

A series of overlapping geometric shapes, primarily triangles and quadrilaterals, in shades of teal and purple, located in the top-left corner of the slide.

AceLeraDev Java

Módulo 8 - Clean Code

A series of overlapping geometric shapes, primarily triangles and quadrilaterals, in shades of teal and purple, located in the bottom-right corner of the slide.

The slide features decorative geometric lines in purple and teal. In the top-left corner, there are overlapping lines forming a hexagonal shape. In the bottom-right corner, there are more overlapping lines, also forming a hexagonal shape. The main content is centered on the left side of the slide.

Tópicos da Aula

- SOLID;
- CLEAN CODE;
- OBJECT CALISTHENICS;
- TDD.

Fama do Clean Code

- O conceito de Clean Code veio com o livro do Programador Robert C. Martin (Uncle Bob).
- Seu livro é chamado de Clean Code: A Handbook of Agile Software Craftsmanship

The image features a light gray background with decorative geometric lines in teal and purple. These lines form various shapes, including triangles and polygons, primarily located in the top-left and bottom-right corners. The text is centered in the middle of the image.

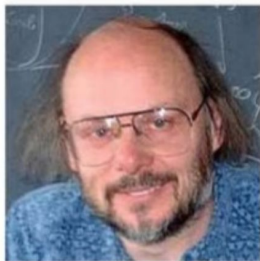
O que é Clean Code (Código Limpo) para voce?

Clean Code é um código mais:

- Eficiente
- Simples
- Direto ao ponto
- Mínimas dependencias
- Sem duplicacao
- Fácil manutencao
- Padroes definidos
- Fácil leitura e entendimento
- Coberto de Testes
- Elegante

O que é Clean Code para alguns programadores famosos:

Elegante e eficiente.
Código limpo faz
bem uma coisa.



Bjarne Stroustrup
Inventor do C++

Simple e direto.
Pode ser lido como
uma conversa.



Grady Booch
Autor de importantes livros sobre
OO e co-criador da UML

Parece ter sido
escrito por alguém
que se importa.



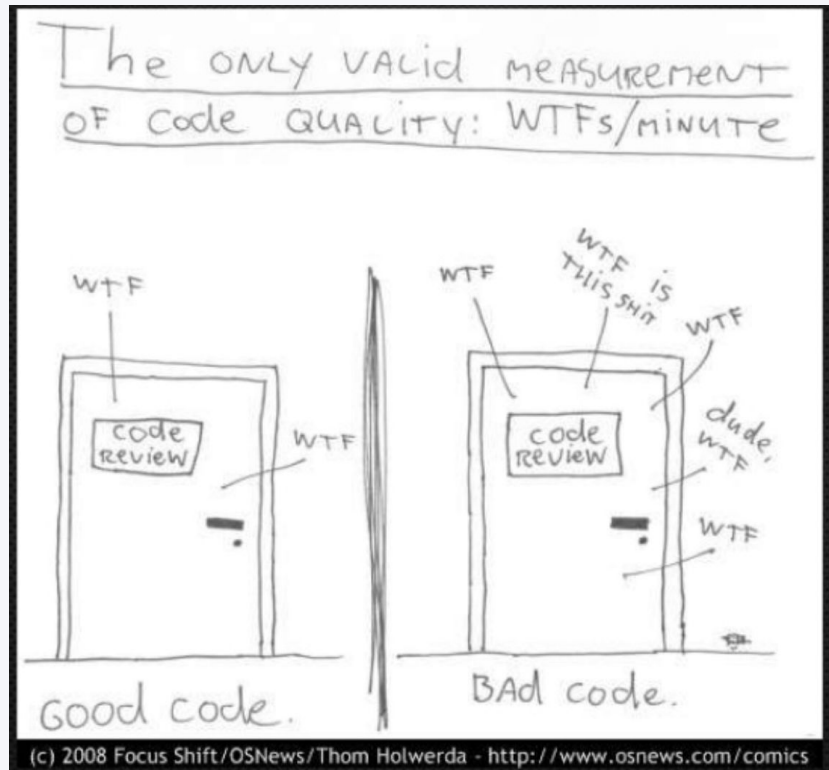
Michael Feathers
Escritor e agile coacher

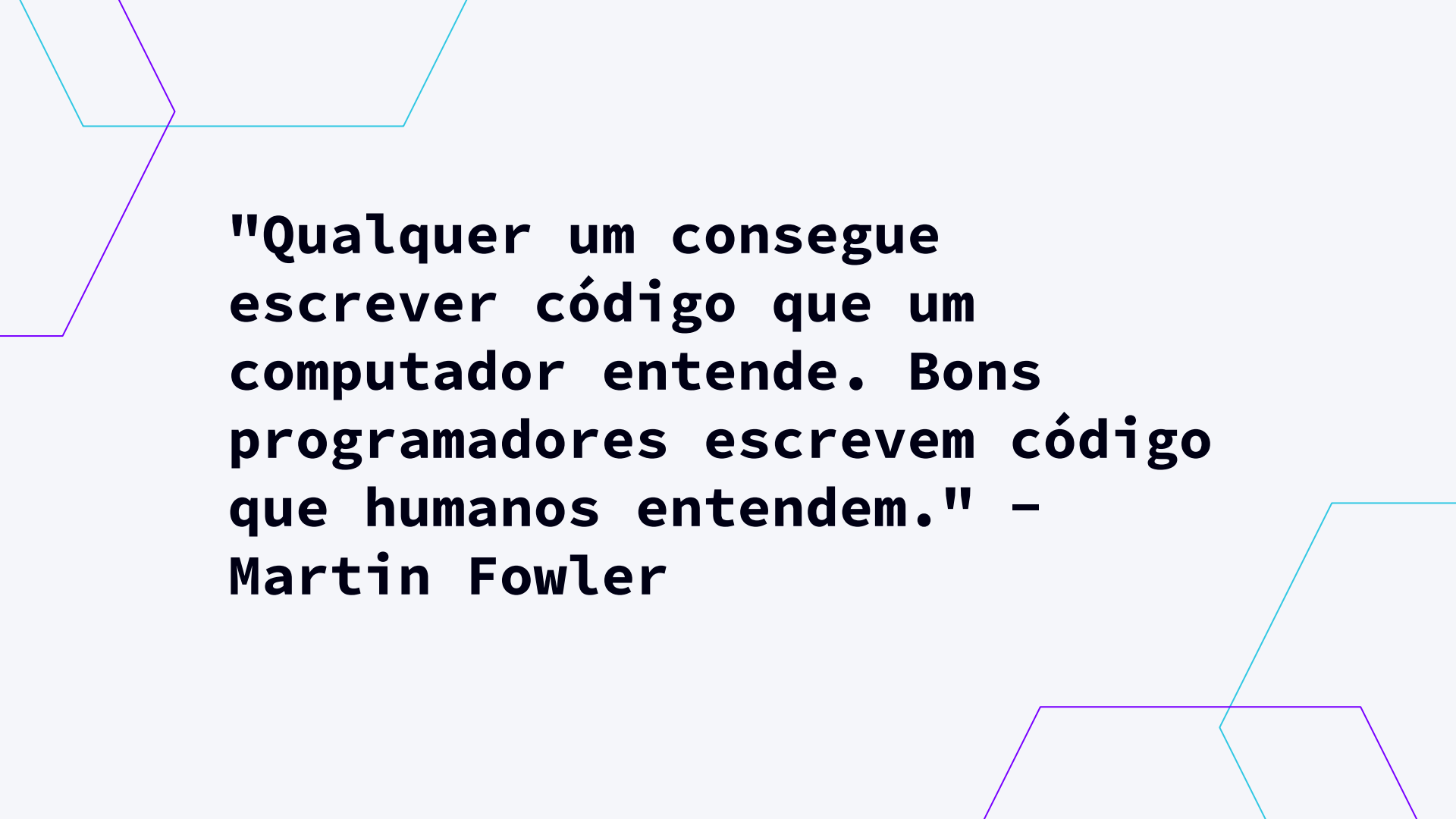
Cada rotina que
você lê faz o que
você espera.



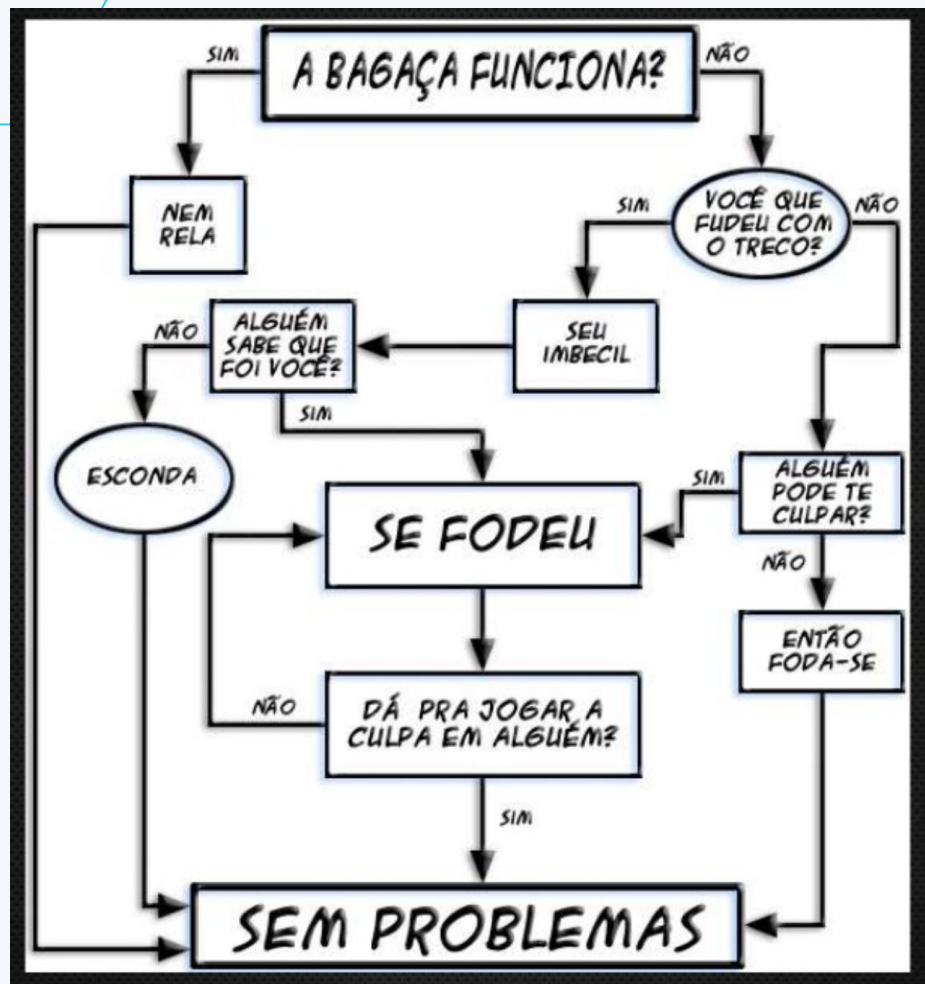
Ward Cunningham
Criador da Wiki, co-criador da
eXtreme Programming, etc

Qual porta representa seu código?



The image features a light blue background with decorative geometric lines in purple and teal. These lines form various shapes, including triangles and polygons, primarily located in the top-left and bottom-right corners, framing the central text.

**"Qualquer um consegue
escrever código que um
computador entende. Bons
programadores escrevem código
que humanos entendem." –
Martin Fowler**



Quais são suas desculpas para não criar um código limpo?

- Cronograma é apertado?
- Sem tempo para frescura?
- O chefe está te pressionando?
- Quer mostrar mais produtividade?

Filho feio não tem pai!



Como podemos mensurar a qualidade de um código?

- Linhas de código?
- Complexidade ciclomática?
- Número de métodos?
- Número de estrutura de decisão?
- Número de classes?
- Linhas de código por método?

Primeira regra do Clean Code:

- NOMES SIGNIFICATIVOS!
 - Nomes que revelem a intenção.

```
int d; // tempo transcorrido em dias

int tempoTranscorridoEmDias;
int diasDesdeCriacaoDoArquivo;
int diasDesdeModificacaoDoArquivo;
int idadeDoArquivoEmDias;
```

Nomes Significativos

```
public List<int> obter()  
{  
    int[] x = new int[3];  
    List<int> lista1 = new List<int>();  
    for (int i = 0; i < lista; i++)  
    {  
        if (x[0] == 4)  
        {  
            lista1.Add(x[0]);  
        }  
    }  
    return lista1;  
}
```

```
public List<int> obterDiasMarcados()  
{  
    int[] diaMarcado = new int[3];  
    List<int> diasMarcados = new List<int>();  
    for (int dia = 0; dia < mes; dia++)  
    {  
        if (diaMarcado[STATUS] == MARCADO)  
        {  
            diasMarcados.Add(diaMarcado[STATUS]);  
        }  
    }  
    return diasMarcados;  
}
```

**Mais importantes que nomes
significativos são nomes
pronunciáveis.**

```
class DtaRcrd102
{
    private DateTime gerdmahms;
    private DateTime moddmahms;
    private string pszqint = "102";
}

class Cliente
{
    private DateTime gerarDataHora;
    private DateTime modificarDataHora;
    private string idRegistro = "102";
}
```

Use nomes buscáveis:

```
for (int j = 0; j < 30; j++)
{
    s = (t[j]*4)/5;
}

const int DIAS_DE_TRABALHO_POR_SEMANA = 5;
int soma = 0;
int diasReaisDeTrabalho = 4;

for (int j = 0; j < NUMERO_DE_TAREFAS; j++)
{
    int tarefasPorDia = trabalhoEstimado[j] * diasReaisDetrabalho;
    int taredasPorSemana = (dias / DIAS_DE_TRABALHO_POR_SEMANA);

    soma += taredasPorSemana;
}
```


Nomeando classes e métodos

- Classes
 - Representadas por substantivos.
 - Ex: Cliente, Perfil, Estoque, etc.
- Métodos
 - Representadas por verbos ou frases verbais
 - Ex: enviarPagamento, salvar, etc.

Desenvolvendo funções:

- 0 mais pequenas possível.
- Menos é mais!
- Extraia trecho em métodos privados.
- Lembre-se dos nomes significativos.
- Direto ao ponto.
- Muitos níveis de indentação = muitas responsabilidades
- Está fazendo mais de uma coisa? Extraia.
- Apenas uma coisa e bem feita.

```
public static String renderPageWithSetupsAndTearardowns(  
    PageData pageData, boolean isSuite) throws Exception {  
  
    if (isTestPage(pageData))  
        → includeSetupAndTearardownPages(pageData, isSuite);  
    return pageData.getHtml();  
}
```

Leitura do Código

- Seu código deve ser lido como uma narrativa;
- Temos sujeitos, verbos e predicados;
- Narrativas são frases em ordem coerente;
- Lembre-se disto ao extrair em métodos privados.
- Muitos argumentos = Code Smell.
- Tente utilizar sempre o máximo 3 argumentos por no seu método.

```
public void render(boolean withXPT0) {  
    doSomething();  
    if (withXPT0) {  
        doSomethingElse();  
    }  
}
```

```
public void renderWithXPT0() {  
    doSomething();  
    doSomethingElse();  
}
```

```
public void renderWithoutXPT0 {  
    doSomething();  
}
```

DRY (Don't Repeat Yourself)

- Comentários não ajudam um código sujo.
- Um código limpo não precisa de comentários, mas porque?

Comentários

- Em geral, comentários servem para explicar um código ruim.
- Um bom código é auto documentado.
- Se ficar difícil de entender, extraia para um método com o nome que faça o que diz, assim a leitura irá fluir melhor.

```
2 // Check to see if the employee is eligible for full benefits
3 if ((employee.flags & HOURLY_FLAG) && (employee.age > 65)) {
4     // ...
5 }
6
7
8 if (employee.isEligibleForFullBenefits()) {
9     // ...
10 }
```



Tratamentos de Erro:

```
2  if (deletePage(page) == E_OK) {
3      if (registry.deleteReference(page.name) == E_OK) {
4          if (configKeys.deleteKey(page.name.makeKey()) == E_OK) {
5              logger.log("page deleted");
6          } else {
7              logger.log("configKey not deleted");
8          }
9      } else {
10         logger.log("deleteReference from registry failed");
11     }
12 } else {
13     logger.log("delete failed"); return E_ERROR;
14 }

15
16 try {
17     deletePage(page);
18     registry.deleteReference(page.name);
19     configKeys.deleteKey(page.name.makeKey());
20 } catch (Exception e) {
21     logger.log(e.getMessage());
22 }
```

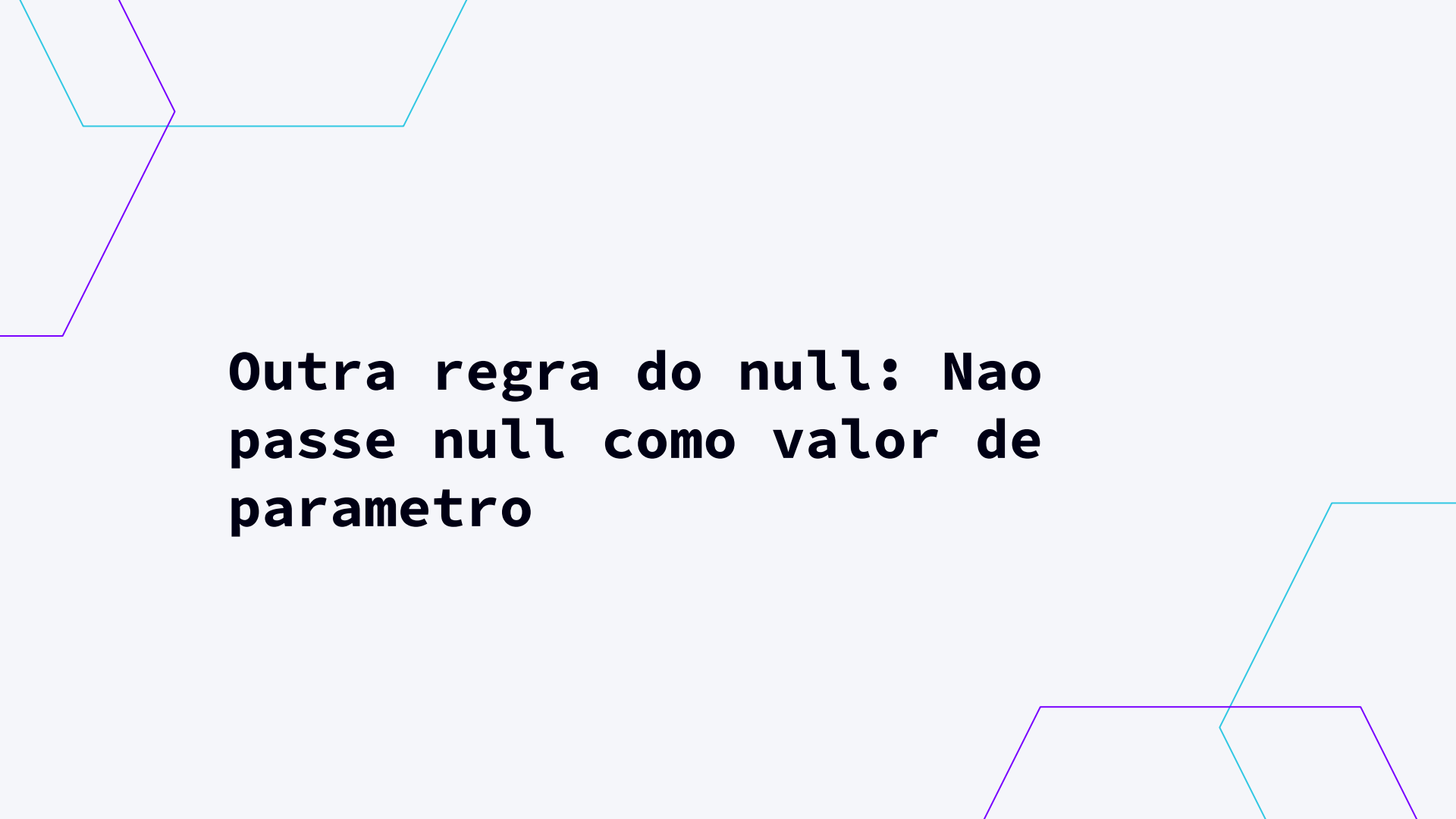


Retornos

- Evitar retornar null.
- Ao retornar null existe a possibilidade de se retornar um NPE.
- Opte por lançar uma exceção ou retornar um objeto especial.

```
2 List<Employee> employees = getEmployees();
3 if (employees != null) {
4     for(Employee e : employees) {
5         totalPay += e.getPay();
6     }
7 }
8
9 List<Employee> employees = getEmployees();
10 for(Employee e : employees) {
11     totalPay += e.getPay();
12 }
```



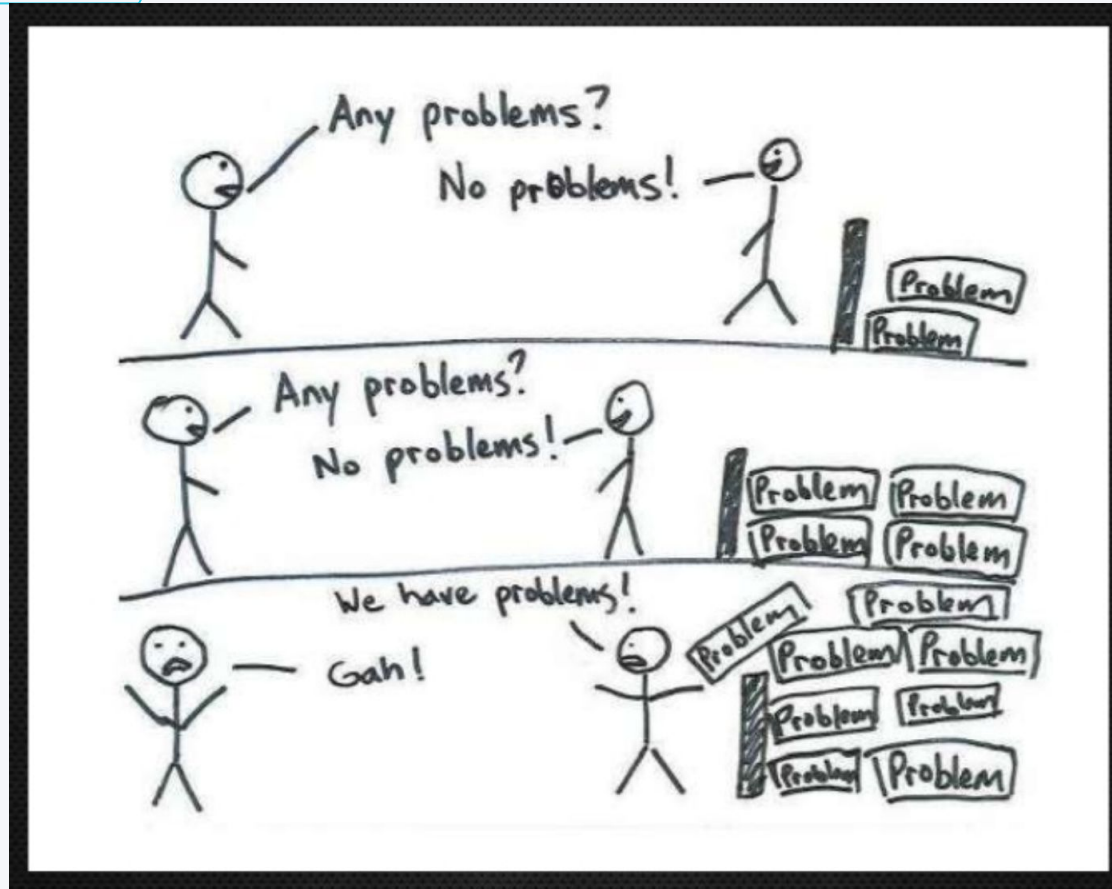
The image features a light gray background with decorative geometric lines in teal and purple. These lines form various angles and shapes, primarily concentrated in the top-left and bottom-right corners, creating a modern, abstract aesthetic.

**Outra regra do null: Nao
passe null como valor de
parametro**

The image features a light gray background with decorative geometric lines in purple and teal. These lines form various shapes, including triangles and polygons, primarily located in the top-left and bottom-right corners. The text is centered in the middle of the image.

**Regra dos escoteiros: "Deixe
a área do acampamento mais
limpa do que você encontrou"**

Não deixe acumular problemas!



Feedback da aula

