "Integer");
92
93                          to.setValue("idPessoa", new Integer(idPessoa));
94                          to.setValue("icFuncao", new Integer(icFuncao));
95                      }
96
97                      // Verifica se existe campos no TO.
98                      if (to != null) {
99                          retorno = getData(nmCache, username, nmEntidade, to);
```

```
40          ActionForward forward           = null;
41          ASPMessageTO messages           = new ASPMessageTO();
42
43          // Verifica se o usuário tem permissão de entrar na página
44          forward = hasPermissao(mapping, request, "quadroHorario.vm", "quadroHorario.titulo");
45          if (forward == null) {
46              // Chama método que recupera o quadro de horário tratando retorno de erros
47              messages = getQuadroHorarioAluno(request);
48
49              // Seta forward
50              forward = mapping.findForward("quadroHorario");
51              saveAllMessages(request, messages);
52          }
53          return forward;
54
55
```

# Introduzir Variável Explicativa

```
function calculateRide(hour, distance) {
    // overnight
    if (hour > 22 || hour < 6) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function calculateRide(hour, distance) {
    // overnight
    if (hour > 22 || hour < 6) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function calculateRide(hour, distance) {
    // overnight
    if () {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function calculateRide(hour, distance) {
    // overnight
    const isOvernight = hour > 22 || hour < 6;
    if (isOvernight) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function calculateRide(hour, distance) {
    const isOvernight = hour > 22 || hour < 6;
    if (isOvernight) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function calculateRide(hour, distance) {
    const isOvernight = hour > 22 || hour < 6;
    if (isOvernight) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

```
function isOvernight (hour) {
      return hour > 22 || hour < 6;
}

function calculateRide(hour, distance) {
    if (isOvernight(hour)) {
        return distance * 3.90;
    } else {
        return distance * 2.10;
    }
}
```

# Código morto

```
3212
3213                    if(to.getStringValue("nrCpf") == null)
3214                        txtLinha.append(";");
3215                    else
3216                        txtLinha.append(to.getStringValue("nrCpf").trim() + ";");
3217
3218                    txtLinha.append(to.getValue("nmAluno") + ";");
3219
3220                    if(to.getStringValue("nrIdentidade") == null)
3221                        txtLinha.append(";");
3222                    else
3223                        txtLinha.append(to.getStringValue("nrIdentidade").trim() + ";");
3224
3225                    txtLinha.append(to.getValue("idDeficienciaFisica") + ";");
3226                    txtLinha.append(to.getValue("idDeficienciaVisual") + ";");
3227                    txtLinha.append(to.getValue("idDeficienciaAuditiva") + ";");
3228
3229 /                  txtLinha.append(request.getParameter("cdHabilitacaoEnade") + ";");
3230 /                  if(request.getParameter("icAlunos") == "1")
3231 /                      txtLinha.append("0" + ";");
3232 /                  else
3233 /                      txtLinha.append("1" + ";");
3234 /                  if(to.getIntegerValue("idSexo") == 0)
3235 /                      txtLinha.append(";");
3236 /                  else
3237 /                      txtLinha.append(to.getValue("idSexo") + ";");
3238
3239                    if(to.getIntegerValue("cdCep") == 0)
3240                        txtLinha.append(";");
3241                    else
3242                        txtLinha.append(to.getValue("cdCep") + ";");
3243
3244                    if(to.getStringValue("dsLogradouro") == null)
3245                        txtLinha.append(";");
3246                    else {
3247                        if(to.getStringValue("dsLogradouro").trim().length() > 60)
3248                            txtLinha.append(to.getStringValue("dsLogradouro").trim().substring(0,60) + ";");
3249                        else
3250                            txtLinha.append(to.getValue("dsLogradouro") + ";");
3251                    }
3252
3253                    if(request.getParameter("numero") == null)
```

```
1088        // Verifica dados obrigatórios das pastas (curso e critério).
1089        if (StringSvc.hasValue(janelaOrigem)) {
1090            if (janelaOrigem.equalsIgnoreCase("registro")) {
1091                messages.addMessages(verificaDadosObrigatoriosPastaRegistro(actionForm, request, metodo));
1092            } else if (janelaOrigem.equalsIgnoreCase("assunto")) {
1093                 //messages.addMessages(verificaDadosObrigatoriosPastaAssunto(actionForm, request, metodo));
1094            } else if (janelaOrigem.equalsIgnoreCase("anexo")) {
1095                 //messages.addMessages(verificaDadosObrigatoriosPastaAnexo(actionForm, request, metodo));
1096            }
1097        }
1098        return messages;
1099    }
1100
1101    /**
```

```
141             outStream.write(fileBlob);
142         } else {
143             response.setContentType("text/html");
144             outStream.println("<HTML>");
145             outStream.println("<HEAD>");
146             outStream.println("    <META NAME=SERVERLANGUAGE CONTENT=JavaScript HTTP-EQUIV=PowerSiteData>");
147             outStream.println("</HEAD>");
148             outStream.println("<HEAD>");
149             outStream.println("    <TITLE>Download de Arquivos</TITLE>");
150             outStream.println("    <LINK HREF=\"./portal/css/hsEstilos.css\" REL=stylesheet>");
151             outStream.println("    <SCRIPT LANGUAGE=javascript SRC=\"./portal/js/funcoes.js\" SCRIPT></SCRIPT>");
152             outStream.println("</HEAD>");
153             outStream.println("<BODY>");
154             outStream.println("<SCRIPT LANGUAGE=\"JavaScript1.2\" TYPE=\"text/javascript\" SRC=\"./portal/js/cframe.js\"
155             //      outStream.println("<a name=\"top\"></a>");
156             //      outStream.println("<table width=\"595\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\" align=\"cen
157             //      outStream.println("    <tr> ");
158             //      outStream.println("<TD width=27><IMG height=39 src=\"./portal/images/mn_inicio.gif\" width=27></TD>"
159             //      outStream.println("<TD class=mnAzul background=\"./portal/images/mn_fundo.gif\" width=403>");
160             //      outStream.println((titlePage != null ? titlePage : "MANUTENÇÂO :: <STRONG>Geração de arquivo</STRONG
161             //      outStream.println("            <TD align=\"right\" background=\"./portal/images/mn_fundo.gif\" width
162             //      outStream.println("</tr>");
163             //      outStream.println("</table>");
164             //      outStream.println("    <TABLE border=0 align=center cellSpacing=0 cellPadding=0 width=595>");
165             //      outStream.println("        <TBODY>");
166             //      outStream.println("        <TR>");
167             //      outStream.println("            <TD><IMG src=\"./portal/images/j_fundo2.gif\"> </TD>");
168             //      outStream.println("        </TR>");
169             //      outStream.println("        <TR>");
170             //      outStream.println("            <TD class=txtNormal width=462><br>" + messageResult + "</TD>");
171             //      outStream.println("        </TR>");
172             //      outStream.println("        <TR>");
173             //      outStream.println("            <TD><IMG src=\"./portal/images/j_fundo2.gif\"> </TD>");
174             //      outStream.println("        </TR>");
175             //      outStream.println("        </TBODY>");
176             //      outStream.println("    </TABLE>");
177             outStream.println("<script>");
178             outStream.println("alert('" + upload.strTran(messageResult, "\n", "") + "')");
179             outStream.println("</script>");
180             outStream.println("</body>");
181             outStream.println("</HTML>");
182         }
183     }
```

```
220         }
221
222         if ((new Integer(icDigitacao)).intValue() == 1) {
223             // Fazer dentro de if para não criar uma referencia ao invés de uma nova cópia.
224             if (true) {
225                 ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notas"));
226                 ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
227                 actionForm.set("notasPrimeira", dynaVectorTOClone);
228             }
229
230             if (true) {
231                 ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notasOriginal"));
232                 ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
233                 actionForm.set("notas", dynaVectorTOClone);
234             }
235         } else {
236             ASPDynaTO[] dynaVectorTO      = ((ASPDynaTO[])actionForm.get("notasPrimeira"));
237             ASPDynaTO[] dynaVectorTOClone = dynaVectorTO.clone();
238             actionForm.set("notas", dynaVectorTOClone);
```

Apagar o código

Linhas em branco

```java
130     */
131     public String getDv43(String numero) {
132
133         int total = 0;
134         int fator = 2;
135
136         int numeros, temp;
137
138         for (int i = numero.length(); i > 0; i--) {
139
140             numeros = Integer.parseInt( numero.substring(i-1,i) );
141
142             temp = numeros * fator;
143             if (temp > 9) temp=temp-9; // Regra do banco NossaCaixa
144
145             total += temp;
146
147             // valores assumidos: 212121...
148             fator = (fator % 2) + 1;
149         }
150
151         int resto = total % 10;
152
153         if (resto > 0)
154             resto = 10 - resto;
155
156         return String.valueOf( resto );
157
158     }
```

# Apagar as linhas em branco

# Código duplicado

# Extrair Método

```
function calculatePenalty (amount, percentage) {
  if (today.getTime() < dueDate.getTime()) return 0;
  return (amount * percentage)/100;
}

function calculateInterest(amount, percentage, dueDate) {
  if (today.getTime() < dueDate.getTime()) return 0;
  const dueDays = (today.getTime() - dueDate.getTime())/(1000*60*60*24)
  return (amount * percentage * dueDays)/100;
}
```

```
function calculatePenalty (amount, percentage) {
  if (today.getTime() < dueDate.getTime()) return 0;
  return (amount * percentage)/100;
}

function calculateInterest(amount, percentage, dueDate) {
  if (today.getTime() < dueDate.getTime()) return 0;
  const dueDays = (today.getTime() - dueDate.getTime())/(1000*60*60*24)
  return (amount * percentage * dueDays)/100;
}
```

```
function calculatePenalty (amount, percentage) {
  if () return 0;
  return (amount * percentage)/100;
}

function calculateInterest(amount, percentage, dueDate) {
  if () return 0;
  const dueDays = (today.getTime() - dueDate.getTime())/(1000*60*60*24)
  return (amount * percentage * dueDays)/100;
}
```

```
function isOverdue (dueDate) {
  return today.getTime() > dueDate.getTime();
}

function calculatePenalty (amount, percentage, dueDate) {
  if (!isOverdue(dueDate)) return 0;
  return (amount * percentage)/100;
}

function calculateInterest(amount, percentage, dueDate) {
  if (!isOverdue(dueDate)) return 0;
  const dueDays = (today.getTime() - dueDate.getTime())/(1000*60*60*24)
  return (amount * percentage * dueDays)/100;
}
```

# Variáveis declaradas longe da utilização

# Mover variáveis

```
function calculateFactor () {
  let length = 64;
  let total, sum = 0;
  let multiplier = 10;

  for (let i = 0; i < length; i++) {
    sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total, sum = 0;
  let multiplier = 10;

  for (let i = 0; i < length; i++) {
    sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total = 0;
  let sum = 0;
  let multiplier = 10;

  for (let i = 0; i < length; i++) {
    sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total = 0;
  let multiplier = 10;

  for (let i = 0; i < length; i++) {
    sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total = 0;
  let multiplier = 10;

  for (let i = 0; i < length; i++) {
    const sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total = 0;

  for (let i = 0; i < length; i++) {
    const sum = i * length;
    if (sum > 10> total += sum;
  }

  return total * multiplier;
}
```

```
function calculateFactor () {
  let length = 64;
  let total = 0;
  for (let i = 0; i < length; i++) {
    const sum = i * length;
    if (sum > 10) total += sum;
  }
  let multiplier = 10;
  return total * multiplier;
}
```

# Condições confusas

```
2875            }
2876        } catch(Exception e){
2877            e.printStackTrace();
2878        }
2879
2880        // Turmas
2881        xmlLista = new XMLSvc();
2882        try {
2883            comp = (AcademicoDelegate) Funcoes.createDelegate(AcademicoDelegateHome.JNDI_NAME, AcademicoDelegateHome
2884            ResultTO result = comp.execute(nmCache, cdUsuario, "exportacaoCursoTurma", new Object[]{whereClause.toSt
2885
2886            if (result.ok) {
2887                list = (ASPVector)result.value;
2888                if (list != null && list.size() > 0) {
2889                    for (int i = 0; i < list.size(); i++) {
2890                        XMLSvc xmlLinha = new XMLSvc();
2891                        ASPTO to = (ASPTO)list.get(i);
2892                        String codigoTurma = to.getValue("cdCursoInstituicao") + "-" + to.getValue("cdTurma").toStri
2893                        xmlLinha.addTag("CODIGO", codigoTurma);
2894                        if (StringSvc.hasValue(to.getStringValue("dsTurma"))){
2895                            xmlLinha.addTag("NOME", to.getStringValue("dsTurma"));
2896                        }
2897                        // Os campos DIAS e AULAS do layout da TURMA estão sendo desprezados  na integração
2898                        //xmlLinha.addTag("DIAS", "");
2899                        //xmlLinha.addTag("AULAS", "");
2900                        xmlLinha.addTag("ABREVIATURA", codigoTurma);
2901                        xmlLinha.addTag("TURNO", to.getStringValue("dsTurno"));
2902                        xmlLista.addTag("REGISTRO", xmlLinha.getFragment());
2903                    }
2904                    xml.addTag("TURMAS", xmlLista.getFragment());
2905                }
2906            }
2907        } catch(Exception e){
2908            e.printStackTrace();
2909        }
2910
2911        // Professores
2912        xmlLista = new XMLSvc();
2913        list = getListTO(ProfessorLocalHome.JNDI_NAME, nmCache, cdUsuario, whereClause.toString(), "professor.cd_pro
2914        if (list != null && list.size() > 0) {
2915            for (int i = 0; i < list.size(); i++) {
2916                XMLSvc xmlLinha = new XMLSvc();
2917                ASPTO to = (ASPTO)list.get(i);
```

```
286                 }
287                 //Pesquisa os Professores realcionados as perguntas
288                 searchPesquisaPerguntaProfessor(request,pesquisa.getIntegerValue("cdPesquisa"),actionForm);
289             }
290         }
291
292
293         if (pesquisaRestrita.size() > 0) {
294             ASPVector pesquisas = pesquisaRestrita;
295
296             if(quadroHorario.size() > 0){
297                 for (int i = 0; i < quadroHorario.size(); i++) {
298                     QuadroHorarioAlunoTO quadroHorarioAluno = (QuadroHorarioAlunoTO) quadroHorario.get(i);
299
300                     for (int j = 0; j < pesquisas.size(); j++) {
301                         PesquisaSocioEconomicaTO pesquisa = (PesquisaSocioEconomicaTO) pesquisas.get(j);
302                         ASPVector questionario = pesquisa.getASPVectorValue("questionario");
303                         ASPVector questionarioCorrigido = new ASPVector();
304
305                         boolean possuiProfessor = false;
306                         for (int k = 0; k < questionario.size(); k++) {
307                             PesquisaPerguntaQuestionarioTO perguntaQuestionario = (PesquisaPerguntaQuestionarioTO
308                             if(perguntaQuestionario.getIntegerValue("cdProfessor") != null){
309                                 possuiProfessor = true;
310                                 break;
311                             }
312                         }
313                         if(possuiProfessor){
314                             for (int k = 0; k < questionario.size(); k++) {
315                                 PesquisaPerguntaQuestionarioTO perguntaQuestionario = (PesquisaPerguntaQuestionar
316                                 if(perguntaQuestionario.getIntegerValue("cdProfessor").intValue() == Integer.pars
317                                     questionarioCorrigido.add(perguntaQuestionario);
318                                 }
319                             }
320                             if(questionarioCorrigido.size() > 0){
321                                 pesquisa.setValue("questionario", questionarioCorrigido);
322                             }
323                         }
324                     }
325                 }
326                 request.getSession().setAttribute("ASPPesquisas", pesquisas);
327             }else{
```

```
159                     result = comp.execute(nmCache, cdUsuario, "pesquisaNotasParciais", new Object[] { whereClause, ""  }, NotaLocal
160                     MessageUtil.actionMessages(messages, result.bufferMessage);
161                     if (result.ok) {
162                         list = (ASPVector)result.value;
163                         listCarga = (ASPVector)result.value;
164                         if(list.size() > 0){
165                             int cdDisciplinaLista = 0;
166                             to = list.getTO(0);
167                             cdDisciplinaLista = to.getIntegerValue("cdDisciplina").intValue();
168                             dynaTO = ASPDynaTO.getInstance(NotasParciaisTO.getOriginalMetaData());
169
170                             if(to.getIntegerValue("qtCargaHoraria") != null){
171                                 cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
172                                 pctFreque = pctFreque + to.getDoubleValue("pctFrequencia");
173                             }else{
174                                 cargaHoraria = 0;
175                                 pctFreque = new Double(0);
176                             }
177                             //Soma de carga horária de todos os periodos habilitados no acesso modulo agregado
178                             for (int i = 1; i < list.size(); i++) {
179                                 to = list.getTO(i);
180                                 if(to.getIntegerValue("qtCargaHoraria") != null){
181                                     if(cdDisciplinaLista == to.getIntegerValue("cdDisciplina").intValue() ){
182                                         if(to.getIntegerValue("cdPeriodo").intValue() <= cdPeriodo){
183                                             cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
184                                         }
185                                     }else{
186                                         toCarga = list.getTO(i - 1);
187                                         toCarga.setValue("qtCargaHoraria", cargaHoraria);
188                                         for (int j = 0; j < listCarga.size(); j++) {
189                                             toCarga = list.getTO(j);
190                                             if(cdDisciplinaLista == toCarga.getIntegerValue("cdDisciplina").intValue() ){
191                                                 toCarga.setValue("qtCargaHoraria", cargaHoraria);
192                                             }
193                                         }
194                                         cargaHoraria = 0;
195                                         cargaHoraria = cargaHoraria + to.getIntegerValue("qtCargaHoraria").intValue();
196                                         cdDisciplinaLista = to.getIntegerValue("cdDisciplina").intValue();
197                                     }
198                                 }else{
199                                     to.setValue("qtCargaHoraria", new Integer(0));
200                                 }
201                             }
202                             for (int j = 0; j < listCarga.size(); j++) {
203                                 toCarga = list.getTO(j);
204                                 if(cdDisciplinaLista == toCarga.getIntegerValue("cdDisciplina").intValue() ){
205                                     toCarga.setValue("qtCargaHoraria", cargaHoraria);
206                                 }
207                             }
208
209                             //Soma do percentual de frequencia de até periodo
210                             cdDisciplinaLista = new Integer(0);
211                             for (int j = 0; j < list.size(); j++) {
```

```
1266        public int read() throws java.io.IOException {
1267            // Do we need to get data?
1268            if( position < 0 ) {
1269                if( encode ) {
1270                    byte[] b3 = new byte[3];
1271                    int numBinaryBytes = 0;
1272                    for( int i = 0; i < 3; i++ ) {
1273                        try {
1274                            int b = in.read();
1275
1276                            // If end of stream, b is -1.
1277                            if( b >= 0 ) {
1278                                b3[i] = (byte)b;
1279                                numBinaryBytes++;
1280                            }   // end if: not end of stream
1281
1282                        }   // end try: read
1283                        catch( java.io.IOException e ) {
1284                            // Only a problem if we got no data at all.
1285                            if( i == 0 )
1286                                throw e;
1287
1288                        }   // end catch
1289                    }   // end for: each needed input byte
1290
1291                    if( numBinaryBytes > 0 ) {
1292                        encode3to4( b3, 0, numBinaryBytes, buffer, 0, options );
1293                        position = 0;
1294                        numSigBytes = 4;
1295                    }   // end if: got data
1296                    else {
1297                        return -1;
1298                    }   // end else
1299                }   // end if: encoding
1300
1301                // Else decoding
1302                else {
1303                    byte[] b4 = new byte[4];
1304                    int i = 0;
1305                    for( i = 0; i < 4; i++ ) {
1306                        // Read four "meaningful" bytes:
1307                        int b = 0;
```

# Remover condição aninhada por cláusulas guarda

```
1.      async function sendEmailCampaign(campaign) {
2.        if (!campaign.sent) {
3.          const recipients = await repository.getRecipients(campaign.id);
4.          if (hasEmailQuota(recipients.length)) {
5.            for (const recipient of recipients) {
6.              if (!isBouncedRecipient(recipient)) {
7.                if (isAllowedRecipient(recipient)) {
8.                  await send(campaign, recipient);
9.                }
10.             }
11.           }
12.         }
13.       }
14.     }
```

```
1.      async function sendEmailCampaign(campaign) {
2.        if (campaign.sent) return;
3.        const recipients = await repository.getRecipients(campaign.id);
4.        if (hasEmailQuota(recipients.length)) {
5.          for (const recipient of recipients) {
6.            if (!isBouncedRecipient(recipient)) {
7.              if (isAllowedRecipient(recipient)) {
8.                await send(campaign, recipient);
9.              }
10.           }
11.         }
12.       }
13.     }
```

```
1.      async function sendEmailCampaign(campaign) {
2.        if (campaign.sent) return;
3.        const recipients = await repository.getRecipients(campaign.id);
4.        if (!hasEmailQuota(recipients.length)) return;
5.        for (const recipient of recipients) {
6.          if (!isBouncedRecipient(recipient)) {
7.            if (isAllowedRecipient(recipient)) {
8.              await send(campaign, recipient);
9.            }
10.         }
11.       }
12.     }
```

```
1.    async function sendEmailCampaign(campaign) {
2.      if (campaign.sent) return;
3.      const recipients = await repository.getRecipients(campaign.id);
4.      if (!hasEmailQuota(recipients.length)) return;
5.      for (const recipient of recipients) {
6.        if (isBouncedRecipient(recipient)) continue;
7.        if (isAllowedRecipient(recipient)) {
8.          await send(campaign, recipient);
9.        }
10.     }
11.   }
```

```
1.    async function sendEmailCampaign(campaign) {
2.      if (campaign.sent) return;
3.      const recipients = await repository.getRecipients(campaign.id);
4.      if (!hasEmailQuota(recipients.length)) return;
5.      for (const recipient of recipients) {
6.        if (isBouncedRecipient(recipient)) continue;
7.        if (!isAllowedRecipient(recipient)) continue;
8.        await send(campaign, recipient);
9.      }
10.   }
```

# Introduzir comando ternário

```
function getClass(element) {
  if (element.buttonClass) {
    return element.buttonClass;
  } else {
    return "btn-lg";
  }
}
```

```
function getClass(element) {
  return (element.buttonClass) ? element.buttonClass : "btn-lg";
}
```

# Remover comando ternário

```
function calculateTax(amount, last12monthRevenue) {
  return (last12monthRevenue <= 120000) ? amount * 0.06 :
((last12monthRevenue <= 240000) ? amount * 0.08 :
((last12monthRevenue <= 360000) ? amount * 0.12 : amount *
0.15));
}
```

```
function calculateTax(amount, last12monthRevenue) {
  if (last12monthRevenue <= 120000) {
    return amount * 0.06;
  }
  if (last12monthRevenue <= 240000) {
    return amount * 0.08;
  }
  if (last12monthRevenue <= 360000) {
    return amount * 0.12;
  }
  return amount * 0.15;
}
```

# Falta de tratamento de exceções

```
1379            int b;
1380            for( i = 0; i < len; i++ ) {
1381                b = read();
1382
1383                //if( b < 0 && i == 0 )
1384                //    return -1;
1385
1386                if( b >= 0 )
1387                    dest[off + i] = (byte)b;
1388                else if( i == 0 )
1389                    return -1;
1390                else
1391                    break; // Out of 'for' loop
1392            }   // end for: each byte read
1393            return i;
1394        }   // end read
1395
```

# Substituir código de erro por exceção

```
1.      class WithdrawMoney {
2.
3.        async execute (account, amount) {
4.          const result = await debitAccount.execute(account, amount);
5.          if (result === 1) {
6.            try {
7.              await atm.dispenseMoney(amount);
8.            } catch (e) {
9.              await creditAccount.execute(account, amount);
10.           }
11.         }
12.         if (result === -1) {
13.           await atm.showMessage("Insuficient balance");
14.         }
15.         if (result === -2) {
16.           await atm.showMessage("Card has expired");
17.         }
18.       }
19.     }
```

```
class WithdrawMoney {

  async execute (account, amount) {
    try {
      await debitAccount.execute(account, amount);
      await atm.dispenseMoney(amount);
    } catch (e) {
      await creditAccount.execute(account, amount);
      await atm.showMessage(e.message);
    }
  }
}
```

Você tem coragem de refatorar o código sem testes?

# Test-Driven Development

Os testes automatizados são a única forma que temos para garantir que o código funciona e continuará funcionando

Isso quer dizer que não faz sentido ter testes manuais?

Os testes manuais são importantes, principalmente para a aceitação por parte dos usuários, de usabilidade e acessibilidade, mas cuidado, esses testes não garantem que não existirá a regressão ao longo do tempo, sempre que o código mudar

Ter testes automatizados não é uma garantia de que não vão existir defeitos, nem com 100% de cobertura

Porque muita gente tenta escrever testes e se frustra, pelo menos no início?

Escrever os testes requer muita disciplina

Existe muita ansiedade, queremos sempre ver tudo "funcionando"

Nos acostumamos a começar pela tela ou pelo banco de dados, não pelo domínio

Não temos informações suficientes para começar a desenvolver

Muitas vezes o design e a arquitetura não favorecem a automação dos testes

Achar que o único teste que é importante é o teste de unidade

O que é um teste automatizado?

# Dado um conjunto de entradas, quando algo acontecer a saída deve suprir as expectativas

Given/Arrange: Definição de todas as informações necessárias para executar o comportamento que será testado

When/Act: Executar o comportamento

Then/Assert: Verificar o que aconteceu após a execução, comparando as informações retornadas com a expectativa que foi criada

BREITLING
ENDURANCE
VINFAST
HOKA
RŌKA
1ST
2ND
3RD
ATHLETIC BREWING CO
IRONMAN
LAKE PLACID
FIRST

# FIRST

- Fast: Os testes devem rodar rápido

# FIRST

- Fast: Os testes devem rodar rápido
- Independent: Não deve existir dependência entre os testes, eles devem poder ser executados de forma isolada

# FIRST

- Fast: Os testes devem rodar rápido
- Independent: Não deve existir dependência entre os testes, eles devem poder ser executados de forma isolada
- Repeatable: O resultado deve ser o mesmo independente da quantidade de vezes que seja executado

# FIRST

- Fast: Os testes devem rodar rápido
- Independent: Não deve existir dependência entre os testes, eles devem poder ser executados de forma isolada
- Repeatable: O resultado deve ser o mesmo independente da quantidade de vezes que seja executado
- Self-validating: O próprio teste deve ter uma saída bem definida que é válida ou não fazendo com que ele passe ou falhe

# FIRST

- Fast: Os testes devem rodar rápido
- Independent: Não deve existir dependência entre os testes, eles devem poder ser executados de forma isolada
- Repeatable: O resultado deve ser o mesmo independente da quantidade de vezes que seja executado
- Self-validating: O próprio teste deve ter uma saída bem definida que é válida ou não fazendo com que ele passe ou falhe
- Timely: Os testes devem ser escritos antes do código-fonte

# Como funciona o Test-Driven Development?

TDD é um método para construir software, não para testá-lo

RED
1. Write a test that fails
TDD
3. Eliminate redundancy
REFACTOR
GREEN
2. Make the code work

"TDD is a way of managing fear during programming"

Kent Beck

# TheThreeRulesOfTdd [add child]

## The Three Laws of TDD.

Over the years I have come to describe Test Driven Development in terms of three simple rules. They are:

1. You are not allowed to write any production code unless it is to make a failing unit test pass.
2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.

You must begin by writing a unit test for the functionality that you intend to write. But by rule 2, you can't write very much of that unit test. As soon as the unit test code fails to compile, or fails an assertion, you must stop and write production code. But by rule 3 you can only write the production code that makes the test compile or pass, and no more.

If you think about this you will realize that you simply cannot write very much code at all without compiling and executing something. Indeed, this is really the point. In everything we do, whether writing tests, writing production code, or refactoring, we keep the system executing at all times. The time between running tests is on the order of seconds, or minutes. Even 10 minutes is too long.

Too see this in operation, take a look at The Bowling Game Kata.

Now most programmers, when they first hear about this technique, think: "*This is stupid!*" "*It's going to slow me down, it's a waste of time and effort, It will keep me from thinking, it will keep me from designing, it will just break my flow.*" However, think about what would happen if you walked in a room full of people working this way. Pick any random person at any random time. A minute ago, all their code worked.

Let me repeat that: *A minute ago all their code worked!* And it doesn't matter who you pick, and it doesn't matter when you pick. **A minute ago all their code worked!**

If all your code works every minute, how often will you use a debugger? Answer, not very often. It's easier to simply hit ^Z a bunch of times to get the code back to a working state, and then try to write the last minutes worth again. And if you aren't debugging very much, how much time will you be saving? How much time do you spend debugging now? How much time do you spend fixing bugs once you've debugged them? What if you could decrease that time by a significant fraction?

But the benefit goes far beyond that. If you work this way, then every hour you are producing several tests. Every day dozens of tests. Every month hundreds of tests. Over the course of a year you will write thousands of tests. You can keep all these tests and run them any time you like! When would you run them? All the time! Any time you made any kind of change at all!

Why don't we clean up code that we know is messy? We're afraid we'll break it. But if we have the tests, we can be reasonably sure that the code is not broken, or that we'll detect the breakage immediately. If we have the tests we become fearless about making changes. If we see messy code, or an unclean structure, we can clean it without fear. Because of the tests, the code becomes malleable again. Because of the tests, software becomes soft again.

But the benefits go beyond that. If you want to know how to call a certain API, there is a test that does it. If you want to know how to create a certain object, there is a test that does it. Anything you want to know about the existing system, there is a test that demonstrates it. The tests are like little design documents, little coding examples, that describe how the system works and how to use it.

Have you ever integrated a third party library into your project? You got a big manual full of nice documentation. At the end there was a thin appendix of examples. Which of the two did you read? The examples of course! That's what the unit tests are! They are the most useful part of the documentation. They are the living examples of how to use the code. They are design documents that are hideously detailed, utterly unambiguous, so formal that they execute, and they cannot get out of sync with the production code.

But the benefits go beyond that. If you have ever tried to add unit tests to a system that was already working, you probably found that it wasn't much fun. You likely found that you either had to change portions of the design of the system, or cheat on the tests; because the system you were trying to write tests for was not designed to be testable. For example, you'd like to test some function 'f'. However, 'f' calls another function that deletes a record from the database. In your test, you don't want the record deleted, but you don't have any way to stop it. The system wasn't

# Three Laws of TDD

Você não pode escrever nenhum código até ter escrito um teste que detecte uma possível falha.

Você não pode escrever mais testes de unidade do que o suficiente para detectar a falha.

Você não pode escrever mais código do que o suficiente para passar nos testes.

(Robert C. Martin)

Como começar?

Dê o primeiro passo, ou seja, crie um exemplo que as pessoas na equipe possam seguir e utilizar como base, ninguém começa com 100% de cobertura

Foque no que tem mais risco e muda com mais frequência, o sucesso não está em ter 100% de cobertura mas sim em automatizar os testes daquilo que dá mais retorno

Usar test patterns como um stub ou mock não é necessariamente ruim, em muitos casos é algo necessário, mas só conseguir testar se utilizar esses recursos pode indicar que o design precisa melhorar

Usar TDD pode parecer muito complicada no início, mas certamente faz sentido definir o cenário desejado antes e escrever o código

Use o tipo de teste que for mais viável para a sua realidade, geralmente são os testes de integração ou E2E

Por fim, lembre-se na maior parte das vezes é uma questão de disciplina não de experiência ou conhecimento técnico