



Lucas-Tito

fix the longmethod code smell using the extract method refactoring technique



GitHub Copilot

Vou ajudar você a refatorar o método `toString()` que está muito longo, dividindo-o em métodos menores e mais específicos. `TodoTracker.java`

O método `toString()` será dividido em métodos auxiliares para melhorar a legibilidade e manutenibilidade do código.

`TodoTracker.java`

Mudanças realizadas:

1. O método `toString()` foi simplificado e agora delega responsabilidades para métodos menores
2. Criado método `appendToDoInfo()` para adicionar informações básicas do `ToDo`
3. Criado método `appendTrackingInfo()` para adicionar informações de rastreamento
4. Criado método `formatDateTime()` para formatar as datas
5. Adicionada verificação de lista vazia no início do método `toString()`

Cada método agora tem uma única responsabilidade, tornando o código mais fácil de manter e entender.



Lucas-Tito

you added a featureEnvy smell in the `appendToDoInfo` method



GitHub Copilot

O Feature Envy existe quando um método está mais interessado nos dados de outra classe do que na sua própria classe. Vou corrigir isso movendo a lógica de formatação para a classe `ToDo`, onde ela pertence. `TodoTracker.java`

Vamos simplificar o método `appendToDoInfo` e mover a responsabilidade de formatação para a classe `ToDo`:

`TodoTracker.java`

`ToDo.java`

Precisamos adicionar um novo método na classe `ToDo`:

`ToDo.java`

Esta mudança:

1. Remove o Feature Envy movendo a responsabilidade de formatação para a classe `ToDo`
2. Mantém o encapsulamento adequado
3. Torna o código mais coeso, com cada classe gerenciando seus próprios dados





Lucas-Tito

o smell de featureEnvy persiste no método `appendToDoInfo`



GitHub Copilot

O Feature Envy ainda persiste porque o método `appendToDoInfo` ainda está mais interessado na formatação do `ToDo` do que nas responsabilidades do `TodoTracker`. Vamos resolver isso de forma diferente:

Passos para correção:

1. Remover o método `appendToDoInfo` completamente
2. Incorporar a lógica diretamente no `toString()` do `TodoTracker`
3. Usar o `toString()` do `ToDo` diretamente

`TodoTracker.java`

Remover o método `appendToDoInfo` e simplificar o `toString()`:

`TodoTracker.java`

Agora:

1. Removemos a camada extra de indireção desnecessária
2. O código está mais direto e coeso
3. Cada classe mantém suas próprias responsabilidades
4. Não há mais necessidade do método `toFormattedString()` na classe `ToDo`

Você pode também remover o método `toFormattedString()` da classe `ToDo`, pois não é mais necessário.

