Aluno: Moises R. Souza.
Curso: Engenharia da Computação
Bloco: Refatoração

Refatoração:

uma mudança feita na estrutura interna do software para deixa-lo mais fácil de entender e barato de modificar sem mudar seu comportamento.
observavel.

Algumas tecnicas de refatoração

Reorganizando as coisas.
Aplicando a técnica 'Extrair Método'
Mover método.
Mover atributo (ou campo)
Extrair Classe.
Encapsular campo.
Renomear Método.

Refatoração : Extrair Método (ExtractMethod)

Quando você têm partes similares do código que podem ser agrupadas, geralmente código que está duplicado, então você deve mover essas partes para um novo método que tenha um nome que faça sentido.

A principal motivação para esta refatoração é eliminar do código métodos muito grandes que realizam mais de uma tarefa e geralmente precisam estar cheios de comentários para que seja compreendido.

Métodos menores aumentam a granularidade do código e quando bem nomeados aumentam a clareza do que foi escrito. A nomeação de métodos é vital para que esta técnica funcione, o nome deve dizer claramente o que o método se propõe a fazer, e o tamanho do nome do método não deve ser um entrave, utilize nomes tão grandes quanto necessário (Ex.: ObterUrlDeRedirecionamentoParaSiteParceiro(codigoParceiro)).

Refatoração : Mover Método (Move Method)

Quando você possui um método que será utilizado muitas vezes por outra classe diferente da classe na qual ele foi definido, crie um método similar nesta classe que o está utilizando, depois copie o corpo do método copiado para o novo método, então substitua as chamadas para utilizarem o método local. Pode-se também avaliar se o método original deve ser preservado ou se pode ser eliminado.

Refatoração : Extrair Classe (Extract Class)

Quando você possui uma classe que está executando a tarefa de duas, então crie uma nova classe, depois mova todos os atributos relevantes para esta nova classe.

A motivação desta técnica é bastante óbvia, sempre ouvimos falar que cada classe deve ser coesa e ter objetivos bastante claros, mas na prática as classes crescem. Quando um desenvolvedor adiciona uma nova funcionalidade em certa classe, talvez ela não parecesse tão errada, mas com o tempo esta funcionalidade irá crescer, e trazer mais responsabilidades para a classe.

Ao extrair estas responsabilidades da classe e a colocando em outra, há um ganho em coesão, legibilidade e reutilização do código.
.

Refatoração : Encapsular Atributo (Encapsulate Field)

Quando um atributo é acessado diretamente e isto parecer ser estranho, devem-se criar métodos de acesso a este atributo.

Existem duas escolas que defendem meios diferentes de acessar atributos de uma classe. A primeira diz que os atributos expostos devem ser acessados diretamente, pois isto torna o código mais legível. A segunda diz que devemos criar métodos de acesso a cada atributo exposto da classe, pois isto facilita que subclasses sobrescrevam os métodos de acesso.

O melhor a fazer neste caso é discutir com outros membros da equipe para determinar qual a melhor maneira de dar acesso aos atributos, de modo a manter coerência com o resto do software.

Uma boa maneira de resolver este problema é criar os atributos sendo acessados diretamente e quando isto começar a parecer estranho, substituir por métodos de acesso.

Quando estes atributos que são acessados estão em uma superclasse, primeiro deve-se encapsular o atributo internamente, na superclasse, e em seguida, sobrescrever o método de acesso na classe derivada.

Refatoração : Renomear Método (Rename Method)

Quando o nome de um método não revela qual a sua intenção, deve-se altera-lo imediatamente, de modo que o novo nome deve descrever claramente o que ele se propõe a fazer.

Coesão de classes

Baixo Acoplamento

Alta Coesão

(entre classes)

Muitas classes e pequenas

Menos dependência por classe

Menor conhecimento

Coesão de uma Classe:

Evite Surpresas

Imutabilidade é uma defesa contra

efeitos colaterais

Vantagens

- Mais seguro e confiável

- Mais legível:

minimiza testes e garantias

diminui o código por tabela

- Paralelismo e escalabilidade

Função Pura

1. Para um certo Input sempre retorna o mesmo Output

2. Não causa nenhum efeito colateral

3. Não depende de nenhum dado externo (autocontida)

Vantagens

- Mais simples e menores

- Mais fácil de entender e testar

Quando usar em código OO?

Sempre que possível

Em muitas situações não é possível: BD, integração, estados etc

Possível em regras de negócio, métodos utilitários, internos etc

Refatoração Tempestiva

“Refatoração (refactoring) é a disciplina técnica para reestruturar um determinado código, alterando sua estrutura interna sem mudar seu comportamento externo”

Martin Fowler

Problema Inicial Solução Inicial

Problema Refinado

Refatoração

Mudança

Solução Final

Validar Solução

Refinamento

Ciclo de

Sprint

Ajuste da solução em função de:

- uma nova visão sobre o problema

- preparar para uma evolução ou mudanç

-