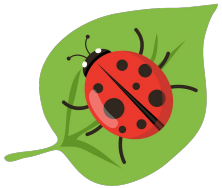
THE DEVELOPER'S CONFERENCE



Code Smells

O que são e como eles afetam a saúde do seu código

Jeziel Lago

Jeziel Lago

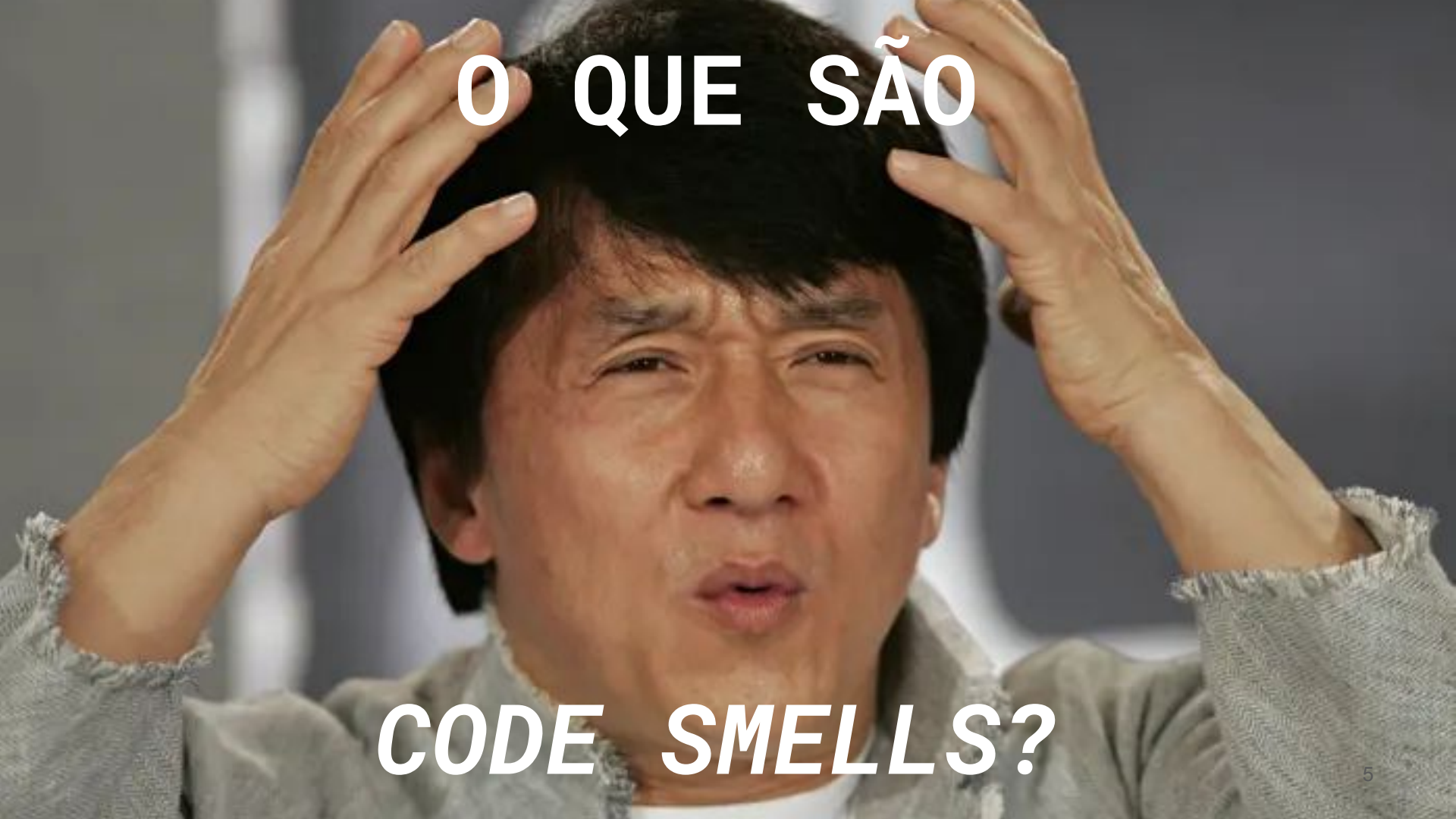
@jeziellago

Desenvolvedor Android na Zup Innovation

Roteiro

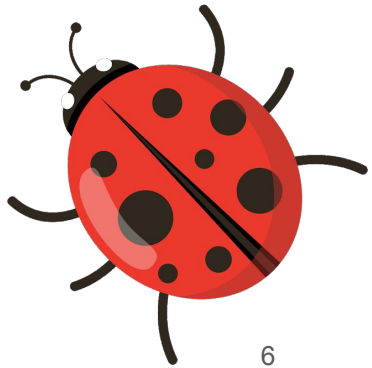
1. O que são code smells?
2. Code Smells clássicos
3. Categorias e exemplos de Code Smells
4. Detecção de Code Smells no Kotlin

Code Smells


O QUE SÃO
CODE SMELLS?

O que são code smells?

"Code Smells são certas *estruturas* no código que indicam **violação** dos **princípios** fundamentais de **design** e impactam negativamente a *qualidade* do projeto"



Code Smells **Clássicos**

1. Alternative Classes w/ Different Interfaces
2. Comments
3. Data Clumps
4. Divergent Change
5. Duplicated Code
6. Feature Envy
7. Inappropriate Intimacy
8. Incomplete Library Client
9. Large Class
10. Lazy Class
11. Long Method
12. Long Parameter List
13. Message Chains
14. Middle Man
15. Parallel Inheritance Hierarchies
16. Primitive Obsession
17. Refused Bequest
18. Shotgun Surgery
19. Speculative Generality
20. Switch Statements
21. Temporary Field

Code Smells **Clássicos**

1. Alternative Classes w/ Different Interfaces
2. Comments
3. Data Clumps
4. Divergent Change
5. Duplicated Code
6. Feature Envy
7. Inappropriate Intimacy
8. Incomplete Library Client
9. Large Class
10. Lazy Class
11. Long Method
12. Long Parameter List
13. Message Chains
14. Middle Man
15. Parallel Inheritance Hierarchies
16. Primitive Obsession
17. Refused Bequest
18. Shotgun Surgery
19. Speculative Generality
20. Switch Statements
21. Temporary Field

Code Smells **Clássicos**

Alternative Classes w/ Different Interfaces
Refused Bequest
Switch Statements
Temporary Field

Parallel Inheritance Hierarchies
Shotgun Surgery
Divergent Change

Speculative Generality
Duplicated Code
Lazy Class

Long Method
Long Parameter List
Data Clumps
Large Class
Primitive Obsession

Message Chains
Middle Man
Feature Envy
Inappropriate Intimacy

Code Smells **Clássicos - "Abusadores de OO"**

Alternative Classes w/ Different Interfaces

Refused Bequest

Switch Statements

Temporary Field

Code Smells **Clássicos - "Dispensáveis"**

Speculative Generality

Duplicated Code

Lazy Class

Code Smells **Clássicos - "Infladores"**

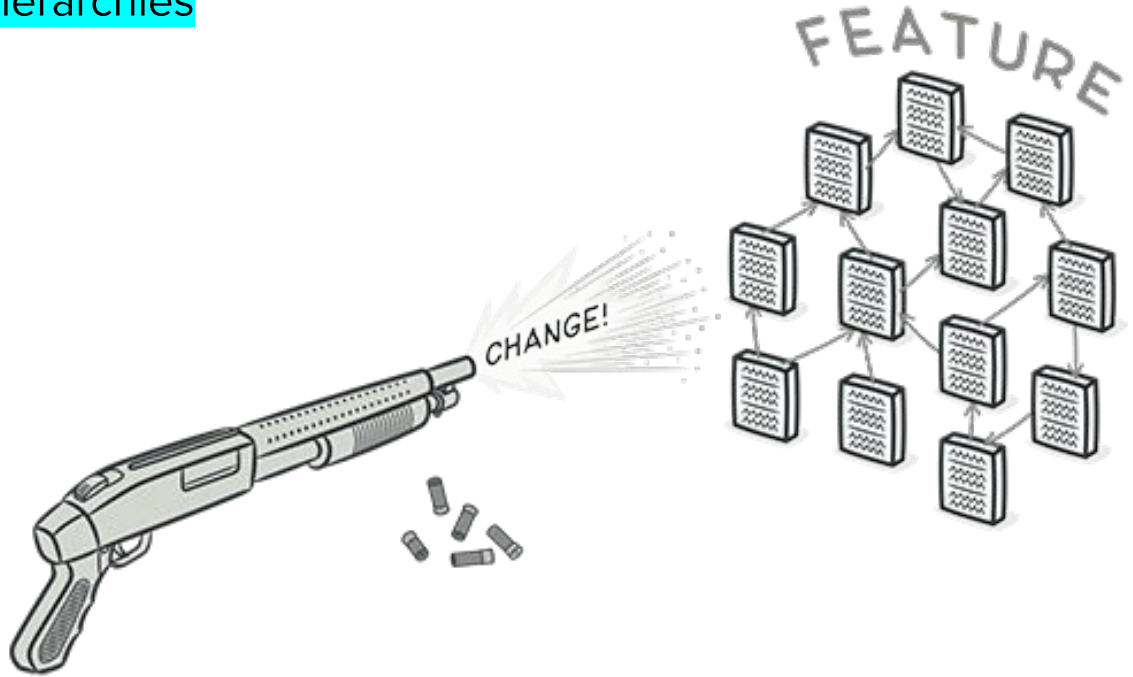
- Long Method
- Long Parameter List
- Data Clumps
- Large Class
- Primitive Obsession

Code Smells **Clássicos - "Barreiras de Mudança"**

Parallel Inheritance Hierarchies

Shotgun Surgery

Divergent Change



Code Smells **Clássicos - "Acopladores"**

Message Chains

Middle Man

Feature Envy

Inappropriate Intimacy

Exemplos

Exemplo - **"Refused Bequest"**

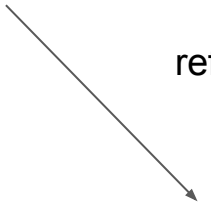
```
abstract class AnimalLegs

class Dog : AnimalLegs()

class Chair : AnimalLegs()
```

refatorando...

```
abstract class Legs

class Dog : Legs()

class Chair : Legs()
```

Exemplo - "Switch Statements"

```
fun configureGame(players: Int, game: Game?) {
        when {
            players == 0 -> {
                setEmptyGame()
            }
            players in 1..3 -> {
                if (game != null) {
                    this.gameSize = Game.SMALL
                    game.players.forEach {
                        configurePlayer(it, game, Game.SMALL)
                    }
                }
            }
            players in 4..10 -> {
                if (game != null) {
                    this.gameSize = Game.MEDIUM
                    game.players.forEach {
                        configurePlayer(it, game, Game.MEDIUM)
                    }
                }
            }
            else -> {
                if (game != null) {
                    this.gameSize = Game.BIG
                    game.players.forEach {
                        configurePlayer(it, game, Game.BIG)
                    }
                }
            }
        }
    }
```



refatorando...

```
fun configureGame(players: Int, game: Game?) {
    when {
        players == 0 -> setEmptyGame()
        players in 1..3 -> configureSmallGame(game)
        players in 4..10 -> configureMediumGame(game)
        else -> configureBigGame(game)
    }
}
```

Exemplo - **"Long Parameter List"**

```
fun savePerson(firstName: String, lastName: String, age: Int, occupation: String, document: String, address: String){
    ...
}
```

refatorando...

```
fun savePerson(person: Person) {
    ...
}
```

Exemplo - "Feature Envy"

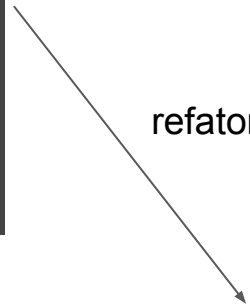
```
class Customer(private val phone: Phone, ...) {

    fun getPhone(): String {
        return "("+ phone.areaCode + ")"
        + phone.prefix + "-"
        + phone.number
    }

}
```

refatorando...

```
class Customer(private val phone: Phone, ...) {

    fun getPhone(): String = phone.formattedPhone()

}
```

Exemplo - **"Message Chains"**

```
class User {
    fun getConfiguration() = userConfig.getConfiguration()
}

class UserConfig {
    fun getConfiguration() = config.getConfiguration()
}

class Config {
    fun getConfiguration() = loadConfiguration()
}
```



refatorando...

```
class User {
    fun getConfiguration() = loadConfiguration()
}
```

Exemplo - **"Middle Man"**

```
class Request(private val requestConfig: RequestConfig) {

    fun getHeaders() = requestConfig.getHeaders()

    fun getTimeoutLimit() = requestConfig.getTimeoutLimit()

    fun getBody() = requestConfig.getBody()
}
```

refatorando...

```
class Request {

    fun getHeaders() = {...}

    fun getTimeoutLimit() = {...}

    fun getBody() = {...}
}
```

Exemplo - "Innapropriate Intimacy"

```
class Message {

    fun notifyUser(
        userName: String,
        userDetail: UserDetail,
        message: String
    ) {
        val messageBody = "Hi, " + userName + "\n" + message
                        + "("userDetail.streetName + ","
                        + userDetail.streetNumber
        this.notificationService.sendSMS(userDetail.number, messageBody)
    }
}
```

refatorando...

```
class Message {

    fun notifyUser(
        userName: String,
        number: String,
        address: String,
        message: String
    ) {
        val messageBody = "Hi, " + userName + "\n" + message + address
        this.notificationService.sendSMS(number, messageBody)
    }
}
```

Detecção de Code Smells no Kotlin

Detecção de Code Smells no **Kotlin**

https://github.com/arturbosch/detekt

Dica: Aplicar os princípios do SOLID pode evitar a inserção de alguns code smells em seu código, assim como o uso correto (não *over*) de *design patterns*.

Obrigado!

@jeziellago

#sejaumzupper

Links

https://www.industriallogic.com/wp-content/uploads/2005/09/smellstorefactorings.pdf

https://refactoring.guru/smells/

https://codeburst.io/write-clean-code-and-get-rid-of-code-smells-aea271f30318

https://github.com/arturbosch/detekt