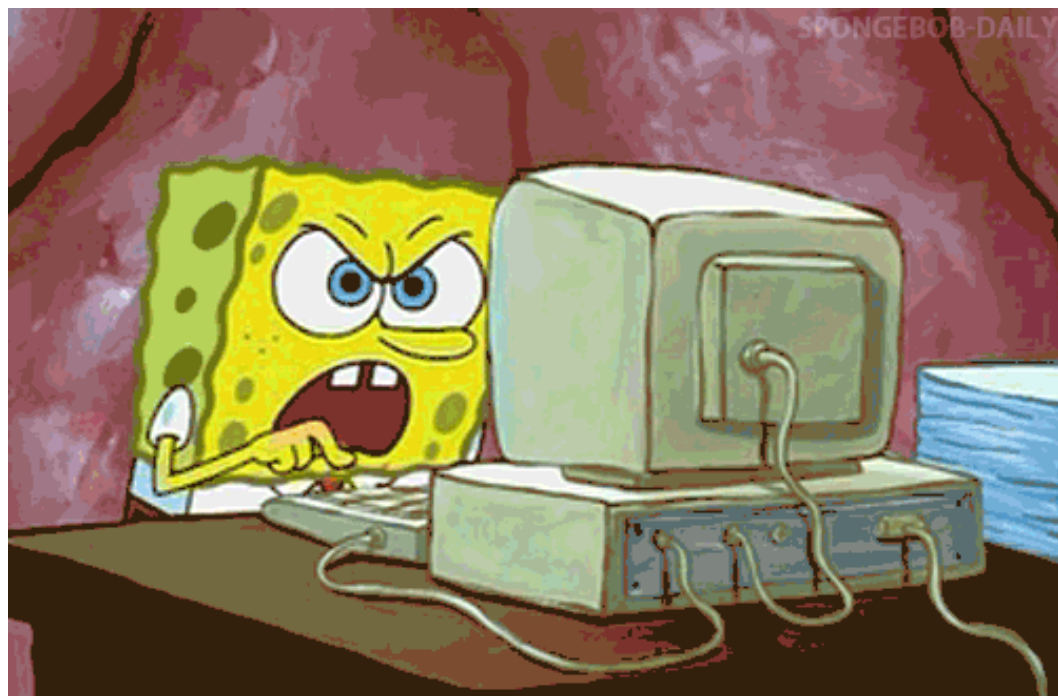


Bad Smells



O que são?

Solução

- ▶ Cada bad smells tem uma solução apropriada
- ▶ Refactoring

Exemplos de bad smells

- ← Duplicação de código
- ← Método deus/método longo

Duplicação de Código

Duplicação de código

"Se você vê as mesmas estruturas de código em mais de um lugar, você pode ter certeza que vai ser melhor se você encontrar uma maneira de unificar isso." - Martin Fowler

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Extract method

Extract method

- criar um novo método
- copiar o código extraído
- procurar por referências a variáveis locais no escopo do método
 - definir o que vai se tornar parâmetro e o que vai ser declarado como variável local temporária no escopo do novo método
- invocar este método no lugar dos trechos com repetição de código
- compile e teste garantindo que o código refatorado tenha o mesmo comportamento do código com bad smells.


```
#include <iostream>

using namespace std;

int main(int argc, char const *argv[]) {
    int arrayA[10], arrayB[15], sum_a, sum_b;

    for(int i = 0; i < 10; i++) {
        cin >> arrayA[i];
    }

    for(int i = 0; i < 15; i++) {
        cin >> arrayB[i];
    }

    sum_a = 0;
    for(int i = 0; i < 10; i++) {
        sum_a += arrayA[i];
    }
    cout << "The sum of elements in arrayA are " << sum_a << endl;

    sum_b = 0;
    for(int i = 0; i < 15; i++) {
        sum_b += arrayB[i];
    }
    cout << "The sum of elements in arrayB are " << sum_b << endl;

    return 0;
}
```

Extract method

```
#include <iostream>

using namespace std;

int main(int argc, char const *argv[]) {
    int arrayA[10], arrayB[15], sum_a, sum_b;

    for(int i = 0; i < 10; i++) {
        cin >> arrayA[i];
    }

    for(int i = 0; i < 15; i++) {
        cin >> arrayB[i];
    }

    sum_a = 0;
    for(int i = 0; i < 10; i++) {
        sum_a += arrayA[i];
    }
    cout << "The sum of elements in arrayA are " << sum_a << endl;

    sum_b = 0;
    for(int i = 0; i < 15; i++) {
        sum_b += arrayB[i];
    }
    cout << "The sum of elements in arrayB are " << sum_b << endl;

    return 0;
}
```

Extract method

```
#include <iostream>

using namespace std;

void read_array(int *array, int size_arr) {
    for(int i = 0; i < size_arr; i++) {
        cin >> array[i];
    }
}

int main(int argc, char const *argv[]) {
    int arrayA[10], arrayB[15], sum_a, sum_b;

    read_array(arrayA, 10);
    read_array(arrayB, 15);

    sum_a = 0;
    for(int i = 0; i < 10; i++) {
        sum_a += arrayA[i];
    }
    cout << "The sum of elements in arrayA are " << sum_a << endl;

    sum_b = 0;
    for(int i = 0; i < 15; i++) {
        sum_b += arrayB[i];
    }
    cout << "The sum of elements in arrayB are " << sum_b << endl;

    return 0;
}
```

Extract method

```
#include <iostream>

using namespace std;

void read_array(int *array, int size_arr) {
    for(int i = 0; i < size_arr; i++) {
        cin >> array[i];
    }
}

int main(int argc, char const *argv[]) {
    int arrayA[10], arrayB[15], sum_a, sum_b;

    read_array(arrayA, 10);
    read_array(arrayB, 15);
```

```
    sum_a = 0;
    for(int i = 0; i < 10; i++) {
        sum_a += arrayA[i];
    }
    cout << "The sum of elements in arrayA are " << sum_a << endl;

    sum_b = 0;
    for(int i = 0; i < 15; i++) {
        sum_b += arrayB[i];
    }
    cout << "The sum of elements in arrayB are " << sum_b << endl;
```

```
    return 0;
```

```
}
```

Extract method

```
#include <iostream>

using namespace std;

void read_array(int *array, int size_arr) {
    for(int i = 0; i < size_arr; i++) {
        cin >> array[i];
    }
}

int sum_elements_array(int *array, int size_arr) {
    int sum_array = 0;
    for(int i = 0; i < size_arr; i++) {
        sum_array += array[i];
    }
    return sum_array;
}

int main(int argc, char const *argv[]) {
    int arrayA[10], arrayB[15], sum;

    read_array(arrayA, 10);
    read_array(arrayB, 15);
    sum = sum_elements_array(arrayA, 10);
    cout << "The sum of elements in arrayA are " << sum << endl;

    sum = sum_elements_array(arrayB, 15);
    cout << "The sum of elements in arrayB are " << sum << endl;

    return 0;
}
```

Extract method

Pull Up method

Pull Up method

- Encontrar subclasses com métodos idênticos
- Verifique se a assinatura dos métodos são iguais, senão for, altere a assinatura para que ambos os métodos sejam atendidos com a chamada da superclasse
- crie um novo método na superclasse com o corpo de um dos métodos
- delete o método de uma das subclasses
- compile e teste
- delete o método restante
- compile e teste

Pull Up Method

```
public class Pessoa {  
    private String nome, usuario, senha;  
    private Integer codigo;  
  
    public Pessoa(String nome, String usuario, String senha, Integer codigo) {  
        this.nome = nome;  
        this.usuario = usuario;  
        this.senha = senha;  
        this.codigo = codigo;  
    }  
  
    public String getNome() {  
        return this.nome;  
    }  
  
    public String getUsuario() {  
        return this.usuario;  
    }  
  
    public String getSenha() {  
        return this.senha;  
    }  
  
    public Integer getCodigo() {  
        return this.codigo;  
    }  
}
```



```
public class Funcionario extends Pessoa {  
    private String endereco;  
    private Double salario, totalVendas;  
  
    public Funcionario(String endereco, Double salario, Double totalVendas,  
String nome, String usuario, String senha, Integer codigo) {  
        super(nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.salario = salario;  
        this.totalVendas = totalVendas;  
    }  
  
    public String getEndereco() {  
        return this.endereco;  
    }  
  
    public Double getSalario() {  
        return this.salario;  
    }  
  
    public Double getTotalVendas() {  
        return this.totalVendas;  
    }  
}
```

```
public class Cliente extends Pessoa {  
    private String endereco;  
    private Double totalCompras;  
  
    public Cliente(String endereco, Double totalCompras, String nome,  
String usuario, String senha, Integer codigo) {  
        super(nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.totalCompras = totalCompras;  
    }  
  
    public String getEndereco() {  
        return this.endereco;  
    }  
  
    public Double getTotalCompras() {  
        return this.totalCompras;  
    }  
}
```

Pull Up Method

```
public class Funcionario extends Pessoa {  
    private String endereco;  
    private Double salario, totalVendas;  
  
    public Funcionario(String endereco, Double salario, Double totalVendas,  
        String nome, String usuario, String senha, Integer codigo) {  
        super(nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.salario = salario;  
        this.totalVendas = totalVendas;  
    }  
  
    public String getEndereco() {  
        return this.endereco;  
    }  
  
    public Double getSalario() {  
        return this.salario;  
    }  
  
    public Double getTotalVendas() {  
        return this.totalVendas;  
    }  
}
```

```
public class Cliente extends Pessoa {  
    private String endereco;  
    private Double totalCompras;  
  
    public Cliente(String endereco, Double totalCompras, String nome,  
        String usuario, String senha, Integer codigo) {  
        super(nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.totalCompras = totalCompras;  
    }  
  
    public String getEndereco() {  
        return this.endereco;  
    }  
  
    public Double getTotalCompras() {  
        return this.totalCompras;  
    }  
}
```

Pull Up Method

Pull Up Method

```
public class Pessoa {  
    private String nome, usuario, senha, endereco;  
    private Integer codigo;  
  
    public Pessoa(String endereco, String nome, String usuario, String senha, Integer codigo) {  
        this.endereco = endereco;  
        this.nome = nome;  
        this.usuario = usuario;  
        this.senha = senha;  
        this.codigo = codigo;  
    }  
  
    public String getEndereco() {  
        return this.endereco;  
    }  
  
    public String getNome() {  
        return this.nome;  
    }  
  
    public String getUsuario() {  
        return this.usuario;  
    }  
  
    public String getSenha() {  
        return this.senha;  
    }  
  
    public Integer getCodigo() {  
        return this.codigo;  
    }  
}
```

```
public class Funcionario extends Pessoa {  
    private Double salario, totalVendas;  
  
    public Funcionario(String endereco, Double salario, Double totalVendas,  
        String nome, String usuario, String senha, Integer codigo) {  
        super(endereco, nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.salario = salario;  
        this.totalVendas = totalVendas;  
    }  
  
    public Double getSalario() {  
        return this.salario;  
    }  
  
    public Double getTotalVendas() {  
        return this.totalVendas;  
    }  
}
```

```
public class Cliente extends Pessoa {  
    private Double totalCompras;  
  
    public Cliente(String endereco, Double totalCompras, String nome,  
        String usuario, String senha, Integer codigo) {  
        super(endereco, nome, usuario, senha, codigo);  
        this.endereco = endereco;  
        this.totalCompras = totalCompras;  
    }  
  
    public Double getTotalCompras() {  
        return this.totalCompras;  
    }  
}
```

Pull Up Method

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Form template method

Form template method

- Decomponha os métodos para extrair tudo que for igual e tudo que for diferente
- Use o Pull Up Method para "puxar" os métodos idênticos para a superclasse
- Para os diferentes usar o Rename Method
- Compile e teste
- Use o Pull Up Method para os métodos originais
- Compile e teste
- Remova os outros métodos, compile e teste após cada remoção

Substitute Algorithm

Substitute Algorithm

- Verifique quais são os algoritmos que conseguem manter o comportamento
- Teste até ter certeza que não haverá mudança de comportamento
- Se o resultado dos testes for positivo, substitua o código

Substitute Algorithm

```
String foundPerson(String[] People) {  
    for(int i = 0; i < people.length; i++) {  
        if(people[i].equals("Don")) {  
            return "Don";  
        }  
        if(people[i].equals("John")) {  
            return "John";  
        }  
        if(people[i].equals("Bruce")) {  
            return "Bruce";  
        }  
    }  
    return "";  
}
```

Substitute Algorithm

```
String foundPerson(String[] People) {  
    for(int i = 0; i < people.length; i++) {  
        if(people[i].equals("Don")) {  
            return "Don";  
        }  
        if(people[i].equals("John")) {  
            return "John";  
        }  
        if(people[i].equals("Bruce")) {  
            return "Bruce";  
        }  
    }  
    return "";  
}
```

Substitute Algorithm

```
String foundPerson(String[] People) {  
    List candidates = Arrays.asList(new String[]{"Don", "John", "Bruce"});  
    for(int i = 0; i < people.length; i++) {  
        if(candidates.contains(people[i]) ) {  
            return people[i];  
        }  
    }  
    return "";  
}
```

Extract Class

Extract class

- Defina quais vão ser as responsabilidades de cada classe
- Crie uma nova classe para conter as responsabilidades definidas no passo anterior
- Faça uma conexão entre as classes
- Mova os campos para a nova classe
- Mova os métodos relacionados a estes campos
- Revise a divisão feita e reduza a interfaces
- Decida se vai ou não expor as novas classes

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern, layered effect. The text is centered in a clean, sans-serif font.

Long method
/ God method

Replace Temp with Query

- Veja quais são as variáveis temporárias
- Declare a variável temporária como final
- Compile
- Remova a expressão e coloque no retorno do método
- Compile e teste

Introduce Parameter Object

- Crie uma nova classe para representar o grupo de parâmetros
- Compile
- Altere a assinatura do método para que, invés do grupo, ele receba o novo Objeto que contém os antigos atributos
- Modifique as assinaturas do grupo de parâmetros para o novo objeto
- Compile e teste

Preserve Whole Object

- Crie um novo parâmetro para todos os objetos dos dados
- Compile e teste
- Determine quais parâmetros você vai obter deste objeto
- Mude a assinatura do método para que invés de receber vários parâmetros receba apenas a instancia do objeto


Replace method with method object

- ▶ Criar um novo objeto para cuidar dos vários comportamentos de um método muito longo

Decompose Conditional

- Para um condicional muito grande e complicado, prefira criar um método que faça a verificação e das partes

```
if(date.before(SUMMER_START) || date.after(SUMMER_END)) {  
    change = quantity * _winterRate + winterServiceCharge;  
} else {  
    change = quantity * _summerRate;  
}  
  
if(notSummer(date)) {  
    change = winterCharge(quantity);  
} else {  
    change = summerCharge(quantity);  
}
```



The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Obrigado!!