Detecção e Refatoração de Code Smells

João Teixeira do Nascimento

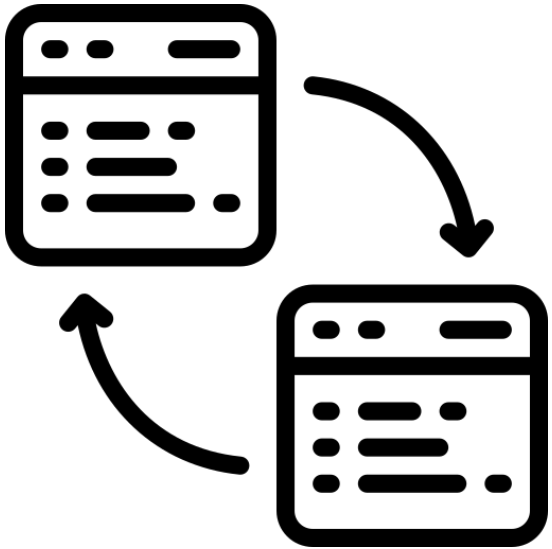
Refatoração

"Mudança feita na estrutura interna do software para deixar mais fácil de entender e mais fácil de modificar" - Martin Fowler.

Os softwares precisam de adaptação com o passar do tempo.

É a etapa de adaptação que ocorre anterior à “atualização”.

Exemplo: Um sistema feito na disciplina de Programação Orientada a Objetos é refatorado para receber um relacionamento muitos para muitos exigido no trabalho da disciplina de Fundamentos de Banco de Dados.



Objetivos da Refatoração

- Remover complexidade
- Melhorar a qualidade do software
- Fazer código ficar mais simples e fácil de entender
- Fazer o código ficar mais flexível
- Fazer o código ficar fácil de modificar

Code Smells

- Indica que existe algo errado no seu código
- O sistema não deixa de funcionar corretamente
- Ex: Afrouxar um parafuso usando uma faca, abrir o porta chip do celular com um brinco

Lista de Code Smells

Fowler cataloga 22 tipos de code smells, são eles:

- **Duplicated Code**
- **Long Method**
- **Large Class**
- **Long Parameter List**
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- **Switch Statements**
- Parallel Inheritance Hierarchies
- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- **Data Class**
- Refused Bequest
- Comments

Lista de Code Smells do experimento

Code Smells detectados pelo PMD:

- Large Class ou God Class
- Data Class
- Duplicated Code

Code Smells detectados pelo Designite:

- Complex Conditional ou Switch Statement
- Long Method
- Long Parameter List
- Magic Number

Data Class

São classes que contém apenas atributos e métodos brutos de acesso(getters e setters), eles ainda não tiveram responsabilidade atribuída e são somente contêineres de dados de outras classes.

Refatoração:

- Move Method
- Extract Method
- Remove Setting Method, Hide Method

God Class(Large Class)

É uma classe muito grande, que pode ter muitas responsabilidades, atributos, métodos e com isso um maior número de linhas.

Exemplo: Uma classe que faz conexão com o banco de dados e autentica usuário

Refatoração:

- Extract Class
- Extract Subclass
- Extract Interface

Duplicated Code

Quando duas partes do código parecem quase idênticas. Pode ser feito sem o conhecimento do autor, quando muitas pessoas trabalham em partes diferentes de um programa chegando a mesma solução que fica duplicada. Também pode ser feito em projetos com prazos curtos para economizar tempo.

Refatoração:

- Extract Method
- Extract Superclass
- Extract Class
- ...

Long Method

Quando um método contém muitas linhas de código, condicionais, variáveis, responsabilidades. É mais fácil adicionar novas atribuições para um método já existente do que criar um novo método.

Refatoração:

- Extract Method
- Replace Method with Method Object
- Decompose Conditional
- ...

Long Parameter List

É quando um método tem uma grande lista de parâmetros, normalmente ocasionado para mesclar diversos tipos de algoritmos. Também pode ser uma forma de diminuir a dependência entre classes.

Refatoração:

- Replace Parameter with Method Call
- Preserve Whole Object
- Introduce Parameter Object

Complex Condicional(Switch Statement)

Quando uma condicional if/else ou switch é usada em um problema que exige uma solução polimórfica. Caso tenha sido feito em diversas partes do sistema, uma mudança exige refatoração de todas as instâncias.

Refatoração

- Extract Method, Move Method
- Replace Type Code with Subclasses
- Replace Type Code with State/Strategy

Magic Number

Números não transmitem significado por si só, isso diminui a legibilidade do código.

Refatoração:

- Replace Magic Number with Symbolic Constant
- Replace with Parameter

Decisões de refatoração

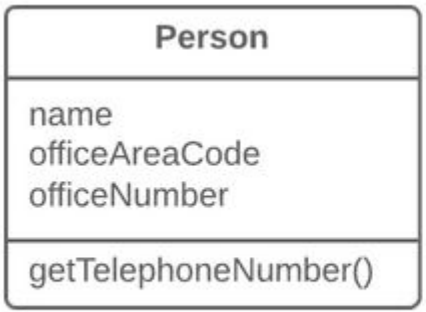
- Não fazer nada
- Descartar por:
  - Ser custoso
  - Detecção errada
- Refatorar

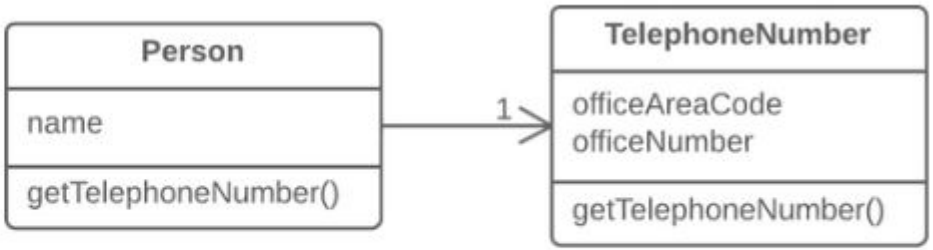
Técnicas de Refatoração

- Extract Class
- Extract Subclass
- Extract Method
- Move Method
- Move Field
- Extract Superclass
- Decompose Conditional
- Replace Magic Number with Symbolic Constant
- Replace Parameter with Method Call
- Introduce Parameter Object
- Replace Conditional with Polymorphism

Extract Class

- Quando uma classe faz o trabalho de duas.
- Solução: Criar uma nova classe e dividir as funções.



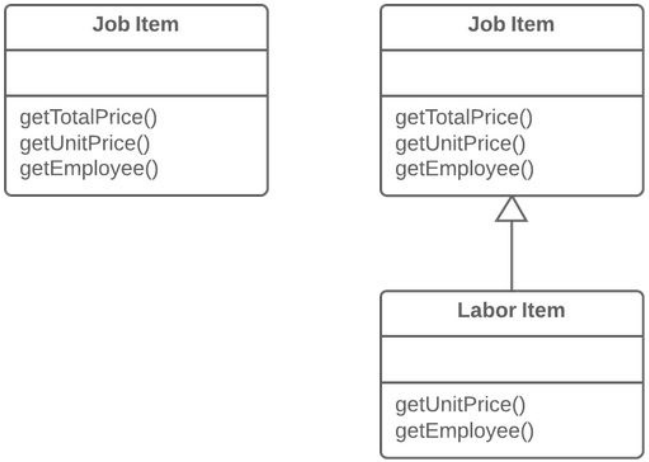


Extract Class

1. Crie uma nova classe para conter funcionalidade relevante.
2. Crie um relacionamento entre a nova classe e a classe antiga.
3. Use move field ou move method para cada atributo e método que você mover para a nova classe.
4. Decida se a nova classe é acessível a partir da classe antiga.

Extract Subclass

- Quando uma classe possui atributos e métodos que podem ser especializados.
- Solução: Criar SubClasses.



Extract Subclass

1. Crie uma nova subclasse da classe de seu interesse.
2. Mova os métodos e atributos necessários da classe pai para a subclasse.
3. Substitua a chamada do construtor pai, com o próprio construtor da subclasse.

Extract Method

- Algumas vezes os métodos têm muita responsabilidade.

```
void printOwing() {
  printBanner();

  // Print details.
  System.out.println("name: " + name);
  System.out.println("amount: " + getOutstanding());
}
```

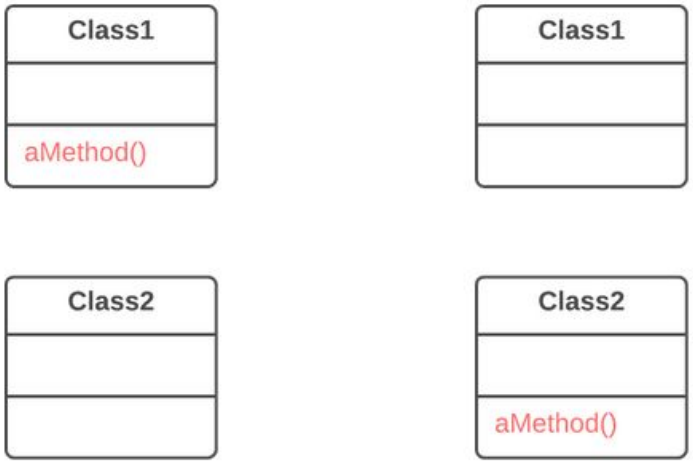
```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}

void printDetails(double outstanding) {
  System.out.println("name: " + name);
  System.out.println("amount: " + outstanding);
}
```

Extract Method

1. Crie um novo método com um nome adequado para a responsabilidade.
2. Copie o código relevante para o novo método, apague o fragmento do método antigo e faça uma chamada para o novo método.
3. No novo método, crie parâmetros para ser passado do antigo método.
4. Retorne os resultados ou outros dados modificados para o antigo método.

Move Method

Um método é mais usado em uma outra classe do que na classe que está inserido.

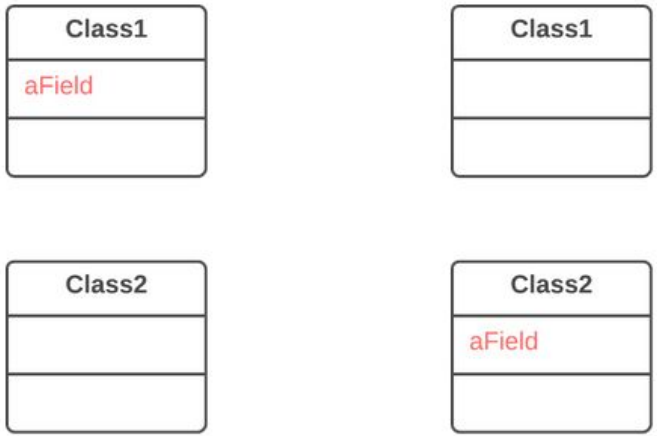


Move Method

1. Cheque todos os elementos usados no método antigo na classe antiga.
2. Se fizer sentido mova-o para uma nova classe

Move Field

Um atributo é usado mais em outra classe do que na classe que está inserido.

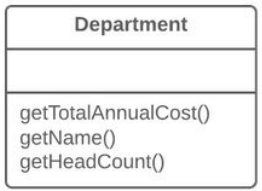


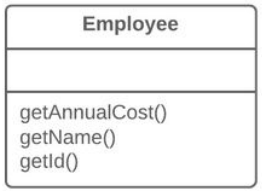
Move Field

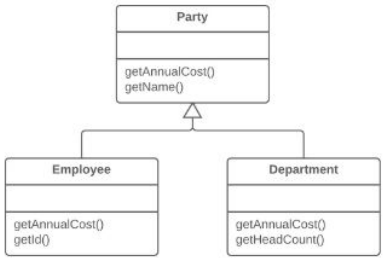
1. Se o atributo é público encapsule-o.
2. Crie uma cópia do atributo na classe destino.
3. Substitua todas as referências da antiga classe por chamadas a esse atributo.
4. Delete o atributo na classe antiga.

Extract Superclass

Quando duas ou mais classes possuem algo em comum.







Extract Superclass

- Crie uma superclasse
- Mova os atributos e métodos em comum para a superclasse
- Procure por trechos no código que possam ser usados na superclasse

Decompose Conditional

Quando o código tem uma condição complexa.

```
if (date.before(SUMMER_START) || date.after(SUMMER_END)) {
  charge = quantity * winterRate + winterServiceCharge;
}
else {
  charge = quantity * summerRate;
}
```

```
if (isSummer(date)) {
  charge = summerCharge(quantity);
}
else {
  charge = winterCharge(quantity);
}
```

Decompose Conditional

1. Extraia a condicional para um novo método com Extract Method.
2. Repita o processo nas outras condições

Replace Magic Number with Symbolic Constant

Quando o código tem um número com significado

```
double potentialEnergy(double mass, double height) {
  return mass * height * 9.81;
}
```

```
static final double GRAVITATIONAL_CONSTANT = 9.81;

double potentialEnergy(double mass, double height) {
  return mass * height * GRAVITATIONAL_CONSTANT;
}
```

Replace Magic Number with Symbolic Constant

1. Declare uma constante e atribua o valor do número mágico para ela
2. Encontre todas as instâncias desse número mágico
3. Para cada instância, verifique se o número mágico corresponde ao propósito da constante.

Replace Parameter With Method Call

Quando o resultado da chamada de um método de consulta é passado como parâmetro de outro método.

```
int basePrice = quantity * itemPrice;
double seasonDiscount = this.getSeasonalDiscount();
double fees = this.getFees();
double finalPrice = discountedPrice(basePrice, seasonDiscount,
```

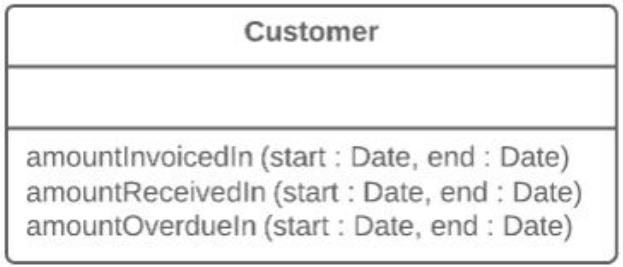
```
int basePrice = quantity * itemPrice;
double finalPrice = discountedPrice(basePrice);
```

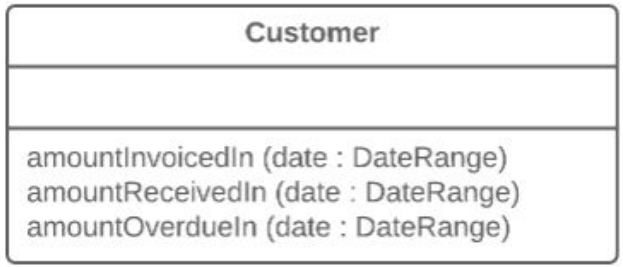
Replace Parameter With Method Call

1. Verifique se a chamada de consulta não necessita de parâmetros que só existam no método atual.
2. No código do método principal, substitua todas as referências ao parâmetro que está sendo substituído.
3. Apague o parâmetro não utilizado.

Introduce Parameter Object

Quando métodos diferentes repetem um mesmo grupo de parâmetros.





Introduce Parameter Object

1. Crie uma nova classe que representa o grupo de parâmetros.
2. Adicione o novo objeto como parâmetro nos métodos que tinham o grupo de parâmetros.
3. Exclua os parâmetros antigos dos locais que eram chamados, substitua por uma chamada ao atributo do novo objeto.

Replace Conditional with Polymorphism

Quando uma condicional tem várias ações, dependendo do tipo ou propriedades do objeto.

```
class Bird {
  // ...
  double getSpeed() {
    switch (type) {
      case EUROPEAN:
        return getBaseSpeed();
      case AFRICAN:
        return getBaseSpeed() - getLoadFactor() * numberOfCoco
      case NORWEGIAN_BLUE:
        return (isNailed) ? 0 : getBaseSpeed(voltage);
    }
    throw new RuntimeException("Should be unreachable");
  }
}
```

```
abstract class Bird {
  // ...
  abstract double getSpeed();
}

class European extends Bird {
  double getSpeed() {
    return getBaseSpeed();
  }
}
class African extends Bird {
  double getSpeed() {
    return getBaseSpeed() - getLoadFactor() * numberOfCoconuts
  }
}
class NorwegianBlue extends Bird {
  double getSpeed() {
    return (isNailed) ? 0 : getBaseSpeed(voltage);
  }
}

// Somewhere in client code
speed = bird.getSpeed();
```

Replace Conditional with Polymorphism

1. Crie as subclasses com base no atributo usado na condicional
2. Se a condicional estiver em um método que também executa outras ações, execute Extract Method.
3. Para cada subclasse de hierarquia, redefina o método que contém a condicional e copie o código da ramificação condicional correspondente para esse local.
4. Exclua esta ramificação do condicional.
5. Repita a substituição até que a condicional esteja vazia. Em seguida, exclua a condicional e declare o método abstrato.

Fontes

- Fowler, Martin. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999.
- https://luzkan.github.io/smells/
- https://refactoring.guru/

Obrigado!