

Guerreiros do Código

Dominando os Padrões de Projeto
Criacionais em Delphi



JULIO CESAR DE ANDRADE

01

PADRÃO DE PROJETO

SINGLETON

Singleton

Garantindo Uma Única Instância

O Singleton garante que uma classe tenha apenas uma instância e fornece um ponto global de acesso.

Exemplo: Gerenciador de Configurações

```
ConfigManager.pas

1  unit ConfigManager;
2
3  interface
4
5  uses
6      System.SysUtils;
7
8  type
9      TConfigManager = class
10     private
11         class var FInstance: TConfigManager;
12         constructor Create;
13     public
14         class function GetInstance: TConfigManager;
15     end;
16
17 implementation
18
19 constructor TConfigManager.Create;
20 begin
21     inherited;
22 end;
23
```

Singleton

```
ConfigManager.pas

24 class function TConfigManager.GetInstance: TConfigManager;
25 begin
26     if not Assigned(FInstance) then
27         FInstance := TConfigManager.Create;
28     Result := FInstance;
29 end;
30
31 end.
```

Uso:

```
ConfigManager.pas

1 var
2     Config: TConfigManager;
3 begin
4     Config := TConfigManager.GetInstance;
5 end.
```

02

PADRÃO DE PROJETO

**FACTORY
METHOD**

Factory Method

Criando Objetos com Flexibilidade

Esse padrão delega a responsabilidade da criação de objetos para subclasses.

Exemplo: Gerador de Relatórios

```
ReportFactory.pas

1  unit ReportFactory;
2
3  interface
4
5  type
6      IReport = interface
7          procedure Generate;
8      end;
9
10     TPDFReport = class(TInterfacedObject, IReport)
11         procedure Generate;
12     end;
13
14     THTMLReport = class(TInterfacedObject, IReport)
15         procedure Generate;
16     end;
17
18     TReportFactory = class
19         class function CreateReport(ReportType: string): IReport;
20     end;
21
22 implementation
23
24 procedure TPDFReport.Generate;
25 begin
26     Writeln('Gerando relatório PDF...');
27 end;
28
29 procedure THTMLReport.Generate;
30 begin
31     Writeln('Gerando relatório HTML...');
32 end;
33
```

Factory Method

```
ReportFactory.pas

34 class function TReportFactory.CreateReport(ReportType: string): IReport;
35 begin
36     if ReportType = 'PDF' then
37         Result := TPDFReport.Create
38     else if ReportType = 'HTML' then
39         Result := THTMLReport.Create
40     else
41         raise Exception.Create('Tipo de relatório inválido');
42 end;
43
44 end.
```

Uso:

```
ReportFactory.pas

1 var
2     Report: IReport;
3 begin
4     Report := TReportFactory.CreateReport('PDF');
5     Report.Generate;
6 end.
```

03

PADRÃO DE PROJETO

**ABSTRACT
FACTORY**

Abstract Factory

Criando Famílias de Objetos Relacionados

O Abstract Factory permite criar objetos relacionados sem especificar suas classes concretas.

Exemplo: UI para Diferentes Sistemas Operacionais

```
UIFactory.pas

1  unit UIFactory;
2
3  interface
4
5  type
6      IButton = interface
7          procedure Render;
8      end;
9
10     IGUIFactory = interface
11         function CreateButton: IButton;
12     end;
13
14     TWindowsButton = class(TInterfacedObject, IButton)
15         procedure Render;
16     end;
17
18     TMacButton = class(TInterfacedObject, IButton)
19         procedure Render;
20     end;
21
22     TWindowsFactory = class(TInterfacedObject, IGUIFactory)
23         function CreateButton: IButton;
24     end;
25
26     TMacFactory = class(TInterfacedObject, IGUIFactory)
27         function CreateButton: IButton;
28     end;
29
30 implementation
31
```

Abstract Factory

```
UIFactory.pas

32 procedure TWindowsButton.Render;
33 begin
34     WriteLn('Renderizando botão do Windows');
35 end;
36
37 procedure TMacButton.Render;
38 begin
39     WriteLn('Renderizando botão do Mac');
40 end;
41
42 function TWindowsFactory.CreateButton: IButton;
43 begin
44     Result := TWindowsButton.Create;
45 end;
46
47 function TMacFactory.CreateButton: IButton;
48 begin
49     Result := TMacButton.Create;
50 end;
51
52 end.
```

Uso:

```
UIFactory.pas

1 var
2     Factory: IGUIFactory;
3     Button: IButton;
4 begin
5     Factory := TWindowsFactory.Create;
6     Button := Factory.CreateButton;
7     Button.Render;
8 end.
```

04

PADRÃO DE PROJETO

BUILDER

Builder

Construindo Objetos Passo a Passo

O Builder é usado para criar objetos complexos separando sua construção de sua representação.

Exemplo: Montagem de um Computador

```
PCBuilder.pas

1  unit PCBuilder;
2
3  interface
4
5  type
6      TComputer = class
7          CPU, GPU: string;
8          procedure ShowSpecs;
9      end;
10
11     IPCBuilder = interface
12         procedure SetCPU;
13         procedure SetGPU;
14         function GetComputer: TComputer;
15     end;
16
17     TGamerPCBuilder = class(TInterfacedObject, IPCBuilder)
18     private
19         FComputer: TComputer;
20     public
21         constructor Create;
22         procedure SetCPU;
23         procedure SetGPU;
24         function GetComputer: TComputer;
25     end;
26
27 implementation
28
29 constructor TGamerPCBuilder.Create;
30 begin
31     FComputer := TComputer.Create;
32 end;
33
```

Builder

```
PCBuilder.pas

34 procedure TGamerPCBuilder.SetCPU;
35 begin
36     FComputer.CPU := 'Intel i9';
37 end;
38
39 procedure TGamerPCBuilder.SetGPU;
40 begin
41     FComputer.GPU := 'RTX 4090';
42 end;
43
44 function TGamerPCBuilder.GetComputer: TComputer;
45 begin
46     Result := FComputer;
47 end;
48
49 procedure TComputer.ShowSpecs;
50 begin
51     Writeln('CPU: ', CPU, ' | GPU: ', GPU);
52 end;
53
54 end.
```

Uso:

```
PCBuilder.pas

1  var
2      Builder: IPCBuilder;
3      PC: TComputer;
4  begin
5      Builder := TGamerPCBuilder.Create;
6      Builder.SetCPU;
7      Builder.SetGPU;
8      PC := Builder.GetComputer;
9      PC.ShowSpecs;
10 end.
```

05

PADRÃO DE PROJETO

PROTOTYPE

Prototype

Clonando Objetos

O Prototype permite criar novos objetos copiando um existente.

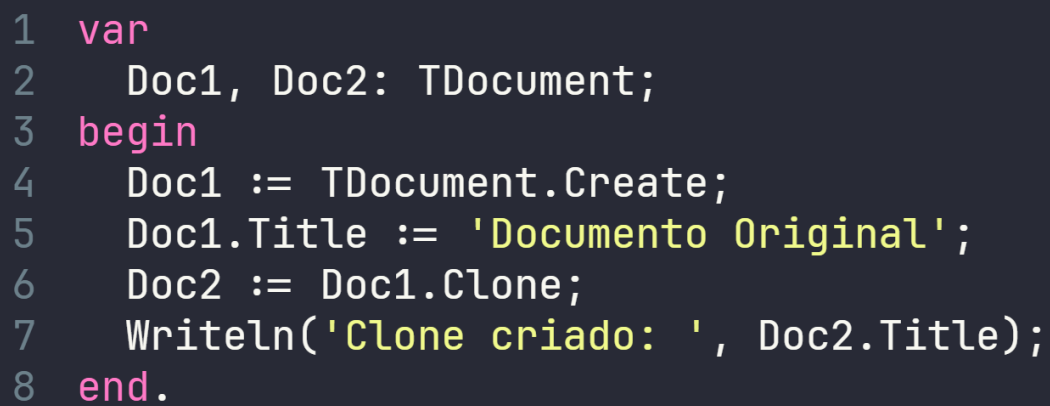
Exemplo: Clonagem de Documentos

```
DocumentPrototype.pas

1  unit DocumentPrototype;
2
3  interface
4
5  type
6      TDocument = class
7      public
8          Title: string;
9          function Clone: TDocument;
10     end;
11
12 implementation
13
14 function TDocument.Clone: TDocument;
15 begin
16     Result := TDocument.Create;
17     Result.Title := Self.Title;
18 end;
19
20 end.
```

Builder

Uso:



```
1  var
2    Doc1, Doc2: TDocument;
3  begin
4    Doc1 := TDocument.Create;
5    Doc1.Title := 'Documento Original';
6    Doc2 := Doc1.Clone;
7    Writeln('Clone criado: ', Doc2.Title);
8  end.
```

AGRADECIMENTOS

OBRIGADO POR LER ATÉ AQUI

Esse Ebook foi gerado por IA e diagramado por humano.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizada uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.



<https://github.com/juliocandrade/prompts-recipe-to-create-a-ebook>

