



Lucas-Tito

fix the longmethod code smell using the extract method refactoring technique



GitHub Copilot

Vou ajudar você a refatorar o código para corrigir o code smell "Long Method" no método `getReferenceCountMap()` usando a técnica Extract Method.

Passo a passo:

1. Dividir o método `getReferenceCountMap()` em métodos menores e mais específicos
2. Criar métodos separados para contar cada tipo de referência
3. Manter o método original mais limpo e legível

`StudyMaterial.java`

O arquivo precisa ser modificado para extrair a lógica de contagem de cada tipo de referência em métodos separados.



`StudyMaterial.java`

Principais mudanças:

1. Extraí a inicialização do mapa para o método `initializeCountMap()`
2. Criei um método `countAllReferences()` para iterar sobre as referências
3. Extraí a lógica de contagem específica para cada tipo em métodos separados
4. Substituí o uso de `getClass()` por `instanceof` que é mais seguro e legível
5. Os novos métodos são privados pois são detalhes de implementação internos



Lucas-Tito

you added smells of feature envy in the methods `countVideoReference` and `countTextReference`



GitHub Copilot

Você está certo! Vamos corrigir o Feature Envy movendo a lógica de contagem para suas respectivas classes, seguindo o princípio Tell Don't Ask.

Passos para corrigir:

1. Mover a lógica de contagem para as classes `VideoReference` e `TextReference`
2. Simplificar os métodos de contagem em `StudyMaterial`

`TextReference.java`



`TextReference.java`

`VideoReference.java`



`VideoReference.java`

StudyMaterial.java

StudyMaterial.java

Principais mudanças:

1. Adicionei o método `shouldBeCountedAsReference()` nas classes `TextReference` e `VideoReference`
2. Agora as classes são responsáveis por determinar se devem ser contadas
3. `StudyMaterial` não precisa mais conhecer a lógica interna de cada tipo de referência
4. O código fica mais orientado a objetos e respeita melhor o encapsulamento



Lucas-Tito

os smells persistem



GitHub Copilot

Você está certo. Ainda temos code smells porque a lógica de contagem ainda está espalhada. Vamos aplicar o padrão Template Method para resolver isso de forma mais elegante.

Passo a passo:

1. Mover a lógica de contagem para a classe base `Reference`
2. Criar um método abstrato na classe base para validação específica
3. Simplificar a contagem em `StudyMaterial`

Reference.java

Reference.java

VideoReference.java

VideoReference.java

TextReference.java

TextReference.java

StudyMaterial.java

StudyMaterial.java

Principais mudanças:

1. Adicionei método abstrato `isValidForCounting()` na classe `Reference`
2. Cada tipo de referência implementa sua própria validação
3. Adicionei método `getReferenceTypeName()` para padronizar os nomes das chaves
4. Simplifiquei a lógica de contagem em um único método
5. Removi os métodos específicos de contagem que estavam duplicando código

Esta solução:

- Remove a duplicação de código
- Centraliza a lógica de contagem
- Torna mais fácil adicionar novos tipos de referência
- Segue o princípio Open/Closed
- Resolve o Feature Envy



Lucas-Tito

o seguinte erro está sendo retornado:


java: org.example.studymaterial.AudioReference is not abstract and does not override abstract method isValidForCounting() in org.example.studymaterial.Reference



GitHub Copilot

Ah sim! Precisamos implementar o método abstrato `isValidForCounting()` na classe `AudioReference` também, já que ela herda de `Reference`.

`AudioReference.java`

 `AudioReference.java`

Explicação:

- Para `AudioReference`, como não há validações específicas mencionadas, vamos assumir que todos os áudios são válidos para contagem
- Se no futuro houver necessidade de validações específicas para áudio, podemos modificar esta implementação

