Google Java Style Guide

Estrutura Geral de Arquivo

```java
// Pacote primeiro
package com.exemplo.minhapasta;

// Imports depois (primeiro os normais, depois os estáticos)
import java.util.List;
import java.util.Map;

import static java.lang.Math.PI;

/**
 * Descrição da classe (Javadoc obrigatório para classes públicas).
 */
public class MinhaClasse {

    // 1. Constantes
    private static final int COLUNA_MAXIMA = 100;

    // 2. Variáveis static
    private static int contador;

    // 3. Variáveis de instância
    private String nome;
    private int idade;

    // 4. Construtores
    public MinhaClasse(String nome) {
        this.nome = nome;
  }

    // 5. Métodos públicos
    public String getNome() {
        return nome;
  }

    // 6. Métodos privados
    private void calcularIdade() {
        // ...
```

```
  }

    // 7. Classes internas (se houver)
    private static class Helper {
        // ...
  }
}
```

Regras de Recuo, Espaçamento e Quebra de Linhas

Regra

Indentação -> 4 espaços

Máximo por linha -> 100 colunas

Quebra de linha -> Sempre após vírgula ou operador, nunca antes.

Chaves -> Sempre usar chaves, mesmo em blocos de uma linha

Exemplo de quebra correta de método com muitos parâmetros:

```
public void meuMetodo(
    int parametro1,
    int parametro2,
    String outroParametro
) {
    // corpo
}
```

Conveções de Nomeação

Elemento | Exemplo

Pacotes -> com.meuprojeto.modulo (tudo minúsculo)

Classes -> Cliente, AtendimentoService (PascalCase)

Interfaces -> Validador, Repositorio

Métodos -> buscarPorId(), validarEntrada() (camelCase)

#Variáveis -> quantidade, dataCriacao (camelCase)

Constantes -> VALOR_MAXIMO, TAXA_DESCONTO (UPPER_CASE_SNAKE_CASE)

Uso de this

- Só usar **quando houver ambiguidade** (ex: entre parâmetro e atributo).

```
public void setNome(String nome) {
    this.nome = nome;
}
```

Importações

- **Proibido usar wildcard (*).**
- **Ordem:**
    1. Java SE
    2. Bibliotecas externas
    3. Código da aplicacão
    4. Imports estáticos (sempre por último)

```
import java.util.List;
import java.util.Map;

import org.joda.time.DateTime;

import static java.lang.Math.PI;
```

Formatação de Condicionais, Loops, etc.

Sempre com chaves, mesmo para uma única linha:

```
if (condicao) {
  executar();
} else {
  outroExecutar();
}
```

Boas Práticas Extras

- Não deixe métodos com mais de **40-50 linhas**, sempre que possível.
- Cada classe deve ter **uma única responsabilidade clara.**
- Evite **métodos estáticos em excesso** (use serviços, ou classes com estado, quando faz sentido).
- Cada arquivo *.java* só pode ter **uma classe pública**, com o mesmo nome do arquivo.
- Comentários de linha (//) **somente para explicações realmente necessárias.**

Exemplo de um construtor seguindo o padrão:

```
public AtendimentoService(
        AtendimentoConsoleUI ui,
```

```
        AnimalRepository animalRepository,
        AtendimentoRepository atendimentoRepository,
        ClienteRepository clienteRepository,
        FuncionarioRepository funcionarioRepository
) {
    this.ui = ui;
    this.animalRepository = animalRepository;
    this.atendimentoRepository = atendimentoRepository;
    this.clienteRepository = clienteRepository;
    this.funcionarioRepository = funcionarioRepository;
}
```