UFC

Refatoração de Code Smells
Projeto Canarinho

Equipe:

Aluno 1: Eduardo Henrique Brito da Silva

Aluno 2: Francisco Jerônimo da Silva Júnior

Introdução

Esta biblioteca é um conjunto de utilitários para trabalhar com padrões brasileiros no Android.
Inspirado em: https://github.com/caelum/caelum-stella.

O foco aqui é o Android. Portanto, não é compatível com aplicações Java puras.

Entre os padrões implementados temos:

Formatador e validador de CPF

Formatador e validador de CNPJ

Formatador e validador de boleto bancário (e linha digitável)

Exemplo de uso

Validar um CPF:

```
if (Validador.CPF.ehValido(cpf))
    Toast.makeText(context, "Válido!", Toast.LENGTH_SHORT).show();
else
    Toast.makeText(context, "Inválido!", Toast.LENGTH_SHORT).show();
```

Formatar um CPF

```
String cpfFormatado = Formatador.CPF.formata(usuario.getCpf());
```

Ferramentas utilizadas

scitools
Understand

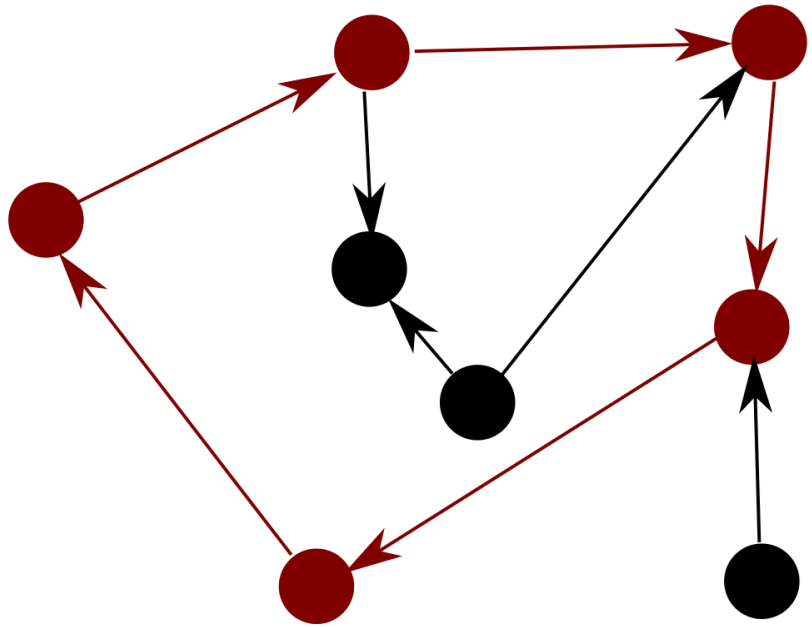
Code Smells

- Unnecessary Abstraction;
- Cyclically-dependent Modularization;
- Dead code;
- Magic Number;
- Too Many Arguments.

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
195	2555	395	3	3	875	57	57

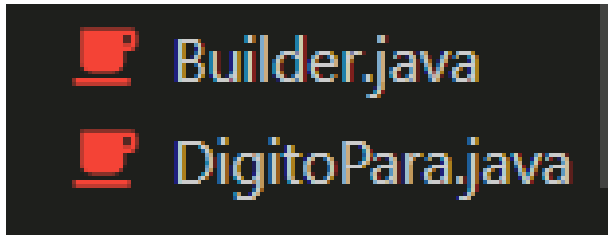
Cyclically-dependent Modularization

- Esse smell surge quando duas ou mais abstrações dependem uma da outra direta ou indiretamente.
- Possível Solução:
  - Mova o código que introduz a dependência cíclica para uma abstração completamente diferente.



Cyclically-dependent Modularization

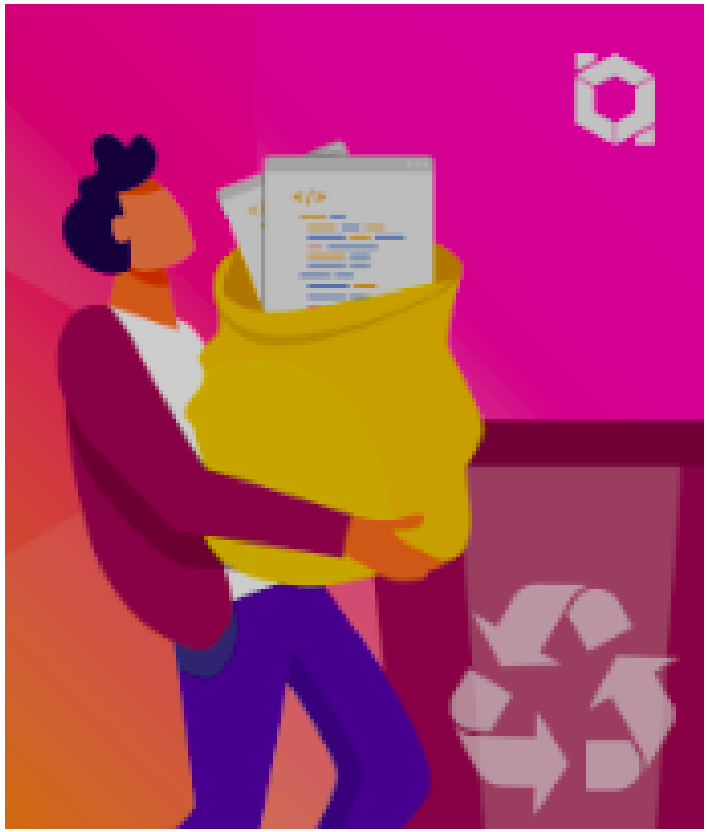
```
private DigitoPara(Builder builder)

protected DigitoPara(Builder.BuilderFinal builder)
```



NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
195	2554	395	3	3	875	57	57

Unnecessary Abstraction

- Esse smell ocorre quando uma abstração que na verdade não é necessária (e, portanto poderia ter sido evitada) é introduzida em um projeto de software.
- Possível solução:
  - Como sugere Fowler, “uma classe que não está fazendo o suficiente para se pagar deve ser eliminada”.



Unnecessary Abstraction

ANTES

```
private static class SingletonHolder {
  private static final FormatadorBoleto
 INSTANCE = new FormatadorBoleto();
}
```

DEPOIS

```
private static final FormatadorBoleto
 INSTANCE = new FormatadorBoleto();
```

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
195	2494	395	3	3	875	45	45

Dead code

- Uma variável, parâmetro, campo, método ou classe que não é usada.
- Possível solução:
    - Remoção do código.

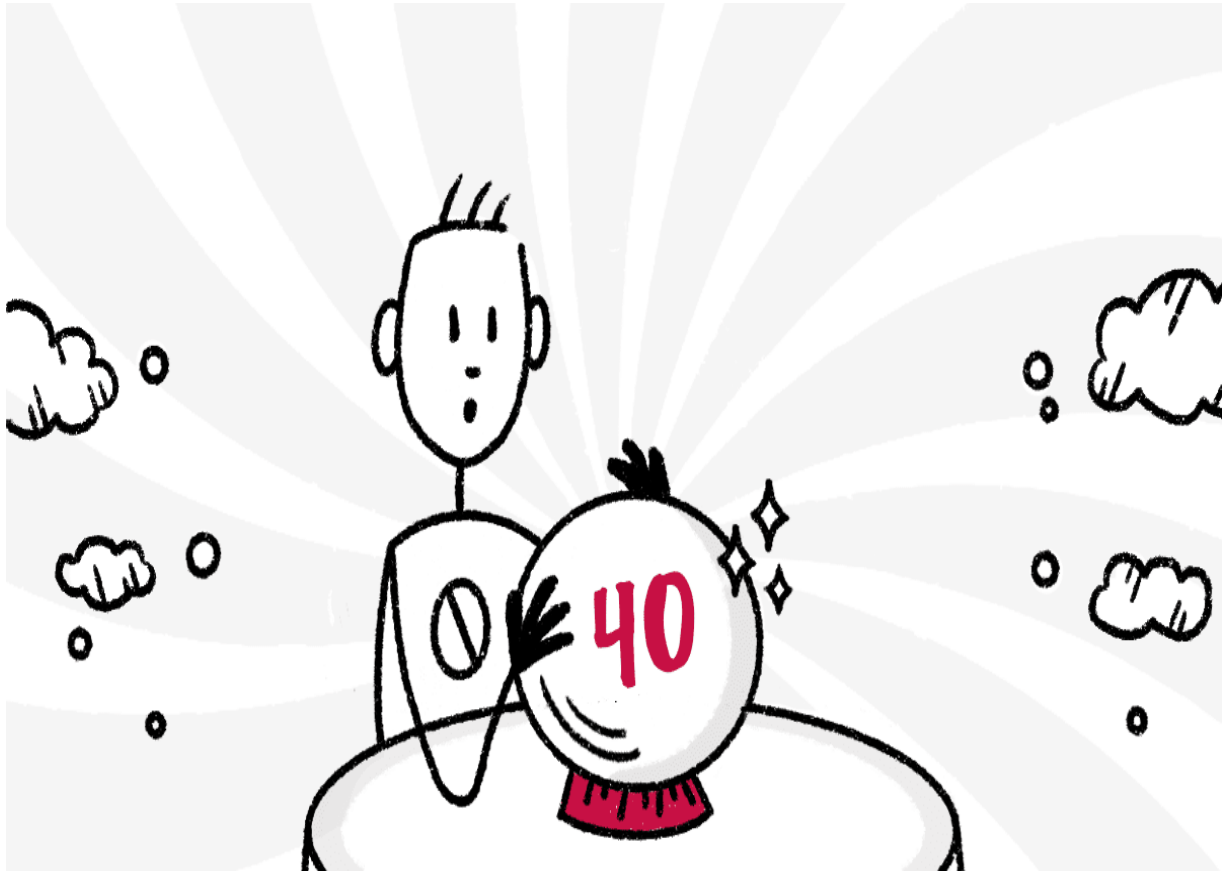
Dead code

```
@Override
public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    // Não faz nada aqui
}

@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
    // Não faz nada aqui
}
```

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
193	2471	391	3	3	875	43	43

Magic Number

- Seu código usa um número que tem um certo significado para ele.
- Possível solução:
    - Substitua esse número por uma constante que tenha um nome legível explicando o significado do número.



Magic Number

ANTES

```
private int somaDigitos(int total) {
    return (total / 10) + (total % 10);
}
```

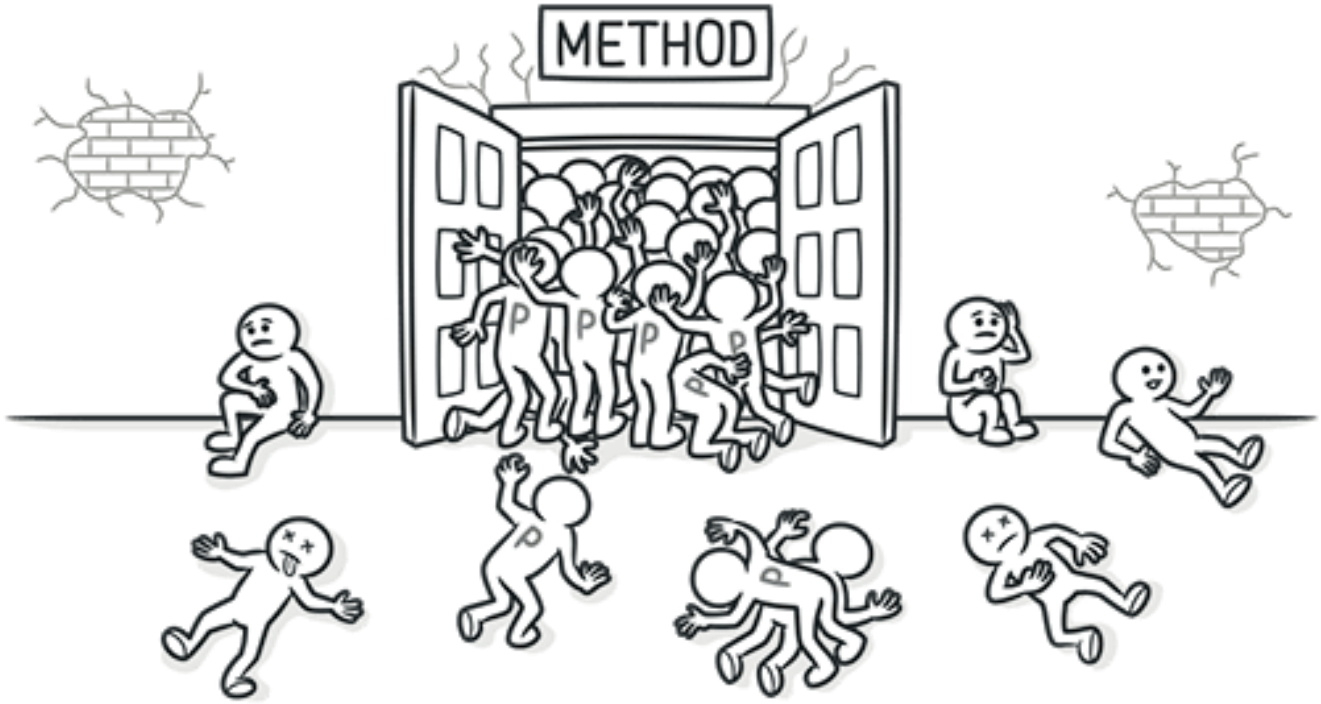
DEPOIS

```
private int somaDigitos(int total) {
    int div_10 = 10;
    int mod_10 = 10;
    return (total / div_10) + (total % mod_10);
}
```

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
193	2496	391	3	3	875	43	43

Too Many Arguments

- Mais de quatro parâmetros para um método.
- Possível solução:
  - Mesclar em um único objeto.



Too Many Arguments

ANTES

```
private boolean validaBloco(String valor, ResultadoParcial resultadoParcial, DigitoPara mod,
                            int tamanhoMinimo, int st, String mensagem) {
```

DEPOIS

```
private boolean validaBloco(String valor, ResultadoParcial resultadoParcial, DigitoPara mod,
                            Map<Integer, String> map) {
```

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
194	2513	392	3	3	0	43	43

Conclusão

- Antes

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
195	2555	395	3	3	875	57	57

- Depois

NOF	NOPF	NOM	NOPM	LOC	WMC	NC	DIT
194	2513	392	3	3	0	43	43

Muito Obrigado!!!
Dúvidas???