



Serviço Nacional de Aprendizagem Industrial
PELO FUTURO DO TRABALHO

SÉRIE TI - SOFTWARE

LÓGICA DE PROGRAMAÇÃO

LEGIBILIDADE DE CÓDIGO-FONTE:
PADRÕES DE NOMENCLATURA E
CONVENÇÕES DE LINGUAGEM



CONFEDERAÇÃO NACIONAL DA INDÚSTRIA – CNI

Robson Braga de Andrade
Presidente

GABINETE DA PRESIDÊNCIA

Teodomiro Braga da Silva
Chefe do Gabinete - Diretor

DIRETORIA DE EDUCAÇÃO E TECNOLOGIA - DIRET

Rafael Esmeraldo Lucchesi Ramacciotti
Diretor de Educação e Tecnologia

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL - SENAI

Robson Braga de Andrade
Presidente do Conselho Nacional

SENAI – Departamento Nacional

Rafael Esmeraldo Lucchesi Ramacciotti
Diretor-Geral

Julio Sergio de Maya Pedrosa Moreira
Diretor-Adjunto

Gustavo Leal Sales Filho
Diretor de Operações



Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

SÉRIE TI - SOFTWARE

LÓGICA DE PROGRAMAÇÃO

**LEGIBILIDADE DE CÓDIGO-FONTE:
PADRÕES DE NOMENCLATURA E
CONVENÇÕES DE LINGUAGEM**



© 2020. SENAI – Departamento Nacional

© 2020. SENAI – Departamento Regional de Santa Catarina

A reprodução total ou parcial desta publicação por quaisquer meios, seja eletrônico, mecânico, fotocópia, de gravação ou outros, somente será permitida com prévia autorização, por escrito, do SENAI.

Esta publicação foi elaborada pela equipe de Educação a Distância do SENAI de Santa Catarina, com a coordenação do SENAI Departamento Nacional, para ser utilizada por todos os Departamentos Regionais do SENAI nos cursos presenciais e a distância.

SENAI Departamento Nacional

Unidade de Educação Profissional e Tecnológica - UNIEP

SENAI Departamento Regional de Santa Catarina

Gerência de Educação

SENAI

Serviço Nacional de
Aprendizagem Industrial
Departamento Nacional

Sede

Setor Bancário Norte • Quadra 1 • Bloco C • Edifício Roberto
Simonsen • 70040-903 • Brasília – DF • Tel.: (0xx61) 3317-
9001 Fax: (0xx61) 3317-9190 • <http://www.senai.br>

Lista de Ilustrações

Figura 1 - Tela do VisualG – Pseudocódigo programa nada legível	10
Figura 2 - Tela do VisualG – Pseudocódigo programa legível.....	12
Figura 3 - Tela da IDE Eclipse – Primeira proposta objeto pessoa.....	13
Figura 4 - Tela da IDE Eclipse – Segunda proposta objeto pessoa.....	14
Tabela 1 - Interpretação das linhas de código em Portugol Pseudocódigo do programa nada legível...	11
Tabela 2 - Interpretação das linhas de código em Java da Primeira proposta objeto pessoa	14

Sumário

Legibilidade de Código-Fonte: Padrões de nomenclatura e convenções de linguagem	9
Apresentação	9
Definição	9
Estrutura	10
Aplicação na indústria	12
Exemplos	13
Palavra do Docente.....	16
Referências.....	17

Legibilidade de Código-Fonte: Padrões de nomenclatura e convenções de linguagem



APRESENTAÇÃO

Olá! Sejam bem-vindos aos estudos de Legibilidade de Código-Fonte: Padrões de Nomenclatura e Convenções de Linguagem!

Muitas vezes, você pode considerar que as regras, as convenções, os padrões podem parecer um pouco exagerados, inconvenientes e que dá um sentimento que se está perdendo tempo ao adotar um conjunto de práticas para permanecer conforme um padrão estabelecido.

Mas, na verdade, as boas práticas no universo da programação (assim como em vários outros) são extremamente importantes, porque, ao programar, o profissional precisa ter a plena consciência de que ele não escreve aquelas linhas de código para si, mas sim para outros. Isso significa que aquilo que pode parecer óbvio para uns, para outros programadores talvez não tenha o menor sentido.

É exatamente isso que você estudará nesse tópico, ou seja, as questões de legibilidade de código-fonte. Afinal, o que é legibilidade?

Bons estudos!

DEFINIÇÃO

O termo legibilidade vem de “estado de ser legível”, e legível é algo que, pela clareza e nitidez caligráfica ou tipográfica, pode ser lido com facilidade. Mas, quando se trata desse termo na computação, refere-se à qualidade com que é escrito (definido) o nome das classes, variáveis, métodos etc.

A legibilidade em um programa é a característica que permite que outros programadores que não tiveram contato com a elaboração daquele código consigam entendê-lo, decifrá-lo e mantê-lo. Neste caso, é preciso ver com clareza de que trata aquele código, para que servem as variáveis, quais as utilidades encontradas em determinadas funções etc.

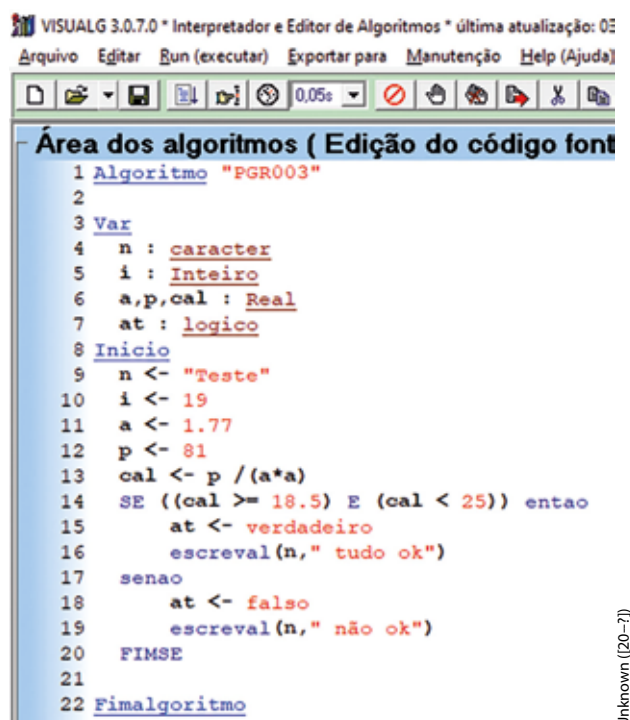
ESTRUTURA

Para entender melhor sobre esses conceitos abordados neste tópico, serão divididas as previsões expostas no título da seguinte forma:

- a) Legibilidade de código-fonte:
 - a. Padrões de nomenclatura;
 - b. Convenções de linguagem.

É possível observar que a legibilidade do código-fonte passa por duas importantes questões: os padrões nas definições de nomes (para classes, variáveis, funções, enuns etc.) e das próprias convenções da linguagem (desde padrões oferecidos nativamente na linguagem até padrões adotados nas comunidades, empresas e metodologias). Mas, o que seriam considerados bons padrões de nomenclatura em uma programação? Como é possível garantir a criação de um código com bom nível de legibilidade?

Imagine que você foi contratado para a vaga de programador em uma determinada *softwarehouse*. Aparentemente o antigo programador não estava se adaptando bem à profissão, desistindo dessa carreira. Na primeira atividade de manutenção e correção das classes trabalhadas pelo antigo colaborador, você se depara com o seguinte cenário:



```
1 Algoritmo "PGR003"
2
3 Var
4 n : caracter
5 i : Inteiro
6 a,p,cal : Real
7 at : logico
8 Inicio
9 n <- "Teste"
10 i <- 19
11 a <- 1.77
12 p <- 81
13 cal <- p / (a*a)
14 SE ((cal >= 18.5) E (cal < 25)) entao
15     at <- verdadeiro
16     escreval(n, " tudo ok")
17 senao
18     at <- falso
19     escreval(n, " não ok")
20 FIMSE
21
22 Fimalgoritmo
```

Figura 1 - Tela do VisualG – Pseudocódigo programa nada legível
Fonte: do Autor (2020)

Logo que começa a ler o programa, você se depara com os seguintes problemas de legibilidade:

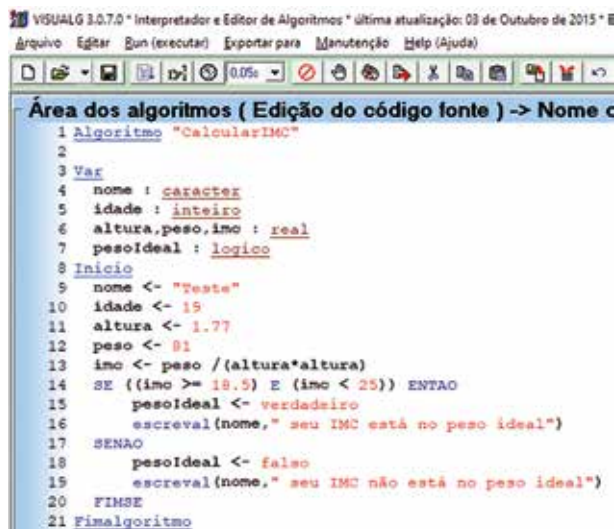


Tabela 1 - Interpretação das linhas de código em Portugol Pseudocódigo do programa nada legível
Fonte: do Autor (2020)

Veja como essa situação pode ser extremamente desconfortável para qualquer profissional ao ter que assumir a manutenção de um determinado código de programação, em que ele perde muito tempo tendo que analisar as linhas para se esgueirar em uma tentativa de adivinhação. Imagine que, nesse caso, você teria que partir para a origem da demanda, tendo que resgatar um documento técnico que foi passado ao antigo programador, para compreender melhor qual foi a solicitação. Neste caso, o documento tinha a seguinte nota técnica:

“Desenvolver um programa que possa definir o nome, idade, altura, peso, para posteriormente calcular o índice de massa corporal da pessoa. Caso a pessoa estiver dentro dos limites de um peso ideal, exibir uma mensagem que a pessoa está “OK” em relação ao seu peso ideal; caso contrário, alertar o usuário que seu IMC está fora dos padrões esperados. Defina valores testes relacionados às variáveis para verificar se a lógica de programação está correta”.

Após a leitura da nota técnica, você se propõe a reformular o código anterior, adotando práticas mais aceitáveis de legibilidade e padrões.

```

1 Algoritmo "CalcularIMC"
2
3 Var
4   nome : caractere
5   idade : inteiro
6   altura, peso, imc : real
7   pesoIdeal : logico
8 Inicio
9   nome <- "Teste"
10  idade <- 15
11  altura <- 1.77
12  peso <- 61
13  imc <- peso / (altura*altura)
14  SE ((imc >= 18.5) E (imc < 25)) ENTAO
15    pesoIdeal <- verdadeiro
16    escreva(nome, " seu IMC está no peso ideal")
17  SENAO
18    pesoIdeal <- falso
19    escreva(nome, " seu IMC não está no peso ideal")
20  FIMSE
21 Fimalgoritmo
  
```

Figura 2 - Tela do VisualG – Pseudocódigo programa legível
Fonte: do Autor (2020)

APLICAÇÃO NA INDÚSTRIA

No setor produtivo, a adoção de padrões é algo muito importante, pois fornece confiabilidade ao processo fabril, garantia de qualidade, escalabilidade, dentre várias outras qualidades, gerando assim um favorecimento de competitividade.

Você, como um futuro profissional qualificado no desenvolvimento de sistemas, deve se preocupar e muito com essa temática. Pense no seguinte cenário: uma indústria moveleira da região deseja passar por uma grande migração tecnológica, procurando revitalizar todo seu parque de Tecnologia da Informação e Comunicação. Para isso, pretende adotar, além da compra de novos equipamentos (computadores, monitores, servidores etc.), linguagens de programação mais modernas e robustas.

A equipe atual de programadores dessa indústria utiliza basicamente linguagem de banco de dados e programação estruturada. Para conseguir migrar para uma nova linguagem de programação agora orientada a objetos, a equipe terá que passar por diversos treinamentos e qualificação tecnológica. Posteriormente a isso, o time terá que definir quais serão os padrões a serem adotados no que se refere às questões de legibilidade de código-fonte da indústria para os profissionais atuais e futuros que vão atuar na programação dos sistemas que serão migrados.

Após um longo debate, a equipe define questões muito importantes, como:

- O Código será desenvolvido sempre em inglês, no que se refere aos nomes de classes, variáveis e métodos. Esse padrão será adotado, porque a empresa pretende criar filiais fora do país e ter a possibilidade de contratar profissionais *home office* em qualquer localidade do mundo.

- b) Todas as classes desenvolvidas deverão ser comentadas tecnicamente nas seguintes questões: autor, data da criação, último atualizador, data da atualização, número do documento técnico que gerou a demanda para criar a classe em questão, objetivo da classe.
- c) Todos os métodos das classes devem estar definidos como ação verbal e nunca em outra conjugação. Ex.: *jump*, *run*, *save*.
- d) Os nomes das classes, atributos, métodos devem ser os mesmos previstos na modelagem do diagrama de classe, fornecido pelo profissional analista de sistemas.

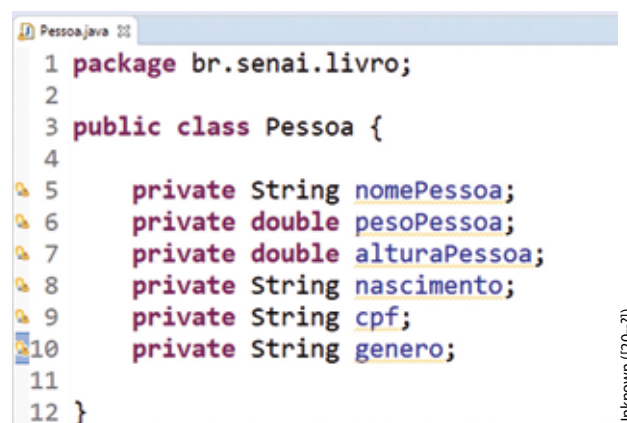
Todas essas questões, quando bem alinhadas e definidas, guiarão a empresa por uma caminhada de atualização muito mais perene e duradoura, ausentando a empresa de diversos inconvenientes de ter que ajustar muito desvio de padrão posteriormente.

É claro que os pontos definidos anteriormente são apenas um fragmento da diversidade de questões que devem ser estabelecidas primariamente em uma migração tecnológica. Porém, serve, de maneira real, como um parâmetro da importância da adoção de padrões de legibilidade de código em um ambiente real.

EXEMPLOS

Para fortalecer mais ainda os conceitos e a importância sobre legibilidade, acompanhe agora alguns exemplos da adoção de padrões em outra linguagem de programação, no caso, em Java.

Imagine que você recebeu a demanda de criar uma classe que represente uma pessoa, que solicitou criar um objeto que possa ser capaz de definir uma pessoa, seu nome, altura, peso, nascimento, CPF e gênero. Tão logo recebeu a nota técnica, partiu para uma primeira tentativa de programação, resultando na seguinte proposta:



```
1 package br.senai.livro;
2
3 public class Pessoa {
4
5     private String nomePessoa;
6     private double pesoPessoa;
7     private double alturaPessoa;
8     private String nascimento;
9     private String cpf;
10    private String genero;
11
12 }
```

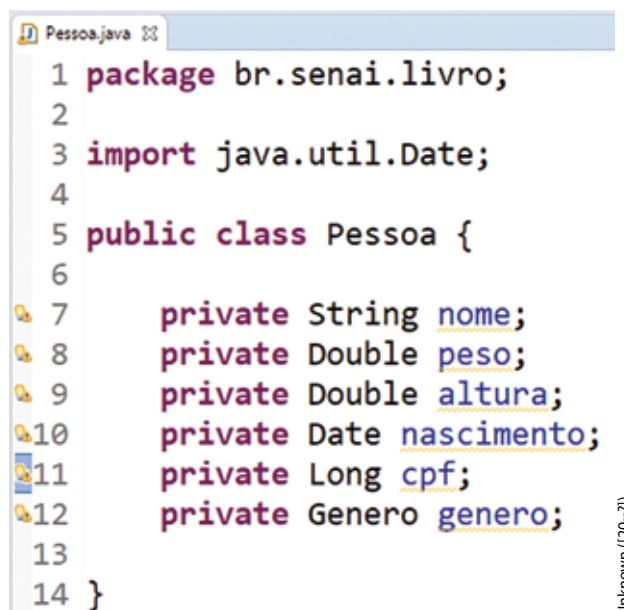
Figura 3 - Tela da IDE Eclipse – Primeira proposta objeto pessoa
Fonte: do Autor (2020)

Ao analisar esse primeiro ensaio de programação, notou-se que talvez houve engano em algumas questões de padronização.

LINHA	PROBLEMA DE LEGIBILIDADE – IDE ECLIPSE
5	Foi criado aqui um atributo que definirá o “nome” de uma determinada “pessoa”. Não faz muito sentido ser redundante falando que é “nomePessoa”, porque na verdade esse atributo pertence a uma classe chamada “Pessoa”. Se eu mantiver assim a referência seria algo como: “Pessoa.nomePessoa”, e isso é redundante.
6 e 7	Passa pelo mesmo problema do atributo anterior: peso e altura são atributos de uma determinada pessoa, não é preciso ser redundantes repetindo isso no nome da variável.
8	Neste caso, propõe-se que nascimento é um tipo de dado caractere, porém no Java há um outro tipo de dado, mais adequado para guardar o nascimento de uma pessoa, chamado de <i>Date</i> .
9	Nesta linha, foi criado um atributo tipo caractere para guardar o CPF de uma pessoa, mas esse tipo de dado poderia ser transformado em um tipo numérico, no caso do Java, poderia ser um <i>Long</i> .
10	O gênero, por se tratar de algo constante, poderia criar um <i>enum</i> para definir melhor os tipos possíveis, não deixando tão aberto para alguém digitar algo errado neste atributo.

Tabela 2 - Interpretação das linhas de código em Java da Primeira proposta objeto pessoa
Fonte: do Autor (2020)

Observando então essas análises de legibilidade, é preciso reformular a proposta de classe pessoa, deixando algumas questões mais bem resolvidas do ponto de vista de padrões.



```

1 package br.senai.livro;
2
3 import java.util.Date;
4
5 public class Pessoa {
6
7     private String nome;
8     private Double peso;
9     private Double altura;
10    private Date nascimento;
11    private Long cpf;
12    private Genero genero;
13
14 }
  
```

Figura 4 - Tela da IDE Eclipse – Segunda proposta objeto pessoa
Fonte: do Autor (2020)

Nessa segunda proposta, é possível notar que o código fica muito mais claro do ponto de vista de não possuir questões redundantes ou com tipos de dados não aderentes.

**SAIBA
MAIS**

Procure sempre se manter atualizado em relação às novidades das linguagens de programação, pois novas formas de declaração e sintaxe de estruturas de programação acabam sendo afetadas, que impactam, obviamente, na necessidade de novos padrões a (e muitas vezes devem) serem adotados pelas empresas e profissionais de desenvolvimento de sistemas.

Outra questão pertinente é a leitura referente a padrões de projetos em desenvolvimento de sistemas. Você irá encontrar muita literatura pertinente ao tema de legibilidade de código, código limpo (clean code) e à adoção de técnicas internacionalizadas para programação. Lembre-se de que trabalhar com TI é atuar em um mercado sempre conectado e global.

PALAVRA DO DOCENTE

Procure sempre se manter atualizado em relação às novidades das linguagens de programação, pois novas formas de declaração e sintaxe de estruturas de programação acabam sendo afetadas, que impactam, obviamente, na necessidade de novos padrões a (e muitas vezes devem) serem adotados pelas empresas e profissionais de desenvolvimento de sistemas.

REFERÊNCIAS

ANSELMO, Fernando. **Aplicando lógica orientada a objetos em Java**: da lógica à certificação. 3. ed. Florianópolis: Visual Books, 2013.

ARAÚJO, Everton Coimbra de. **Algoritmos**: fundamentos e pratica. Florianópolis: Visual Books, 2005.

FREEMAN, Eric; FREEMAN, Elisabeth. **Use a cabeça**: padrões de projetos. 2. ed. Rio de Janeiro (RJ): Alta Books, 2009.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 24. ed. São Paulo (SP): Érica, 2011.

PREISS, Bruno R. **Estruturas de dados e algoritmos**: padrões de projetos orientados a objetos com Java. Rio de Janeiro (RJ): Campus, 2001.

SOUZA, Marco Antonio de; GOMES, Marcelo Marques; SOARES, Márcio José; CONCILIO, Ricardo. **Algoritmos e lógica de programação**. São Paulo (SP): Thomson Pioneira, 2005.



Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO