

ACOPLADORES

# ACOPLADORES

Acoplamento  
excessivo

Depende do  
contexto

# ALGUNS INDICATIVOS

**Funcionalidade  
invejosa**

**Intimidade  
inapropriada**

**Cadeias**

**O homem do meio**

## Funcionalidade invejosa

Acessa dados de  
outra classe

Poucos acessos à  
sua própria  
classe

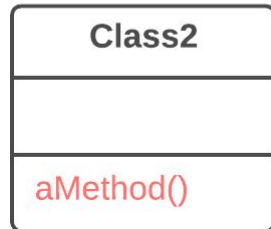
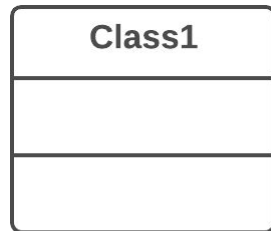
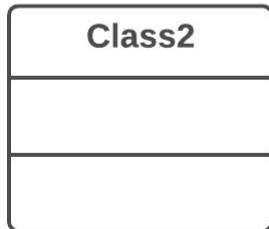
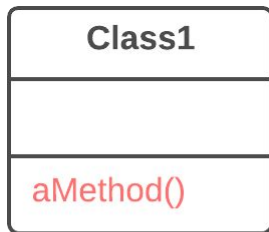
Movimentação de  
atributos

**Funcionalidade  
invejosa**

**Mover método**

**Funcionalidade  
invejosa**

**Mover método**



**Mais coesão**

**Menos acoplamento**

**Diminuir  
dependências**

Funcionalidade  
invejosa

Mover método

Verificar  
atributos  
usados

Declarar o  
método na  
nova classe

Decidir como  
vai receber o  
objeto

Referenciar o  
novo método

Se forem usados  
apenas no método,  
mova-os também

Se forem usados em  
outros métodos,  
mover os métodos (?)

Se já tiver  
atributo ou método  
que retorna o  
objeto apropriado,  
pula

Se for possível  
deletar o método  
por completo,  
referenciar onde  
foi chamado

**Funcionalidade  
invejosa**

**Extrair método**



Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```

Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

Mais legível

Menos duplicado e  
mais reuso

Isola partes  
independentes

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**



**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

**Funcionalidade  
invejosa**

**Extrair variável**

# ALGUNS INDICATIVOS

Funcionalidade  
invejosa

Intimidade  
inapropriada

Cadeias

O homem do meio

**Intimidade  
inapropriada**

Acessa atributos  
de outra classe

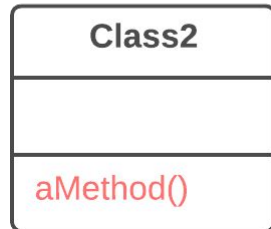
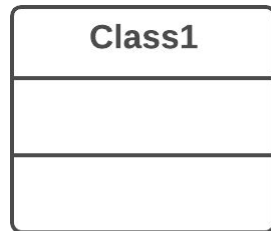
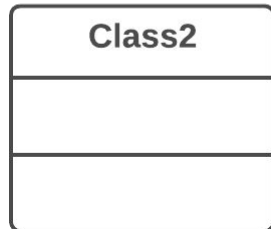
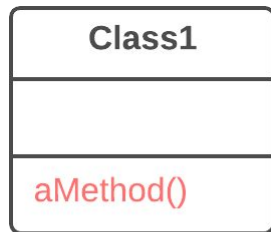


**Funcionalidade  
invejosa**

**Mover método**

**Funcionalidade  
invejosa**

**Mover método**



**Mais coesão**

**Menos acoplamento**

**Diminuir  
dependências**

Funcionalidade  
invejosa

Mover método

Verificar  
atributos  
usados

Se forem usados  
apenas no método,  
mova-os também

Se forem usados em  
outros métodos,  
mover os métodos (?)

Declarar o  
método na  
nova classe

Decidir como  
vai receber o  
objeto

Se já tiver  
atributo ou método  
que retorna o  
objeto apropriado,  
pula

Referenciar o  
novo método

Se for possível  
deletar o método  
por completo,  
referenciar onde  
foi chamado

**Funcionalidade  
invejosa**

**Mover método**

**Funcionalidade  
invejosa**

**Mover método**

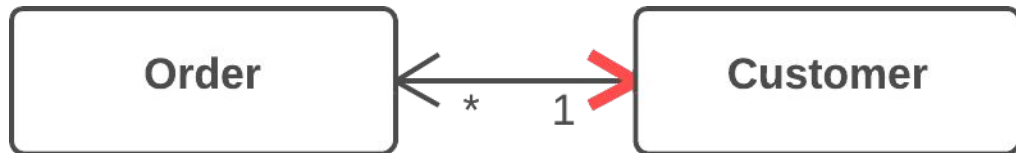
**Mover  
atributo**

**Funcionalidade  
invejosa**

**Relacionamento  
unidirecional**

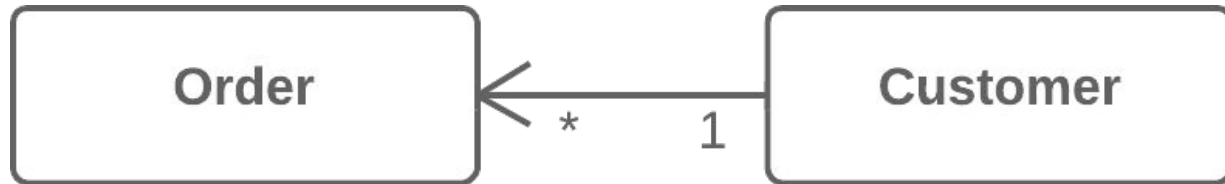
Funcionalidade  
invejosa

Relacionamento  
unidirecional



Menos acoplamento

Diminuir  
dependências



**Funcionalidade  
invejosa**

**Relacionamento  
unidirecional**

**Verificar  
necessidade do  
relacionamento**

**Usar através  
de parâmetros  
ou métodos**

**Deletar  
campos  
não  
usados**



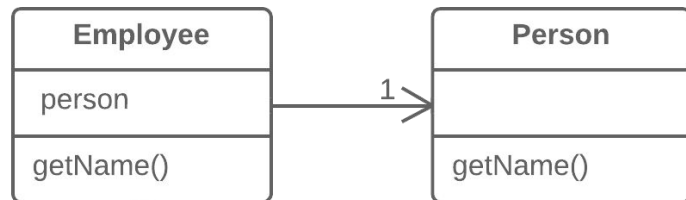
**Funcionalidade  
invejosa**

**Herança**

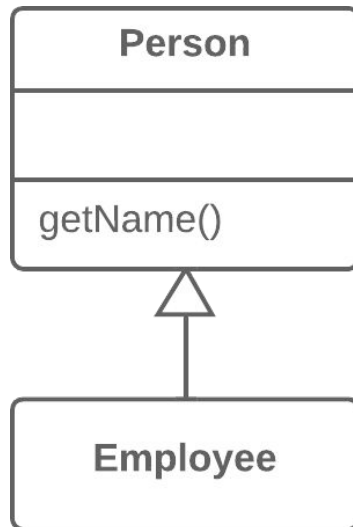
**Trocar  
delegação por  
herança**

**Funcionalidade  
invejosa**

**Herança**



`return $this->person->getName();`



**Mais coesão**

**Funcionalidade  
invejosa**

**Herança**

**Transformar em  
subclasse**

**Colocar  
objeto onde  
fazia  
referência**

**Deletar  
métodos  
que  
faziam  
delegação**

# ALGUNS INDICATIVOS

Funcionalidade  
invejosa

Intimidade  
inapropriada

Cadeias

O homem do meio

## Cadeias

`$a->b()->c()->d()`

Dependência

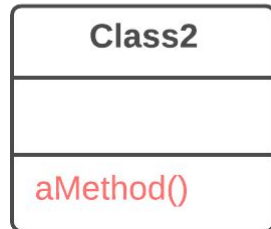
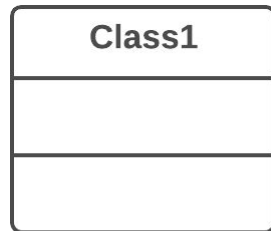
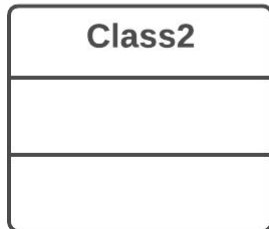
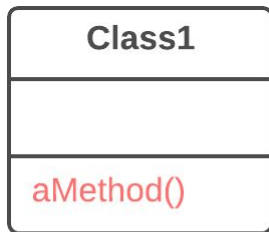
Movimentação de  
atributos

**Funcionalidade  
invejosa**

**Mover método**

**Funcionalidade  
invejosa**

**Mover método**



**Mais coesão**

**Menos acoplamento**

**Diminuir  
dependências**

Funcionalidade  
invejosa

Mover método

Verificar  
atributos  
usados

Se forem usados  
apenas no método,  
mova-os também

Se forem usados em  
outros métodos,  
mover os métodos (?)

Declarar o  
método na  
nova classe

Decidir como  
vai receber o  
objeto

Se já tiver  
atributo ou método  
que retorna o  
objeto apropriado,  
pula

Referenciar o  
novo método

Se for possível  
deletar o método  
por completo,  
referenciar onde  
foi chamado



**Funcionalidade  
invejosa**

**Extrair método**

Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```

Funcionalidade  
invejosa

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

Mais legível

Menos duplicado e  
mais reuso

Isola partes  
independentes

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**



**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

**Funcionalidade  
invejosa**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

# ALGUNS INDICATIVOS

Funcionalidade  
invejosa

Intimidade  
inapropriada

Cadeias

O homem do meio

0 homem do meio

`$a->b()->c()->d()`

Delega outra  
classe

Cadeias

Laranja



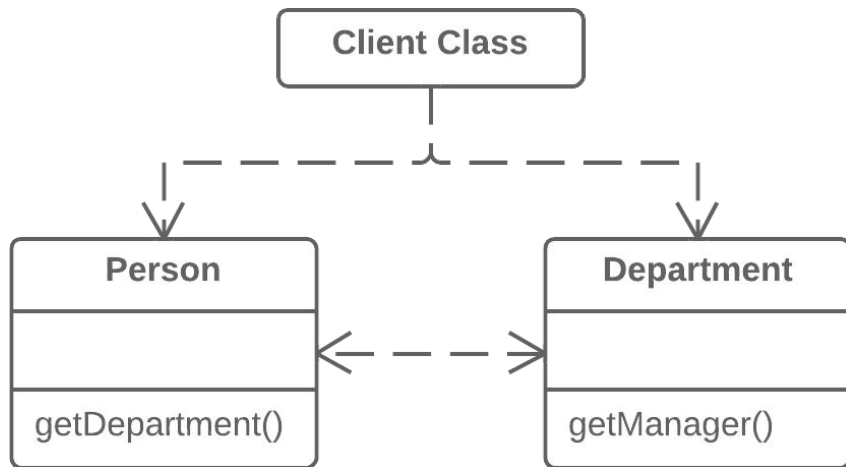
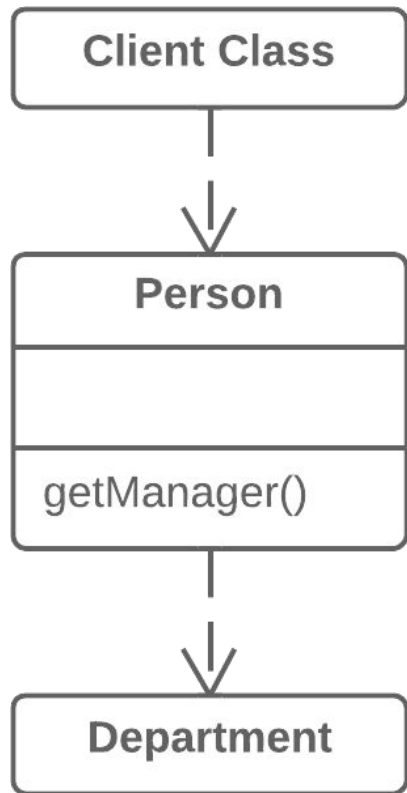
0 homem do meio

Remover homem do  
meio

0 homem do meio

Remover homem do meio

Mais coesão



**0 homem do meio**

**Remover homem do  
meio**

**Criar método de acesso  
para classe que foi  
delegada**

**Trocar  
métodos**

FIM

Intimidade  
inapropriada

Extraír variável

Mais legível

```
def renderBanner(self):  
    if (self.platform.toUpperCase().indexOf(&quot;MAC&quot;)> -1) and \  
        (self.browser.toUpperCase().indexOf(&quot;IE&quot;)> -1) and \  
        self.wasInitialized() and (self.resize > 0):  
        # do something
```

```
def renderBanner(self):  
    isMacOs = self.platform.toUpperCase().indexOf(&quot;MAC&quot;)> -1  
    isIE = self.browser.toUpperCase().indexOf(&quot;IE&quot;)> -1  
    wasResized = self.resize > 0  
  
    if isMacOs and isIE and self.wasInitialized() and wasResized:  
        # do something
```

**Intimidade  
inapropriada**

**Extrair variável**

**Criar  
variável**

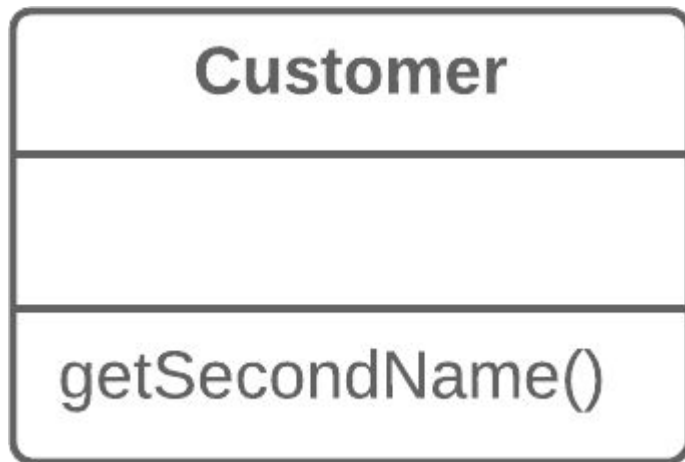
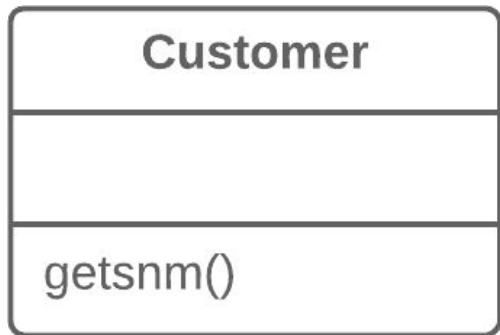
**Trocar parte da  
expressão pela  
variável**

**Intimidade  
inapropriada**

**Extraír variável**

**Mais legível**

**Método cresceu e o  
nome precisa  
refletir a  
funcionalidade**



**Intimidade  
inapropriada**

**Extraír variável**

**Crear  
método com  
novo nome**

**Copiar  
código para  
o novo  
método**

**Chamar  
novo  
método no  
antigo**

**Trocar  
referências  
para o novo  
método**

**Apagar o  
método  
antigo**



**Comentários**

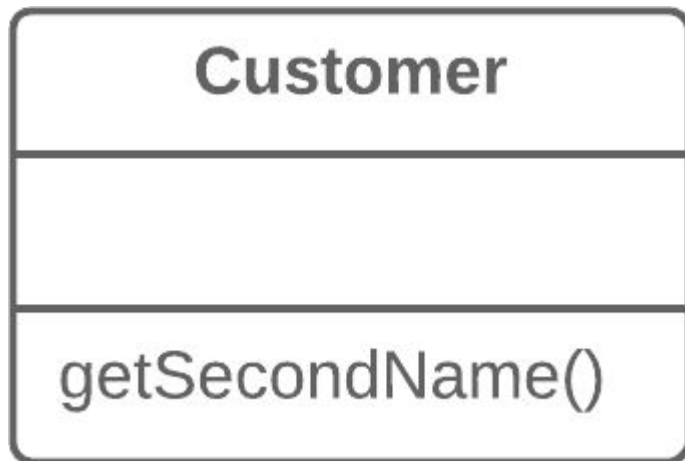
**Renomear método**

Comentários

Extrair variável

Mais legível

Método cresceu e o  
nome precisa  
refletir a  
funcionalidade



**Comentários**

**Extraír variável**

**Crear  
método com  
novo nome**

**Copiar  
código para  
o novo  
método**

**Chamar  
novo  
método no  
antigo**

**Trocar  
referências  
para o novo  
método**

**Apagar o  
método  
antigo**

# ALGUNS INDICATIVOS

**Comentários**

**Código duplicado**

**Classe preguiçosa**

**Classe de dados**

**Código morto**

**Generalidade  
especulativa**

**Código duplicado**

Falta de  
comunicação

Parece  
diferente, mas  
faz a mesma  
coisa

Código quase  
certo (c+v)

**Código duplicado**

**Extrair método**

Código duplicado

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

Código duplicado

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```



Código duplicado

Extrair método

```
def printBill(self):  
    self.printMessage()  
  
    # print details  
    print("name:", self.name)  
    print("amount:", self.getQuantity())
```

```
def printBill(self):  
    self.printMessage()  
    self.printDetails(self.getQuantity())  
  
def printDetails(self, quantity):  
    print("name:", self.name)  
    print("amount:", quantity)
```

Mais legível

Menos duplicado e  
mais reuso

Isola partes  
independentes

Código duplicado

Extrair método

Extrair método

Dois ou mais métodos  
na mesma classe

Extrair método

Subclasse de mesmo  
nível

Mover método classe  
pai

Mover para o pai e  
usar

Método construtor

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro



**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

**Código duplicado**

**Extrair método**

**Criar  
novo  
método**

**Coloca o  
código no  
novo método**

**Chama o  
método**

**Retorna se  
necessário**

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

Código duplicado

Extrair método

Criar  
novo  
método

Coloca o  
código no  
novo método

Chama o  
método

Retorna se  
necessário

Se as variáveis  
não são usadas  
fora desse pedaço  
de código, serão  
variáveis locais  
dele do método

Se as variáveis  
forem usadas  
anteriormente,  
deve, ser passadas  
como parâmetro

Se a variável  
assume um novo  
valor, talvez deve  
retornar.  
Verifique

**Código duplicado**

**Método template**

**Código duplicado**

**Método template**

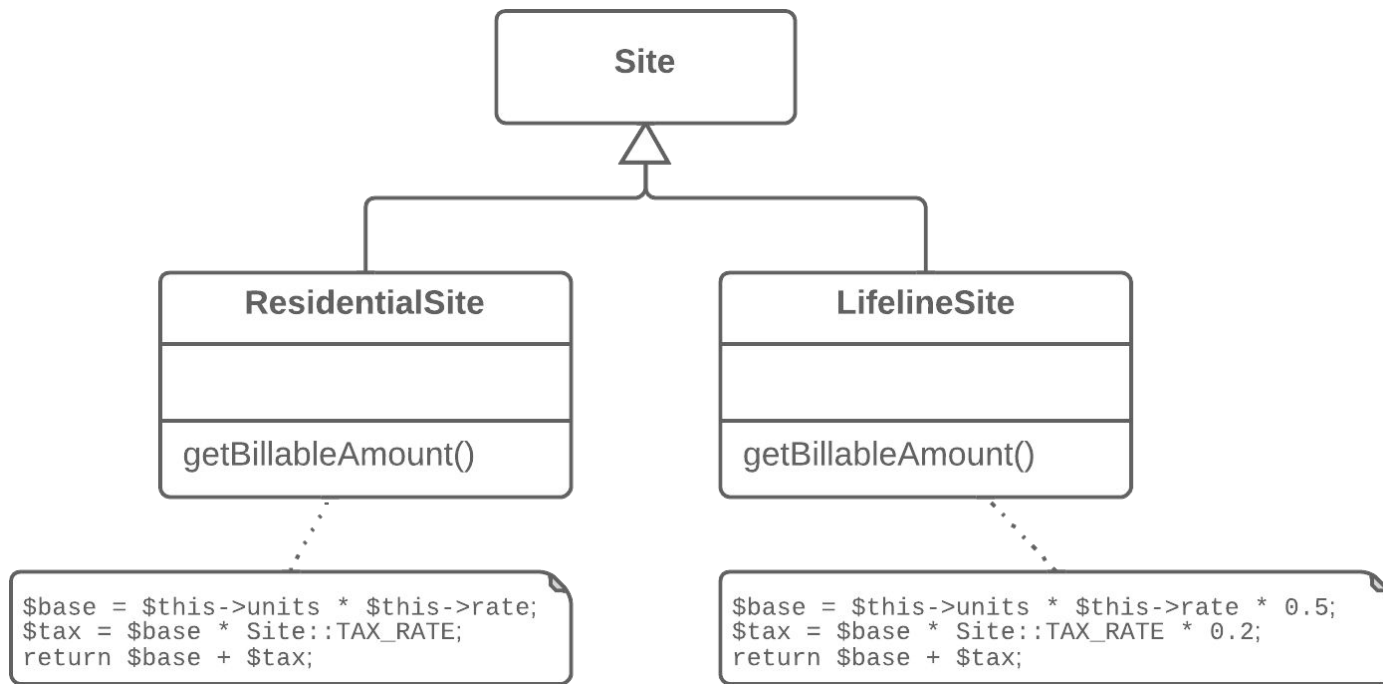
**Duplicação sutil**

**Serve o mesmo  
propósito**

**De formas  
diferentes**

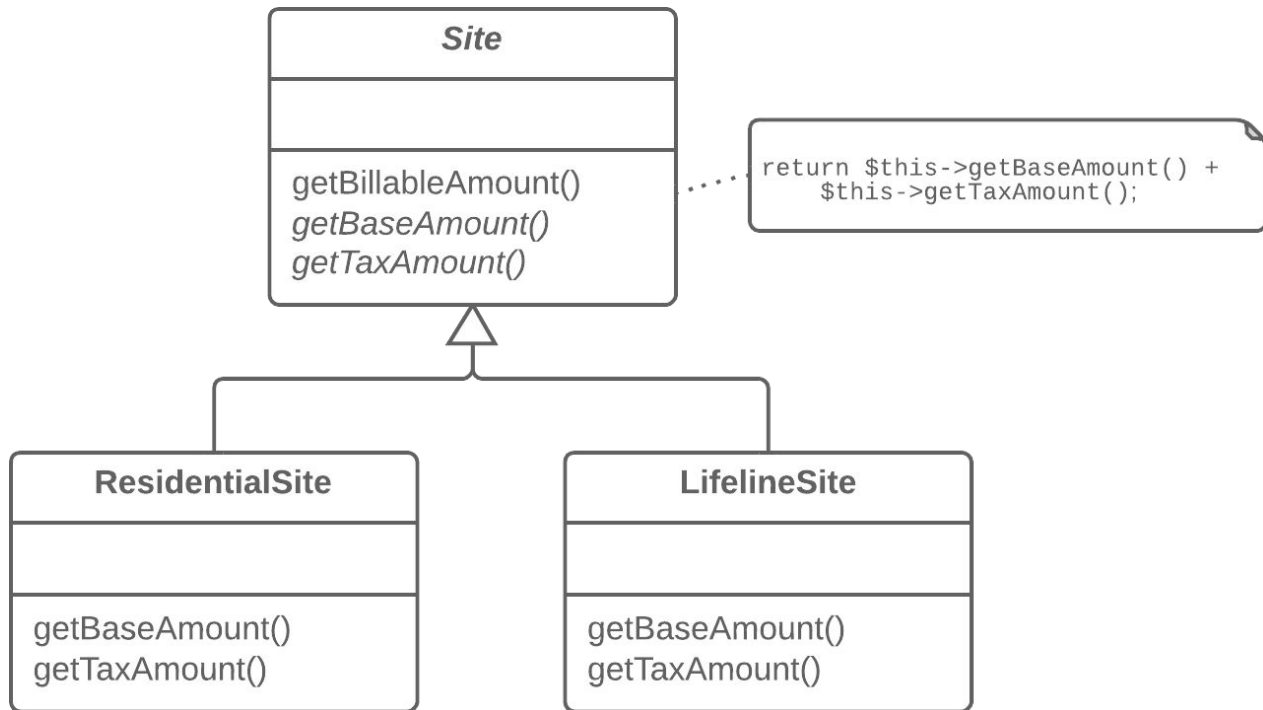
Código duplicado

Método template



Código duplicado

Método template





**Código duplicado**

**Método template**

**Dividir  
algoritmo em  
partes comuns**

**Mover  
métodos em  
comum para  
cima**

**Mover  
assinaturas  
dos demais  
para cima**

**Sobrescrever  
métodos  
diferentes**

# ALGUNS INDICATIVOS

**Comentários**

**Código duplicado**

**Classe preguiçosa**

**Classe de dados**

**Código morto**

**Generalidade  
especulativa**

# ALGUNS INDICATIVOS

**Comentários**

**Código duplicado**

**Classe preguiçosa**

**Classe de dados**

**Código morto**

**Generalidade  
especulativa**

## Classe preguiçosa

Manter código é  
custoso

Não está sendo  
usada

Custo  
desnecessário

Suporte a  
desenvolvimento  
futuro

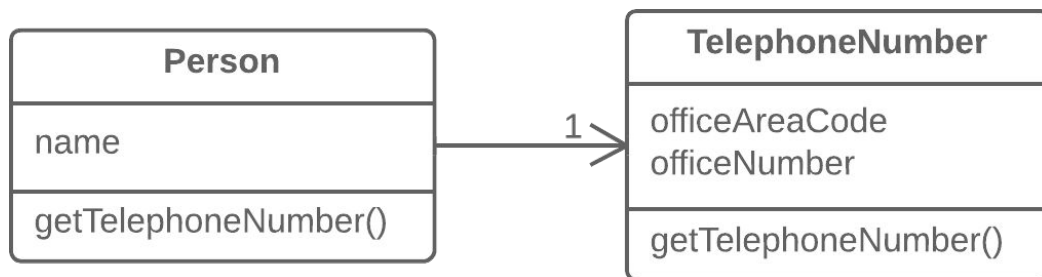
Pouco retorno

Classe preguiçosa

Classe interna

Classe preguiçosa

Classe interna



Classe preguiçosa

Classe interna

## Person

name  
officeAreaCode  
officeNumber

getTelephoneNumber()

**Classe preguiçosa**

**Classe interna**

**Técnica mover  
método**

**Técnica  
mover  
atributo**

**Deletar  
original**



**Classe preguiçosa**

**Compactar  
hieraquia**

Classe preguiçosa

Compactar  
hieraquia

Complexidade  
desnecessária

Mais inteligível

Employee



Salesman

Employee

Classe preguiçosa

Compactar  
hieraquia

Complexidade  
desnecessária

Mais inteligível

Employee



Salesman

Employee

Mais de  
uma  
subclasse

**Classe preguiçosa**

**Compactar  
hieraquia**

**Técnica mover  
método**

**Técnica  
mover  
atributo**

**Deletar  
original**

# ALGUNS INDICATIVOS

**Comentários**

**Código duplicado**

**Classe preguiçosa**

**Classe de dados**

**Código morto**

**Generalidade  
especulativa**

# ALGUNS INDICATIVOS

Comentários

Código duplicado

Classe preguiçosa

Classe de dados

Código morto

Generalidade  
especulativa

## Classe de dados

Atributos

Métodos de  
acesso

Nenhum  
comportamento

Avaliar a  
utilização

Trazer  
comportamento

Limitar acesso

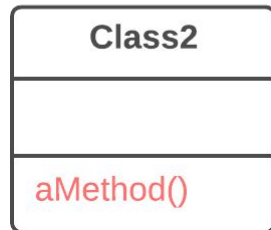
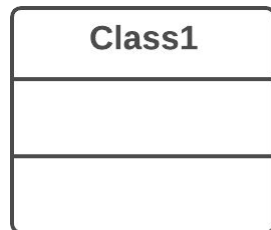
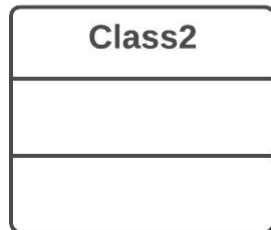
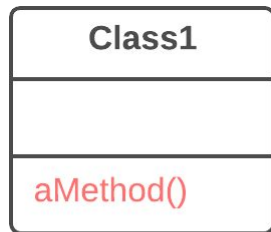
**Classe de dados**

**Mover método**



Switch complexos

Mover método



Mais coesão

Menos acoplamento

Diminuir dependências

Switch complexos

Mover método

Verificar  
atributos  
usados

Declarar o  
método na  
nova classe

Decidir como  
vai receber o  
objeto

Referenciar o  
novo método

Se forem usados  
apenas no método,  
mova-os também

Se forem usados em  
outros métodos,  
mover os métodos (?)

Se já tiver  
atributo ou método  
que retorna o  
objeto apropriado,  
pula

Se for possível  
deletar o método  
por completo,  
referenciar onde  
foi chamado

**Classe de dados**

**Encapsular  
atributos**

**Switch complexos**

**Mover método**

Atributo público

Atributo privado

Métodos de acesso

**Mais coesão**

**Menos acoplamento**

**Diminuir  
dependências**

**Switch complexos**

**Mover método**

**Criar  
métodos de  
acesso**

**Utilizar  
métodos**

**Tornar  
atributos  
privados**

# ALGUNS INDICATIVOS

Comentários

Código duplicado

Classe preguiçosa

Classe de dados

Código morto

Generalidade  
especulativa

# ALGUNS INDICATIVOS

Comentários

Código duplicado

Classe preguiçosa

Classe de dados

Código morto

Generalidade  
especulativa

## Código morto

Variável  
esquecida

Parâmetro não  
usado

Método não é  
chamado

Remover código  
não usado



# ALGUNS INDICATIVOS

Comentários

Código duplicado

Classe preguiçosa

Classe de dados

Código morto

Generalidade  
especulativa

# ALGUNS INDICATIVOS

Comentários

Código duplicado

Classe preguiçosa

Classe de dados

Código morto

Generalidade  
especulativa

## Generalidade especulativa

Mudança de  
requisitos

Suporte para o  
futuro

Nenhum momento  
para limpar

Remover arquivos  
desnecessários

Remover código  
não usado

## Código morto

Mudança de  
requisitos

Suporte para o  
futuro

Nenhum momento  
para limpar

Remover arquivos  
desnecessários

Remover código  
não usado

Técnicas de outros  
dispensáveis