

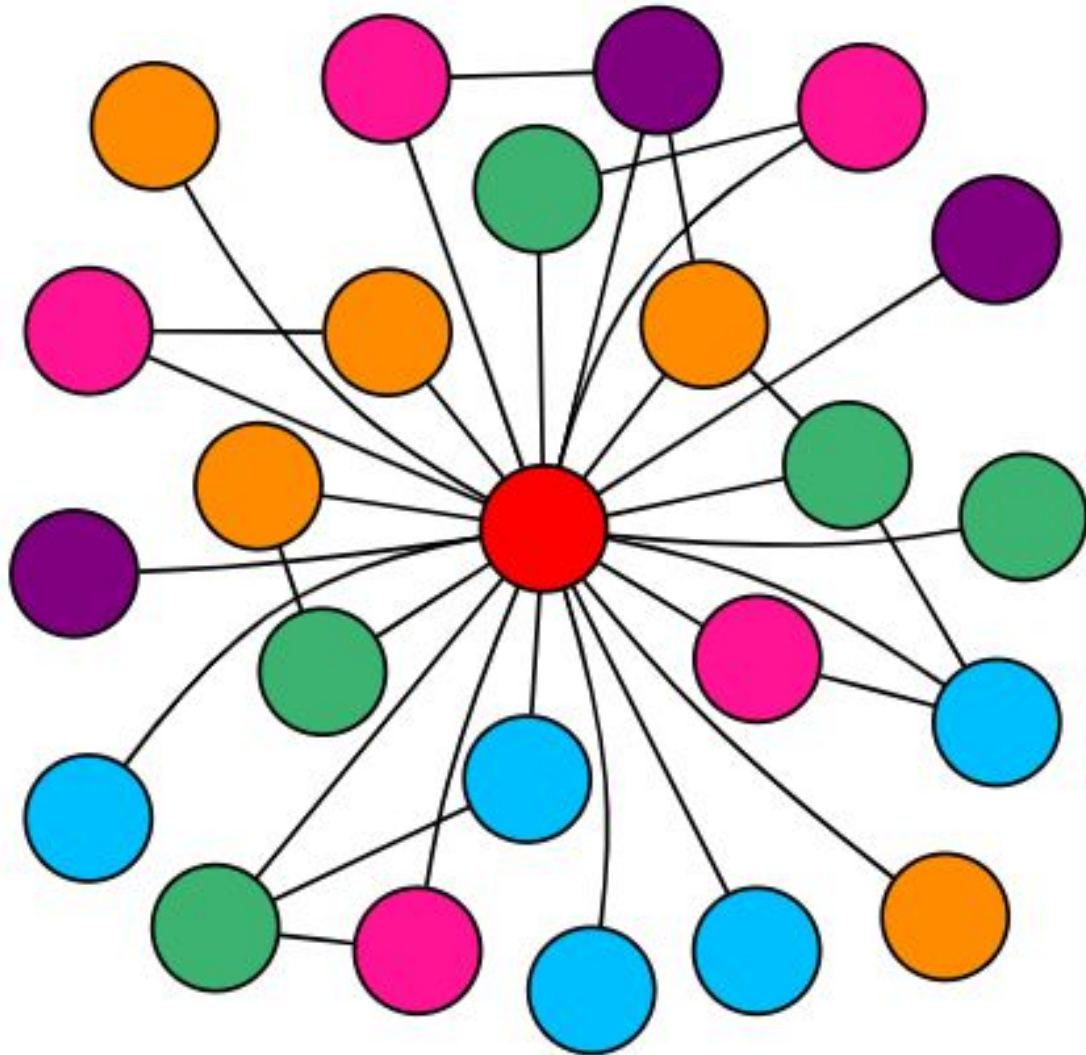
Evolução e refatoração

Matheus Barbosa
matheus.barbosa@dcx.ufpb.br

Refatoração?
O que é isso?

**Problemas de
reuso**

**Problemas de
modularidade**



Code smells

Odores de código são indicadores de problemas que podem ser resolvidos durante a refatoração. Odores de código são fáceis de identificar e corrigir, mas podem ser apenas sintomas de um problema mais profundo com o código.

- Método/classe Longo(a)
- Lista de parâmetros longa
- Nomes misteriosos
- Código Morto
- Comentários
- Inveja de recursos

<https://refactoring.guru/refactoring/smells>

Algumas ferramentas para detectar code smells

sonarqube 

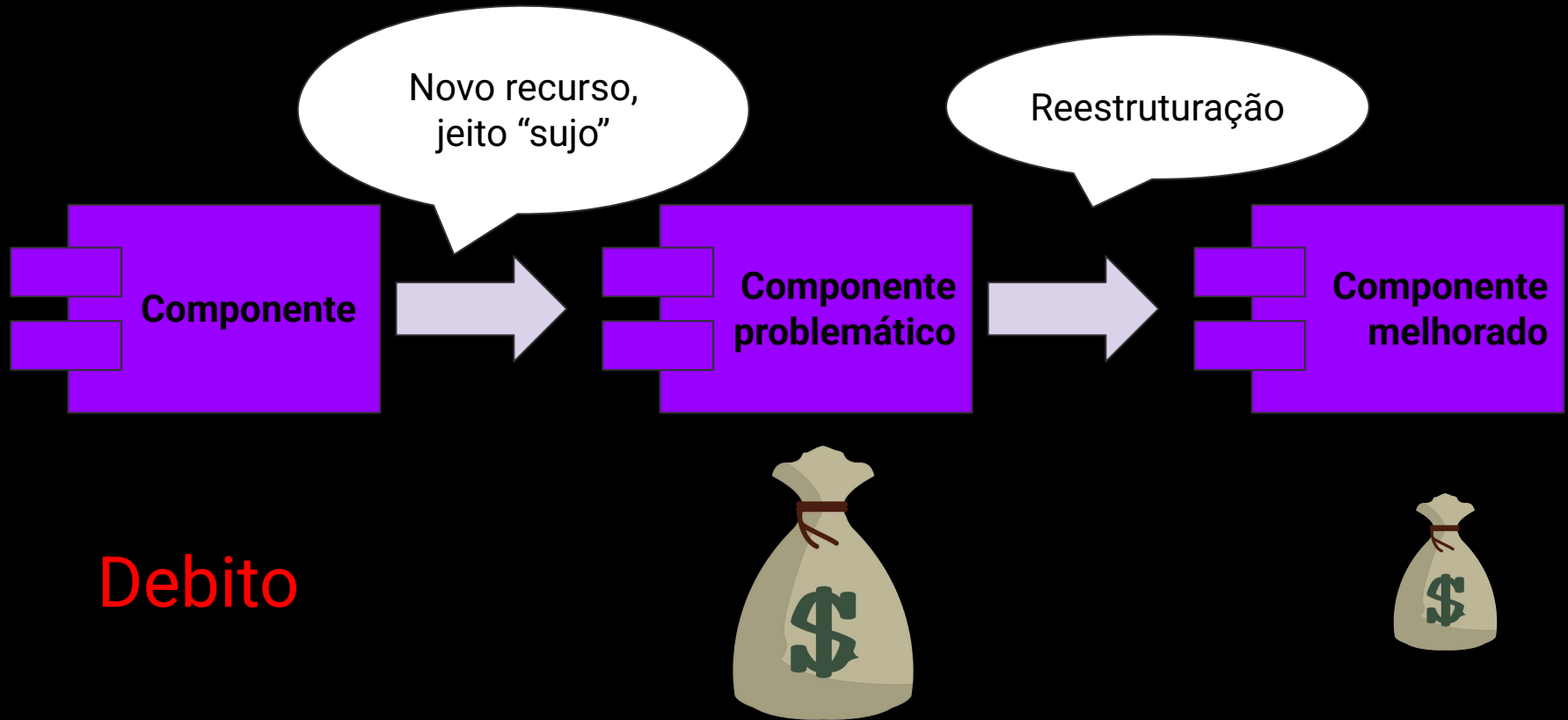
 TM
FindBugs

Pmd
DON'T SHOOT THE MESSENGER

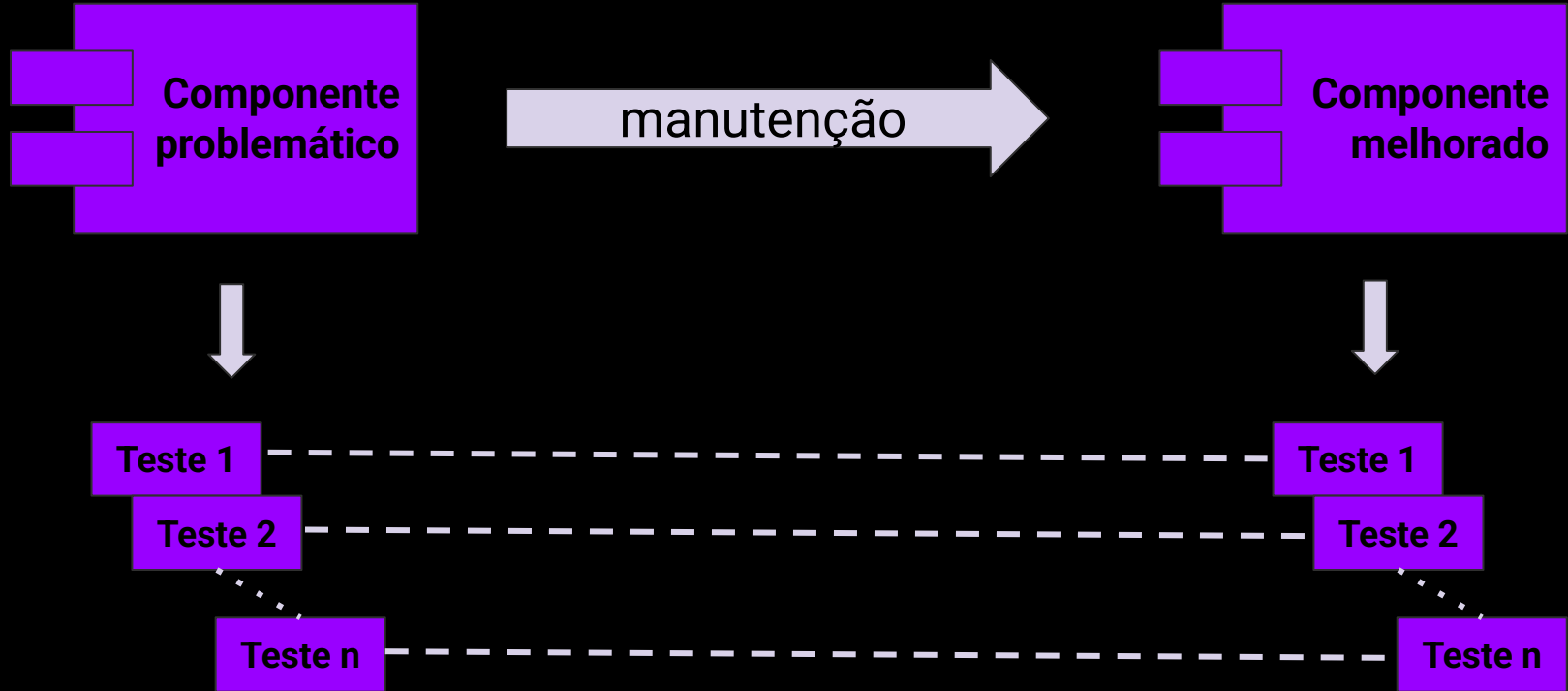
checkstyle 

Quando e como resolver
esses problemas?

Debito técnico



Preservando o comportamento



Refatorações são:

O processo de melhoria da
qualidade interna de um
software, sem alterar seu
comportamento observável.

Testes não garantem a
preservação do
comportamento

Testes são uma **aproximação** para preservação do comportamento

A aproximação é tão boa quanto o conjunto de testes

O que realmente queremos
dizer com preservação do
comportamento?

Observações induzem noções de equivalência

- **Entrada-saída** (foco em refatoração)
- Sequência intermediária de eventos possíveis
- Sequências de eventos levando a falhas e divergências
- Tempo dos eventos
- Probabilidade dos eventos
- Uso de memória

Preservação de comportamento (refinamento)

Ao substituir um componente ou programa por outro, o novo ainda se comporta corretamente (ou "melhor") em termos das saídas esperadas para as mesmas entradas

Seguindo padrões de
transformação conhecidos...

Para orientar o processo,
podemos usar ferramentas e
modelos de transformação
existentes (também
chamados de refatorações!)

O livro "Refactoring: Improving the Design of Existing Code" (1999), de Martin Fowler

<https://refactoring.com/catalog/>

<https://refactoring.guru/refactoring/techniques>

Série especial da Addison-Wesley

*"Qualquer tolo escreve um código que um computador possa entender.
Bons programadores escrevem códigos que os seres humanos podem entender."
—M. Fowler (1999)*



REFATORAÇÃO

Aperfeiçoando o design de códigos existentes

Martin Fowler

com contribuições de
Kent Beck



novatec

SEGUNDA EDIÇÃO

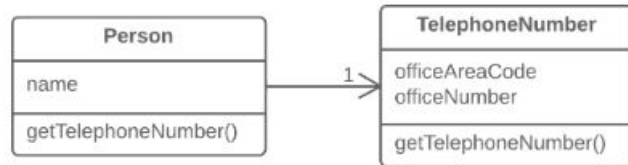
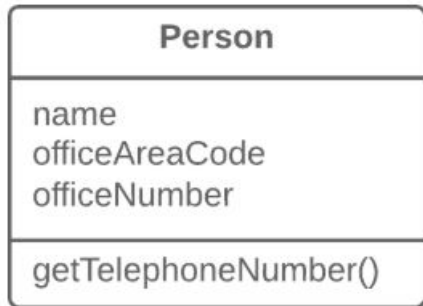
Extract Class

Problem

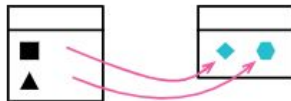
When one class does the work of two, awkwardness results.

Solution

Instead, create a new class and place the fields and methods responsible for the relevant functionality in it.



Extract Class



```
class Person {  
  get officeAreaCode() {return this._officeAreaCode;}  
  get officeNumber() {return this._officeNumber;}  
}
```



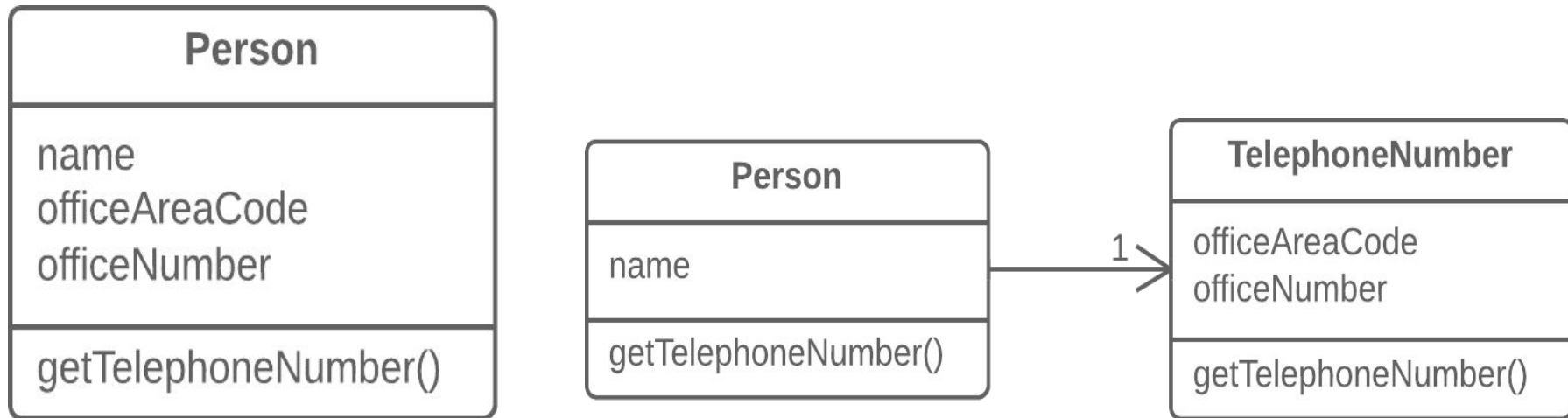
```
class Person {  
  get officeAreaCode() {return this._telephoneNumber.areaCode;}  
  get officeNumber() {return this._telephoneNumber.number;}  
}  
  
class TelephoneNumber {  
  get areaCode() {return this._areaCode;}  
  get number() {return this._number;}  
}
```

inverse of *Inline Class*

Extract Class

Problema: Quando uma classe faz o trabalho de duas, o resultado é estranheza.

Solução: Em vez disso, crie uma nova classe e coloque nela os campos e métodos responsáveis pela funcionalidade relevante.



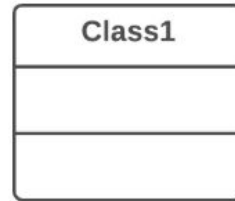
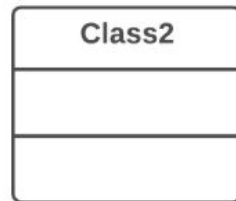
Move Field

Problem

A field is used more in another class than in its own class.

Solution

Create a field in a new class and redirect all users of the old field to it.

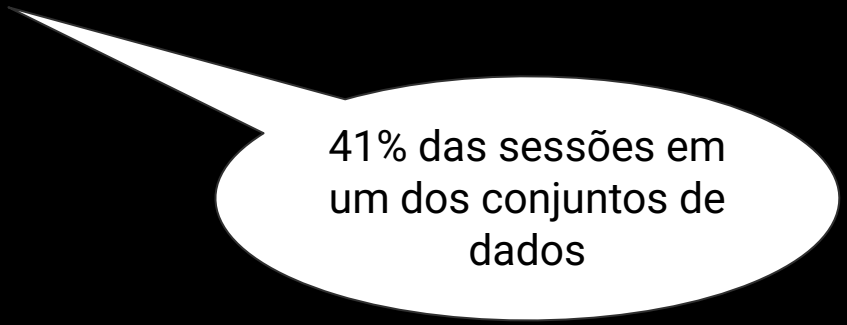


Esteja ciente das
pré-condições de refatoração
e outros detalhes

Como as refatorações ocorrem na pratica?

Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. 2009. How we refactor, and how we know it. In Proceedings of the 31st International Conference on Software Engineering (ICSE '09). IEEE Computer Society, USA, 287–297. <https://doi.org/10.1109/ICSE.2009.5070529>

As refatorações são realizadas com frequência



41% das sessões em um dos conjuntos de dados

Refactoring Tool	Toolsmiths				Mylyn				Users				Everyone	
	Uses	Use %	Batched	Batched %	Uses	Use %	Batched	Batched %	Uses	Use %	Batched	Batched %	Uses	Use %
Rename	670	28.7%	283	42.2%	2706	53.6%	1146	42.4%	1862	61.5%	1009	54.2%	49672	71.8%
Extract Local Variable	568	24.4%	127	22.4%	260	5.2%	57	21.9%	322	10.6%	106	32.9%	4917	7.1%
Inline	349	15.0%	132	37.8%	110	2.2%	52	47.3%	137	4.5%	52	38.0%	1426	2.1%
Extract Method	280	12.0%	28	10.0%	316	6.3%	27	8.5%	259	8.6%	57	22.0%	3345	4.8%
Move	147	6.3%	50	34.0%	958	19.0%	459	47.9%	171	5.6%	98	57.3%	3869	5.6%
Change Method Signature	93	4.0%	26	28.0%	191	3.8%	73	38.2%	55	1.8%	20	36.4%	1642	2.4%
Convert Local To Field	92	3.9%	12	13.0%	22	0.4%	10	45.5%	27	0.9%	10	37.0%	504	0.7%
Introduce Parameter	41	1.8%	20	48.8%	1	0.1%	0	-	16	0.5%	11	68.8%	162	0.2%
Extract Constant	22	0.9%	6	27.3%	278	5.5%	91	32.7%	81	2.7%	48	59.3%	1039	1.5%
Convert Anonymous To Nested	18	0.8%	0	0.0%	25	0.5%	0	0.0%	19	0.6%	7	36.8%	86	0.1%
Move Member Type to New File	15	0.6%	0	0.0%	55	1.1%	4	7.3%	12	0.4%	5	41.7%	343	0.5%
Pull Up	12	0.5%	0	0.0%	17	0.3%	2	11.8%	36	1.2%	4	11.1%	397	0.6%
Encapsulate Field	11	0.5%	8	72.7%	29	0.6%	17	58.6%	4	0.1%	2	50.0%	406	0.6%
Extract Interface	2	0.1%	0	0.0%	29	0.6%	2	6.9%	15	0.5%	0	0.0%	492	0.7%
Generalize Declared Type	2	0.1%	0	0.0%	0	0.0%	0	-	4	0.1%	2	50.0%	56	0.1%
Push Down	1	0.1%	0	-	3	0.1%	2	66.7%	1	0.1%	0	-	80	0.1%
Infer Generic Type Arguments	0	0.0%	0	-	31	0.6%	13	41.9%	3	0.1%	0	0.0%	179	0.3%
Use Supertype Where Possible	0	0.0%	0	-	1	0.1%	0	-	2	0.1%	0	0.0%	47	0.1%
Introduce Factory	0	0.0%	0	-	0	0.0%	0	-	1	0.1%	0	-	31	0.1%
Extract Superclass	7	0.3%	0	0.0%	16	0.3%	0	0.0%	*	-	*	*	158	0.2%
Extract Class	1	0.1%	0	0.0%	0	0.0%	0	-	*	-	*	*	289	0.4%
Introduce Parameter Object	0	0.0%	0	-	0	0.0%	0	-	*	-	*	*	64	0.1%
Introduce Indirection	0	0.0%	0	-	0	0.0%	0	-	*	-	*	*	51	0.1%
Total	2331	100%	692	29.7%	5048	100.0%	1955	38.7%	3027	100%	1431	47.3%	69255	100%

A refatoração baseada em IDE ocorre frequentemente (40%) em lotes, onde os desenvolvedores aplicam a mesma refatoração em sequência

Os programadores **intercalam**
frequentemente a refatoração com outros
tipos de mudanças nos programas

A maioria (90%) das refatorações são realizadas manualmente, sem a ajuda de ferramentas

Atividade

1. Identifique os *code smells* presentes
2. Aplique pelo menos **duas técnicas** de refatoração

```
public class ProcessadorDePedidos {

    public double calcularPrecoTotal(Pedido pedido) {
        double total = 0.0;
        for (Item item : pedido.getItems()) {
            double precoBase = item.getPreco() * item.getQuantidade();
            double desconto = 0.0;
            if (item.getCategoria().equals("eletronicos")) {
                desconto = precoBase * 0.1;
            } else if (item.getCategoria().equals("livros")) {
                desconto = precoBase * 0.05;
            }
            total += precoBase - desconto;
        }
        return total;
    }
}
```