Universidade Federal de Santa Maria
Laboratório de Orientação a Objetos
Enzo Zanini Becker
Vitória Luiza Camara

Trabalho 2 - Violação de boas práticas

1. **Nomes não informativos:** O nome subjetivo de certas variáveis, classes e objetos acaba não deixando claro a função das mesmas, podendo gerar confusão no código.
    a. https://github.com/zaidmukaddam/Uni_Java_Codes/blob/7ffe0ea7f9894d426d21883c715f4864bc53eeb5/Z.java#L13

2. **Classes gigantes:** A classe `Macro` possui muitos métodos e atributos, o que pode torná-la difícil de entender e dar manutenção.
    a. https://github.com/plantuml/plantuml/blob/91e87b9691630d27249f186ece2ffcae823a2bf8/src/smetana/core/Macro.java#L1220

3. **Métodos longos:** O método `build` da classe `URLBuilder` tem um tamanho considerável, o que pode dificultar sua compreensão e manutenção.
    a. https://github.com/apache/dubbo/blob/155c2f2c186cc797a2f31d9488e66cca07f892c4/dubbo-common/src/main/java/org/apache/dubbo/common/URLBuilder.java#L355

4. **Acoplamento forte:** A classe `Dependency` está fortemente acoplada a detalhes de implementação específicos, o que pode dificultar a reutilização e a manutenção.
    a. https://github.com/JetBrains/intellij-community/blob/208c3e8d2b69b72d690c33aab0ebd69252267850/plugins/devkit/devkit-core/src/dom/Dependency.java#L19

5. **Herança desnecessária:** A classe `CakeDao` está herdando de uma classe sem adicionar funcionalidades relevantes, o que pode ser considerado uma herança desnecessária.
    a. https://github.com/iluwatar/java-design-patterns/blob/1c801da4a7cb7dabe157b43d6eb7d40a709d4704/layers/src/main/java/dao/CakeDao.java#L32

6. **Uso excessivo de tipos genéricos:** O método `createExecutorService` da classe `ThreadPool` usa tipos genéricos de forma extensiva, o que pode tornar o código mais complexo e difícil de entender.
    a. https://github.com/elastic/elasticsearch/blob/7706bedfe816c2415f0da04555d27ab306fdd33d/server/src/main/java/org/elasticsearch/threadpool/ThreadPool.java#L539

7. **Expressões lambda complexas:** A expressão lambda na classe `Lambda` parece ser complexa, o que pode tornar o código difícil de entender e manter.

    a. https://github.com/Z3Prover/z3/blob/2682c2ef2b3f31f065cc54b83e91f6d42c60db2f/src/api/java/Lambda.java#L26

8. **Ignorar anotações de supressão e avisos:** O código na classe `Output` parece ignorar algumas anotações de supressão e avisos, o que pode indicar problemas de qualidade de código.
    a. https://github.com/tensorflow/tensorflow/blob/95adf8a2f3c100dedd20d073c66edf758b713ff9/tensorflow/java/src/main/java/org/tensorflow/Output.java#L62

9. **Capturar exceções genéricas:** O código na classe `JsrTest` captura uma exceção genérica, o que pode dificultar a identificação e o tratamento adequado de problemas específicos.
    a. https://github.com/NationalSecurityAgency/ghidra/blob/20f5bd9bece9050fdf64019443cf2ccfc6c29029/Ghidra/Processors/JVM/resources/JsrTest.java#L19

10. **Não fechar recursos adequadamente:** O código na classe `GraphOperation` parece não fechar recursos adequadamente, o que pode levar a vazamentos de recursos e problemas de desempenho.
    a. https://github.com/tensorflow/tensorflow/blob/95adf8a2f3c100dedd20d073c66edf758b713ff9/tensorflow/java/src/main/java/org/tensorflow/GraphOperation.java#L43