

Builder

Padrões de Projeto Criacional I

Prof. Me Jefferson Passerini

O padrão **Builder** separa a construção de um objeto de sua representação de modo que o mesmo processo de construção de um objeto possa criar diferentes representações.

Aplicabilidade (Quando utilizar?)

- Quando um algoritmo que cria um objeto complexo deve ser independente das partes que o compõem e de como tais partes são montadas.
- Quando o processo de construção deve permitir representações diferentes para o objeto que é construído.

Componentes

- **Builder** → especifica uma interface abstrata para a criação de partes de um objeto (produto), tal interface deverá ser seguida por todos os BuildersConcretos.

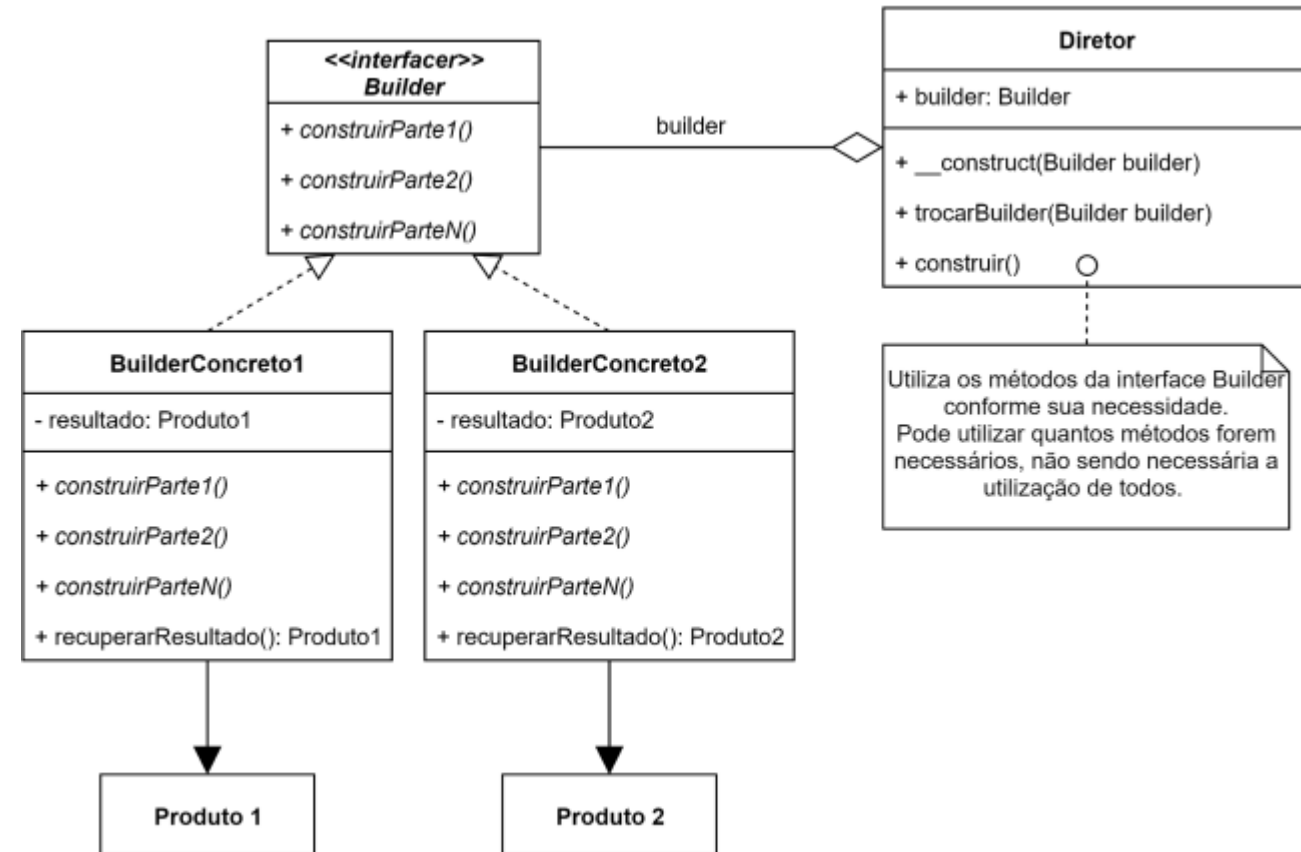
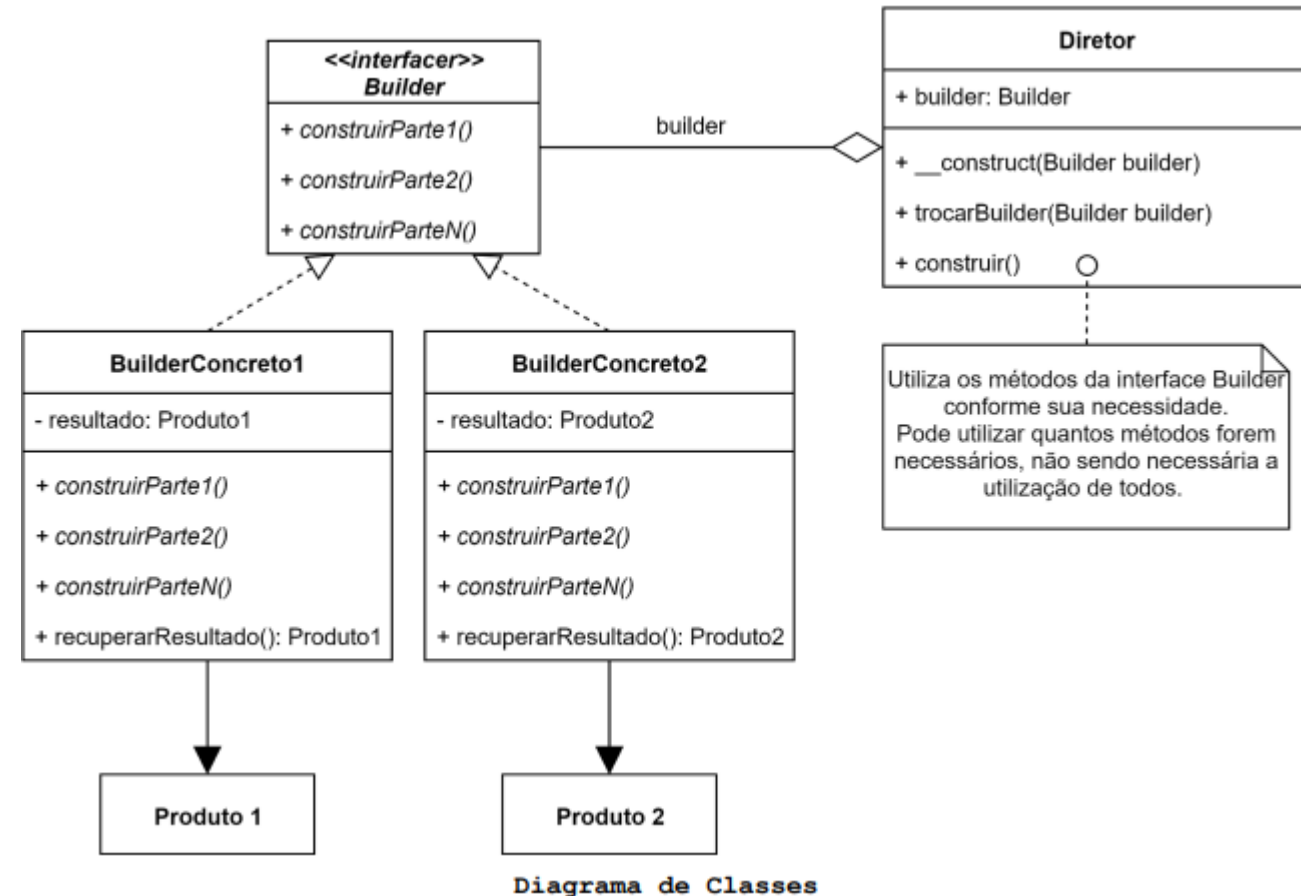


Diagrama de Classes

Componentes

- **BuilderConcreto:**

- Constrói e monta partes do Produto utilizando a implementação concreta da interface Builder a qual implementa.
- Define e mantém a representação (Produto) que cria.
- Fornece uma interface (método) para a recuperação do produto criado.



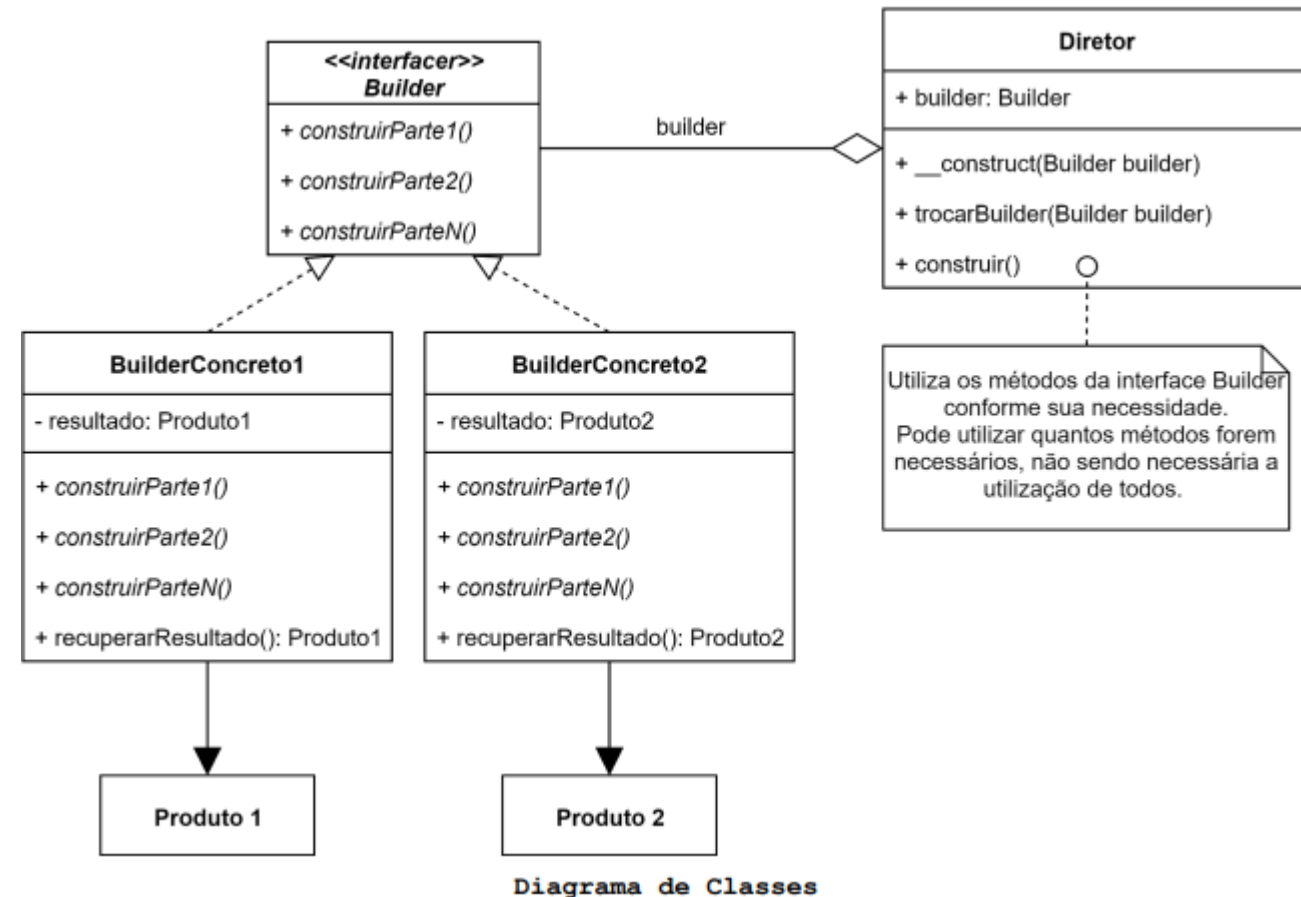
Componentes

- **Diretor:**

- Constrói um objeto conforme suas necessidades. Para isso ele utiliza a interface de Builder.

- **Produto:**

- Representa o objeto complexo em construção. BuilderConcreto constrói a representação interna do produto e define o processo pelo qual ele será criado.
- Produtos são objetos resultantes. Produtos construídos por diferentes BuildersConcretos não precisam pertencer a mesma interface ou hierarquia de classes.



Motivação (Por que utilizar?)

- Durante o processo de desenvolvimento de um software podemos nos deparar com objetos muito complexos de se construir, ou então um objeto que pode ser construído de muitas formas diferentes.
- Observe a classe ao lado, é possível notar a grande quantidade de atributos existentes.
- Existem infinitas combinações de valores que podem ser assumidos por eles, o que acarretaria na criação de objetos diferentes.

GeradorPDF

- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeigh
- bool hasFooter
- int footerHeigh
- string pageColor
- string encode

```
public function __construct (
    string pageOrientation,
    string unit,
    int PageSizeX,
    int PageSizeY,
    int maginTop,
    int maginRight,
    int maginBottom,
    int maginLeft,
    bool hasHeader,
    int headerHeigh,
    bool hasFooter,
    int footerHeigh,
    string pageColor,
    string encode
);
```

```
//Métodos Getters;
```

```
//Métodos Setters;
```

```
- __toString(): String
```

Motivação (Por que utilizar?)

- Não se trata de uma classe complexa, a classe GeradorPDF possui 14 atributos, todos são inicializados com algum valor e são redefinidos no construtor da classe.
- Isso implica que tal construtor precisa receber 14 parâmetros para criar um objeto da classe GeradorPDF.
- A classe pode não ser complexa, porém, a instanciação de seus objetos é trabalhosa e muito extensa.

GeradorPDF

- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeigh
- bool hasFooter
- int footerHeigh
- string pageColor
- string encode

```
public function __construct (
    string pageOrientation,
    string unit,
    int PageSizeX,
    int PageSizeY,
    int maginTop,
    int maginRight,
    int maginBottom,
    int maginLeft,
    bool hasHeader,
    int headerHeigh,
    bool hasFooter,
    int footerHeigh,
    string pageColor,
    string encode
);
```

```
//Métodos Getters;
```

```
//Métodos Setters;
```

```
- __toString(): String
```


Motivação (Por que utilizar?)

- Além dos atributos e método construtor, a classe também possui métodos Getter's e Setter's, e um método toString() que formata o objeto antes de sua impressão.
- Deste modo, existem nessa classe 28 métodos (Getter's e Setter's) mais o construtor.
- Você pode observar o código no projeto problema fornecido com este material.

GeradorPDF

- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeigh
- bool hasFooter
- int footerHeigh
- string pageColor
- string encode

```
public function __construct (
    string pageOrientation,
    string unit,
    int PageSizeX,
    int PageSizeY,
    int maginTop,
    int maginRight,
    int maginBottom,
    int maginLeft,
    bool hasHeader,
    int headerHeigh,
    bool hasFooter,
    int footerHeigh,
    string pageColor,
    string encode
);
```

```
//Métodos Getters;
```

```
//Métodos Setters;
```

```
- __toString(): String
```

GeradorPDF

- string pageOrientation

- string unit

- int PageSizeX

- int PageSizeY

- int marginTop

- int marginRight

- int marginBottom

- int marginLeft

- bool hasHeader

- int headerHeigh

- bool hasFooter

- int footerHeigh

- string pageColor

- string encode

public function __construct (

string pageOrientation,

string unit,

int PageSizeX,

int PageSizeY,

int maginTop,

int maginRight,

int maginBottom,

int maginLeft,

bool hasHeader,

int headerHeigh,

bool hasFooter,

int footerHeigh,

string pageColor,

string encode

);

//Métodos Getters;

//Métodos Setters;

- __toString(): String

Classe GeradorPDF

Motivação (Por que utilizar?)

- O problema em questão não está na classe GeradorPDF mas na instanciação de seus objetos.
- Repare em quantos parâmetros são necessários para criar um objeto PDF A4 e outra para PDF A3.

Atributos	A4	A3
pageOrientation	portrait	portrait
unit	mm	mm
PageSizeX	210	297
PageSizeY	297	420
marginTop	30	60
marginRight	20	40
marginBottom	30	60
marginLeft	20	40
hasHeader	true	false
headerHeigh	15	0
hasFooter	true	false
footerHeigh	15	0
pageColor	#ffffff	#ffffff
encode	UTF-8	UTF-8

Especificações do PDF A4 e A3

Motivação (Por que utilizar?)

- É necessário muita informação para criação de 2 objetos.
- O Cliente (quem cria objetos da classe GeradorPDF) é o responsável por conhecer os valores necessários para configuração e criação dos objetos.
- Além disso, o Cliente ainda pode ter mais responsabilidades o que violaria o princípio da responsabilidade única.

//Problema

```
GeradorPDF newPDFA4 = new GeradorPDF("portrait", "mm", 210, 297, 30, 20, 30, 20, true, 15, true, 15, "#ffffff", "UTF-8");

Console.WriteLine(newPDFA4.ToString());

GeradorPDF newPDFA3 = new GeradorPDF("portrait", "mm", 297, 420, 30, 20, 30, 20, true, 15, true, 15, "#ffffff", "UTF-8");

Console.WriteLine(newPDFA3.ToString());
```

GeradorPDF

- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeigh
- bool hasFooter
- int footerHeigh
- string pageColor
- string encode

```
public function __construct (
    string pageOrientation,
    string unit,
    int PageSizeX,
    int PageSizeY,
    int maginTop,
    int maginRight,
    int maginBottom,
    int maginLeft,
    bool hasHeader,
    int headerHeigh,
    bool hasFooter,
    int footerHeigh,
    string pageColor,
    string encode
);

//Métodos Getters;

//Métodos Setters;

- __toString(): String
```

Motivação (Por que utilizar?)

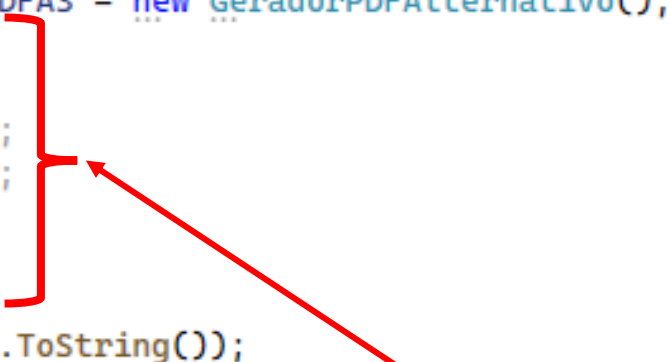
- Uma solução possível seria **eliminar o construtor da classe GeradorPDF** e como os atributos possuem valores “padrão” na sua criação seriam gerados com esses valores.
- E no Cliente só passaríamos os atributos que tem que serem alterados.

```
//Solução alternativa - removendo o construtor.
GeradorPDFAlternativo PDF4 = new GeradorPDFAlternativo();
PDF4.hasFooter = true;
PDF4.hasHeader = true;
PDF4.footerHeight = 15;
PDF4.headerHeight = 15;

Console.WriteLine(PDF4.ToString());

GeradorPDFAlternativo PDF3 = new GeradorPDFAlternativo();
PDF3.hasFooter = true;
PDF3.hasHeader = true;
PDF3.footerHeight = 15;
PDF3.headerHeight = 15;
PDF3.pageSizeX = 297;
PDF3.pageSizeY = 420;

Console.WriteLine(PDF3.ToString());
```



GeradorPDF

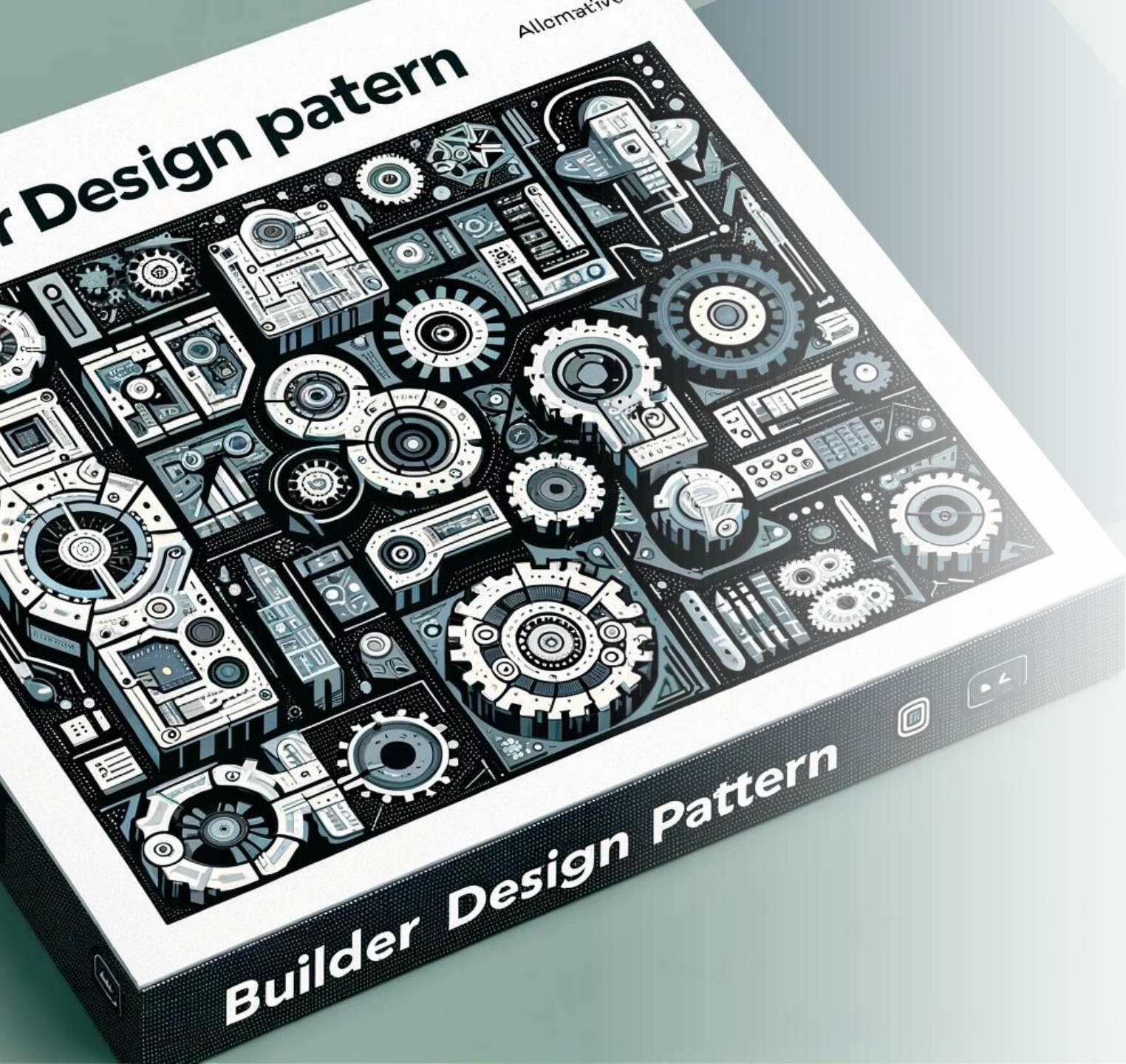
- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeight
- bool hasFooter
- int footerHeight
- string pageColor
- string encode

```
public function __construct (
    string pageOrientation,
    string unit,
    int PageSizeX,
    int PageSizeY,
    int marginTop,
    int marginRight,
    int marginBottom,
    int marginLeft,
    bool hasHeader,
    int headerHeight,
    bool hasFooter,
    int footerHeight,
    string pageColor,
    string encode
);
```

```
//Métodos Getters;

//Métodos Setters;

- __toString(): String
```

Builder

Implementação C#

Padrões de Projeto Criacional I

Prof. Me Jefferson Passerini

- string pageOrientation
- string unit
- int PageSizeX
- int PageSizeY
- int marginTop
- int marginRight
- int marginBottom
- int marginLeft
- bool hasHeader
- int headerHeigh
- bool hasFooter
- int footerHeigh
- string pageColor
- string encode

```
public function __construct (  
    string pageOrientation,  
    string unit,  
    int PageSizeX,  
    int PageSizeY,  
    int maginTop,  
    int maginRight,  
    int maginBottom,  
    int maginLeft,  
    bool hasHeader,  
    int headerHeigh,  
    bool hasFooter,  
    int footerHeigh,  
    string pageColor,  
    string encode  
);
```

```
//Métodos Getters;
```

```
//Métodos Setters;
```

```
- __toString(): String
```

- Estamos criando dois objetos diferentes a partir da mesma classe.
- A definição diz: ***“o padrão Builder separa a construção de um objeto complexo de sua representação”***.
- Então temos que criar duas classes que têm como responsabilidade apenas a construção de objetos da classe GeradorPDF.
- Continuando com o método construtor fora da classe GeradorPDF, iremos criar as classes **BuilderA4** e **BuilderA3** que irão criar objetos a partir da especificação de A4 e A3 respectivamente.

- A definição diz: ***“o padrão Builder separa a construção de um objeto complexo de sua representação”***.
- Assim o GeradorPDF é a representação do produto não é mais responsável pela sua criação e sim as classes BuilderA4 e BuilderA3

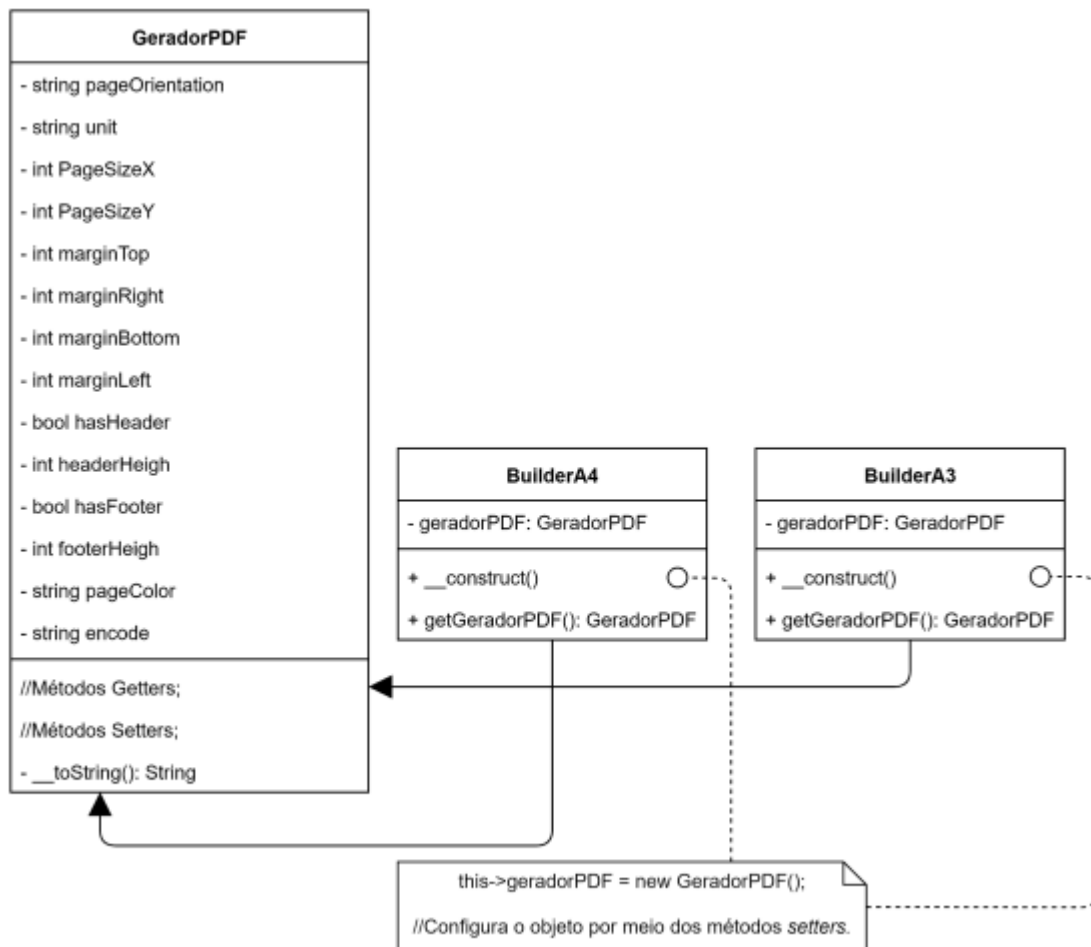
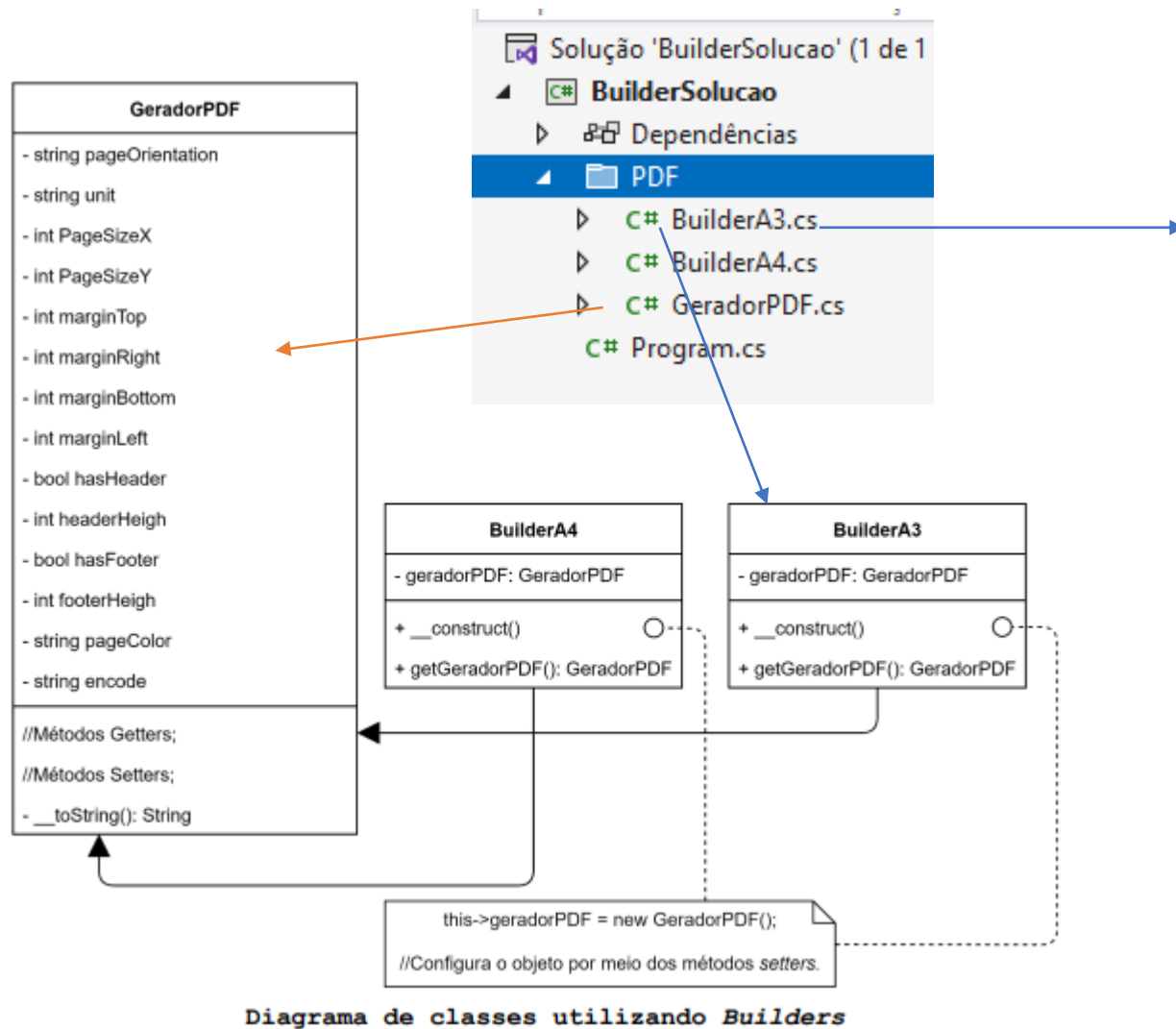


Diagrama de classes utilizando Builders



```

1  using BuilderSolucao.PDF;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace BuilderSolucao.PDF
9  {
10     3 referências
11     public class BuilderA3 {
12         private GeradorPDF geradorPDF;
13
14         1 referência
15         public BuilderA3() {
16             this.geradorPDF = new GeradorPDF();
17             this.geradorPDF.pageOrientation = "portrait";
18             this.geradorPDF.unit = "mm";
19             this.geradorPDF.pageSizeX = 297;
20             this.geradorPDF.pageSizeY = 420;
21             this.geradorPDF.marginTop = 30;
22             this.geradorPDF.marginRight = 20;
23             this.geradorPDF.marginBottom = 30;
24             this.geradorPDF.marginLeft = 20;
25             this.geradorPDF.hasHeader = true;
26             this.geradorPDF.headerHeight = 15;
27             this.geradorPDF.hasFooter = true;
28             this.geradorPDF.footerHeight = 15;
29             this.geradorPDF.pageColor = "#ffffff";
30             this.geradorPDF.encode = "UTF-8";
31         }
32
33         1 referência
34         public GeradorPDF getPDF() {
35             return this.geradorPDF;
36         }
37     }
38 }

```

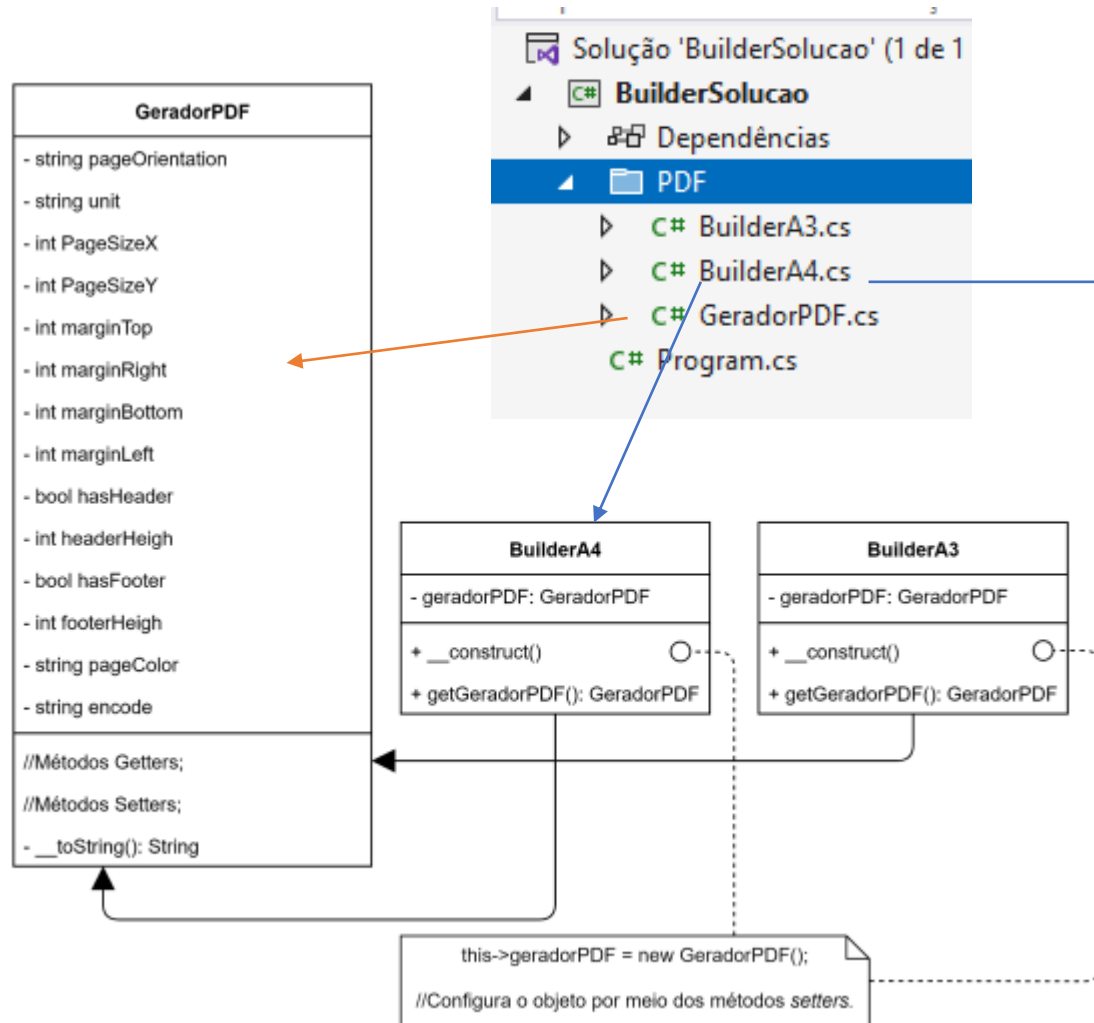
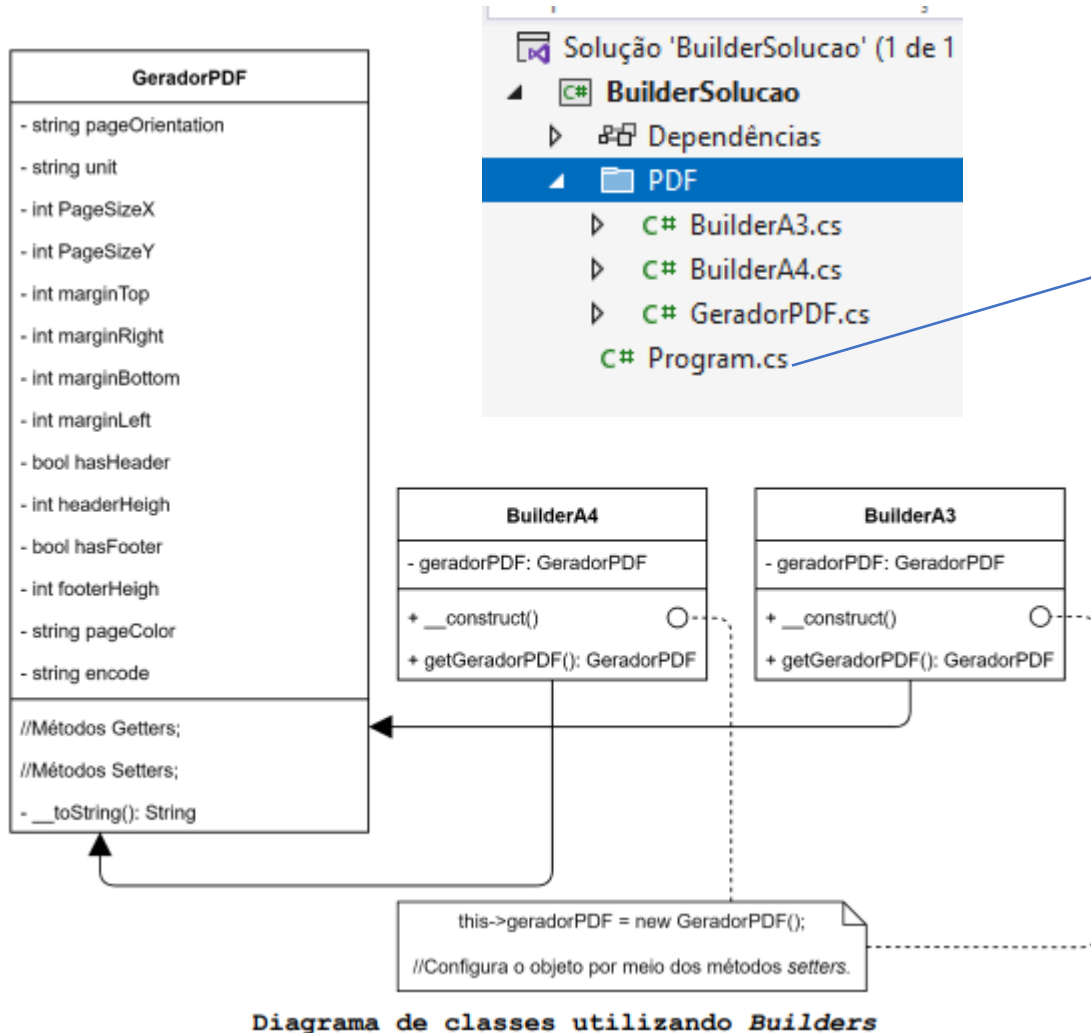


Diagrama de classes utilizando Builders

```

1  using BuilderSolucao.PDF;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace BuilderSolucao.PDF
9  {
10     3 referências
11     public class BuilderA4 {
12         private GeradorPDF geradorPDF;
13
14         1 referência
15         public BuilderA4() {
16             this.geradorPDF = new GeradorPDF();
17             this.geradorPDF.pageOrientation = "portrait";
18             this.geradorPDF.unit = "mm";
19             this.geradorPDF.pageSizeX = 210;
20             this.geradorPDF.pageSizeY = 297;
21             this.geradorPDF.marginTop = 30;
22             this.geradorPDF.marginRight = 20;
23             this.geradorPDF.marginBottom = 30;
24             this.geradorPDF.marginLeft = 20;
25             this.geradorPDF.hasHeader = true;
26             this.geradorPDF.headerHeight = 15;
27             this.geradorPDF.hasFooter = true;
28             this.geradorPDF.footerHeight = 15;
29             this.geradorPDF.pageColor = "#ffffff";
30             this.geradorPDF.encode = "UTF-8";
31         }
32
33         1 referência
34         public GeradorPDF getPDF()
35         {
36             return this.geradorPDF;
37         }
38     }
39 }
  
```



```

1  using BuilderSolucao.PDF;
2
3
4  BuilderA4 builderA4 = new BuilderA4();
5  GeradorPDF pdfA4 = builderA4.getPDF();
6  Console.WriteLine(pdfA4.ToString());
7
8  BuilderA3 builderA3 = new BuilderA3();
9  GeradorPDF pdfA3 = builderA3.getPDF();
10 Console.WriteLine(pdfA3.ToString());
11

```

- A definição diz: “**o padrão Builder separa a construção de um objeto complexo de sua representação**”.

```

PageOrientation = portrait
Unit = mm
PageSizeX = 210
PageSizeY = 297
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HasHeader = True
HeaderHeight = 15
HasFooter = True
FooterHeight = 15
PageColor = #ffffff
Encode = UTF-8

```

```

PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HasHeader = True
HeaderHeight = 15
HasFooter = True
FooterHeight = 15
PageColor = #ffffff
Encode = UTF-8

```

- Com o Builder a responsabilidade de criação e configuração dos objetos sai do **Cliente**.

- Devido aos Builders o código Cliente ficou mais simples.
- As classes BuilderA4 e BuilderA3 controlam o processo de criação de determinados objetos da classe de acordo com suas especificações.
- ***Porém o código ainda não está flexível: não podemos criar um PDF sem cabeçalho ou rodapé por exemplo.***
- Outra parte da definição do Builder diz: ***“um mesmo processo de construção de um objeto pode criar diferentes representações dele”***, em outras palavras um processo de criação pode criar objetos diferentes.

- Vamos dividir o processo de criação dos objetos em partes;
- Elas podem ou não ter uma ordem definida.
- Algumas partes podem ser facultativas e outras obrigatórias, isso varia conforme a necessidade do contexto do seu código.
- No nosso caso podemos quebrar as classes BuilderA4 e BuilderA3 nos seguintes métodos.

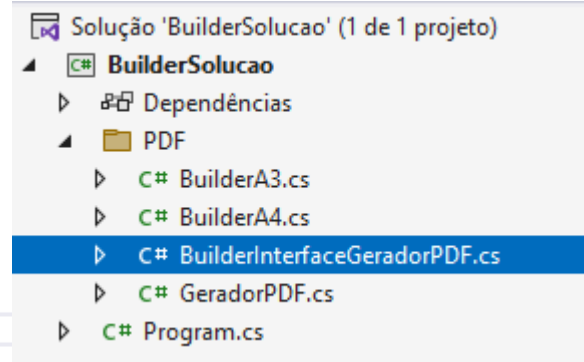


- `getGeradorPDF()`: Retorna o objeto já configurado.
- `setPageConfiguration()`: Define as configurações básicas das páginas do PDF.
- `setMargin()`: Configura as margens das páginas do PDF.
- `setHeader()`: Configura o cabeçalho das páginas do PDF.
- `setFooter()`: Configura o rodapé das páginas do PDF.

- Para garantir que todos os Builders tenham tais métodos, ***vamos criar a interface BuilderInterfaceGeradorPDF*** a qual todos eles devem implementar.
- Essa interface dita o processo de criação que todos os Builders devem seguir.
- Cada Builder poderá criar objetos diferentes, porém, seguindo processo de criação ditado pela interface.

- `getGeradorPDF()`: Retorna o objeto já configurado.
- `setPageConfiguration()`: Define as configurações básicas das páginas do PDF.
- `setMargin()`: Configura as margens das páginas do PDF.
- `setHeader()`: Configura o cabeçalho das páginas do PDF.
- `setFooter()`: Configura o rodapé das páginas do PDF.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BuilderSolucao.PDF
8  {
9      0 referências
10     public interface BuilderInterfaceGeradorPDF
11     {
12         0 referências
13         public GeradorPDF getGeradorPDF();
14         0 referências
15         public void setPageConfiguration();
16         0 referências
17         public void setMargin();
18         0 referências
19         public void setHeader();
20         0 referências
21         public void setFooter();
22     }
23 }
```



- Nosso projeto irá ficar com a estrutura do diagrama.
- As classes BuilderA4 e BuilderA3 devem implementar a interface BuilderInterfaceGeradorPDF.

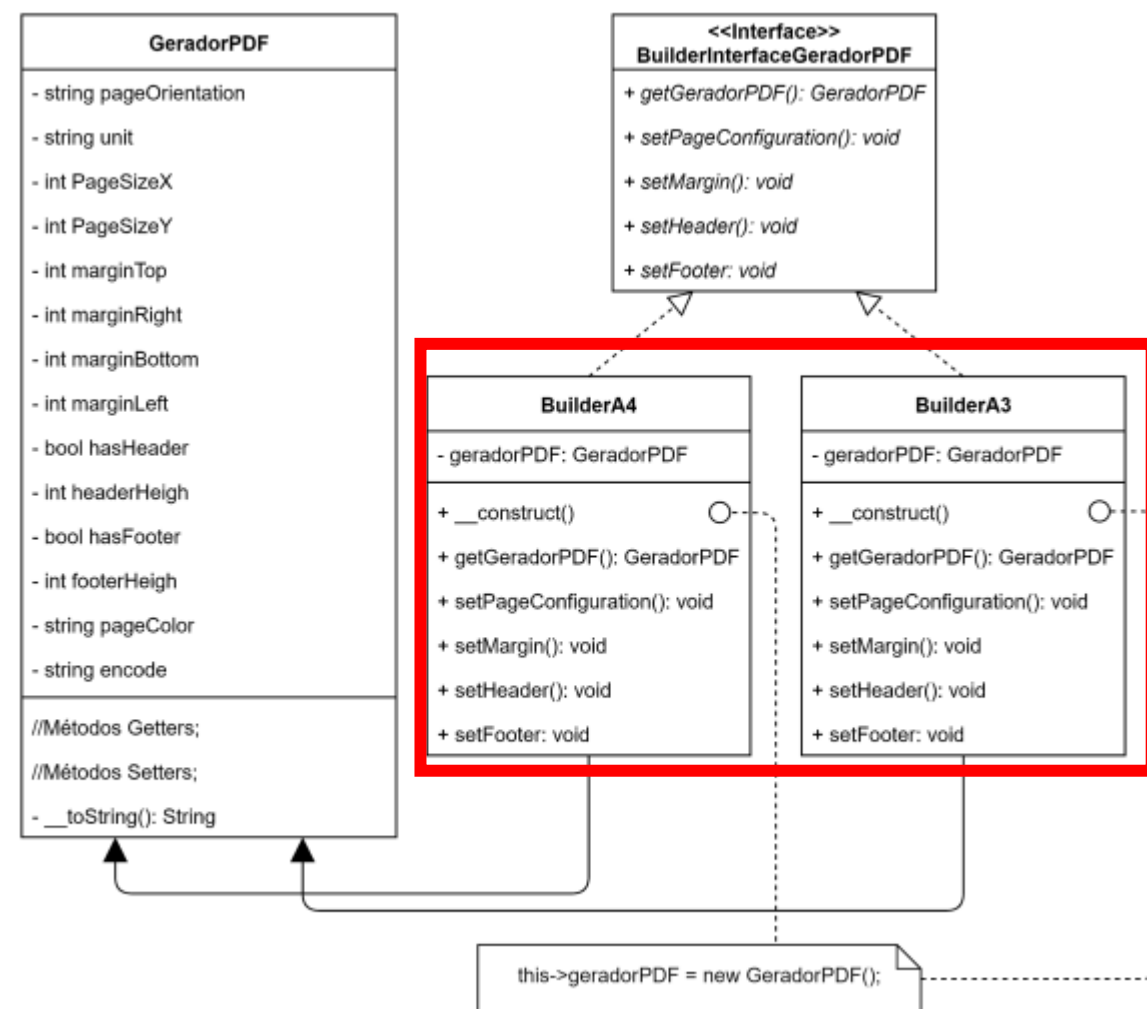


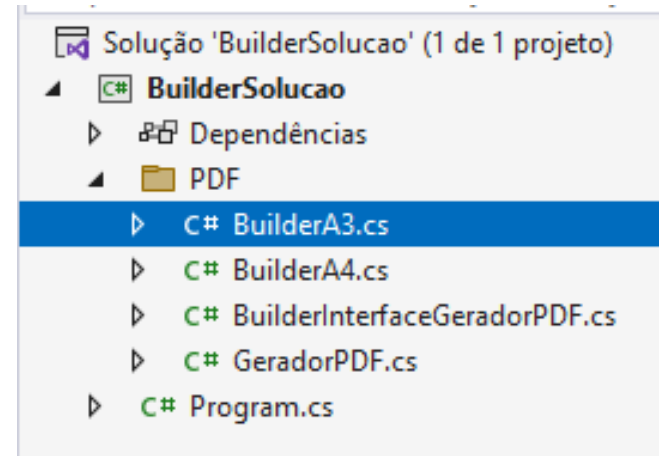
Diagrama de classes utilizando Builders divididos em passos de criação

```
1  using BuilderSolucao.PDF;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace BuilderSolucao.PDF
9  {
10     3 referências
11     public class BuilderA3 : BuilderInterfaceGeradorPDF{
12         private GeradorPDF geradorPDF;
13
14         1 referência
15         public BuilderA3() {
16             this.geradorPDF = new GeradorPDF();
17         }
18
19         1 referência
20         public GeradorPDF getGeradorPDF()
21         {
22             return this.geradorPDF;
23         }
24
25         1 referência
26         public void setFooter()
27         {
28             this.geradorPDF.hasFooter = true;
29             this.geradorPDF.footerHeight = 15;
30         }
31
32         1 referência
33         public void setHeader()
34         {
35             this.geradorPDF.hasHeader = true;
36             this.geradorPDF.headerHeight = 15;
37         }
38     }
```

```
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
1 referência
public void setMargin()
{
    this.geradorPDF.marginLeft = 20;
    this.geradorPDF.marginRight = 20;
    this.geradorPDF.marginBottom = 30;
    this.geradorPDF.marginTop = 30;
}

1 referência
public void setPageConfiguration()
{
    this.geradorPDF.pageOrientation = "portrait";
    this.geradorPDF.unit = "mm";
    this.geradorPDF.pageSizeX = 297;
    this.geradorPDF.pageSizeY = 420;
    this.geradorPDF.pageColor = "#ffffff";
    this.geradorPDF.encode = "UTF-8";
}
```




```
1  using BuilderSolucao.PDF;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace BuilderSolucao.PDF
9  {
10     3 referências
11     public class BuilderA4 : BuilderInterfaceGeradorPDF{
12         private GeradorPDF geradorPDF;
13
14         1 referência
15         public BuilderA4() {
16             this.geradorPDF = new GeradorPDF();
17         }
18
19         1 referência
20         public GeradorPDF getGeradorPDF()
21         {
22             return this.geradorPDF;
23         }
24
25         1 referência
26         public void setFooter()
27         {
28             this.geradorPDF.hasFooter = true;
29             this.geradorPDF.footerHeight = 15;
30         }
31
32         1 referência
33         public void setHeader()
34         {
35             this.geradorPDF.hasHeader = true;
36             this.geradorPDF.headerHeight = 15;
37         }
38     }
```

```
34
35
36     1 referência
37     public void setMargin()
38     {
39         this.geradorPDF.marginLeft = 20;
40         this.geradorPDF.marginRight = 20;
41         this.geradorPDF.marginBottom = 30;
42         this.geradorPDF.marginTop = 30;
43     }
44
45     1 referência
46     public void setPageConfiguration()
47     {
48         this.geradorPDF.pageOrientation = "portrait";
49         this.geradorPDF.unit = "mm";
50         this.geradorPDF.pageSizeX = 297;
51         this.geradorPDF.pageSizeY = 420;
52         this.geradorPDF.pageColor = "#ffffff";
53         this.geradorPDF.encode = "UTF-8";
54     }
55 }
```

Solução 'BuilderSolucao' (1 de 1 projeto)

- BuilderSolucao
 - Dependências
 - PDF
 - BuilderA3.cs
 - BuilderA4.cs**
 - BuilderInterfaceGeradorPDF.cs
 - GeradorPDF.cs
 - Program.cs

```
PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HasHeader = True
HeaderHeight = 15
HasFooter = True
FooterHeight = 15
PageColor = #ffffff
Encode = UTF-8
```

```
PageOrientation = portrait
Unit = mm
PageSizeX = 210
PageSizeY = 297
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HasHeader = False
HeaderHeight = 0
HasFooter = False
FooterHeight = 0
PageColor = #ffffff
Encode = UTF-8
```

```
1 using BuilderSolucao.PDF;
2
3 BuilderA4 builderA4 = new BuilderA4();
4 builderA4.setPageConfiguration();
5 builderA4.setMargin();
6 builderA4.setHeader();
7 builderA4.setFooter();
8 GeradorPDF pdfA4 = builderA4.getGeradorPDF();
9 Console.WriteLine(pdfA4.ToString());
10
11 BuilderA3 builderA3 = new BuilderA3();
12 builderA3.setPageConfiguration();
13 builderA3.setMargin();
14 GeradorPDF pdfA3 = builderA3.getGeradorPDF();
15 Console.WriteLine(pdfA3.ToString());
```

- Agora nosso Builder é mais flexível.
- Temos um definição com e outra sem cabeçalho e rodapé.
- HasHeader e HasFooter assumiram os valores padrões da classe GeradorPDF → “false”.
- Como afirma a definição conseguimos criar outra representação de um objeto de GeradorPDF, podemos obter PDFs A4 e A3 com ou sem cabeçalho ou rodapé.
- Com o mesmo processo de criação.

- Embora nosso código ficou mais flexível, o cliente voltou a ter responsabilidade a respeito da criação dos objetos, nessa versão atual do código Cliente precisa ainda chamar os métodos dos Builders para que os objetos sejam criados corretamente.
- Para resolver esse problema o padrão Builder sugere a criação de um diretor.

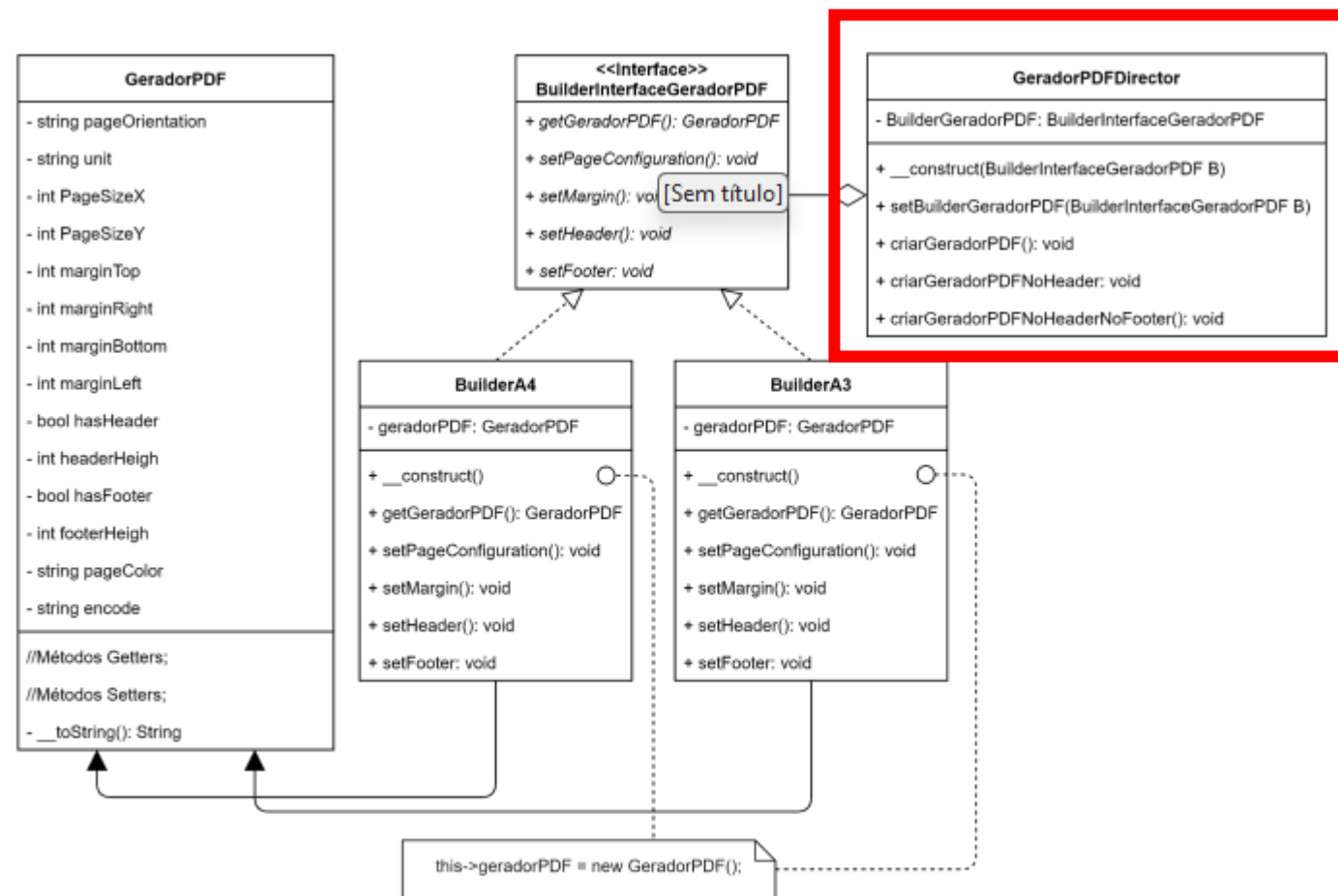


Diagrama de classes utilizando o padrão Builder completo

- Trata-se de uma classe que controla quais métodos dos Builders serão chamados em cada contexto, e também em qual ordem tais métodos serão chamados
- Vamos criar a classe **GeradorPDFDirector**.

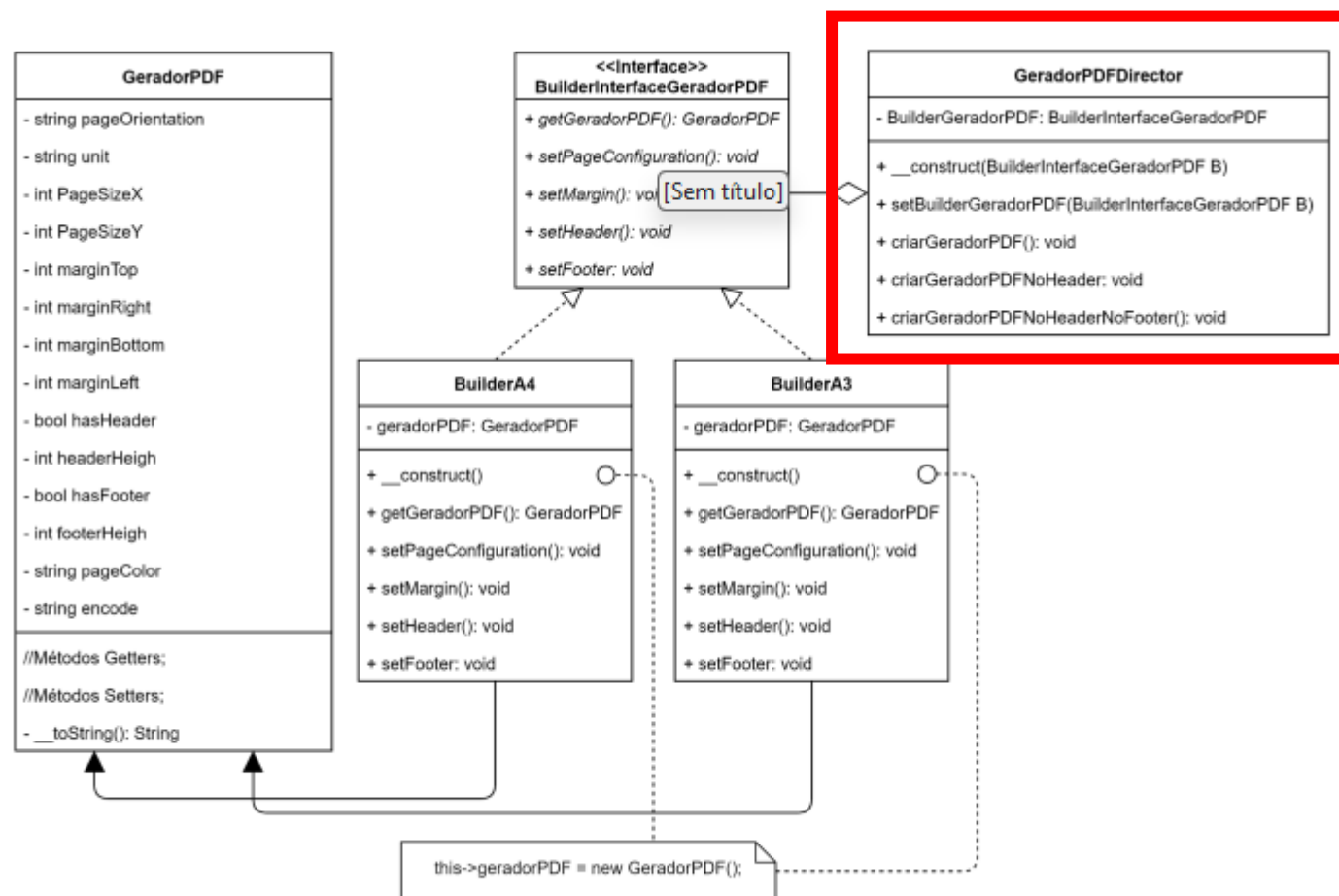


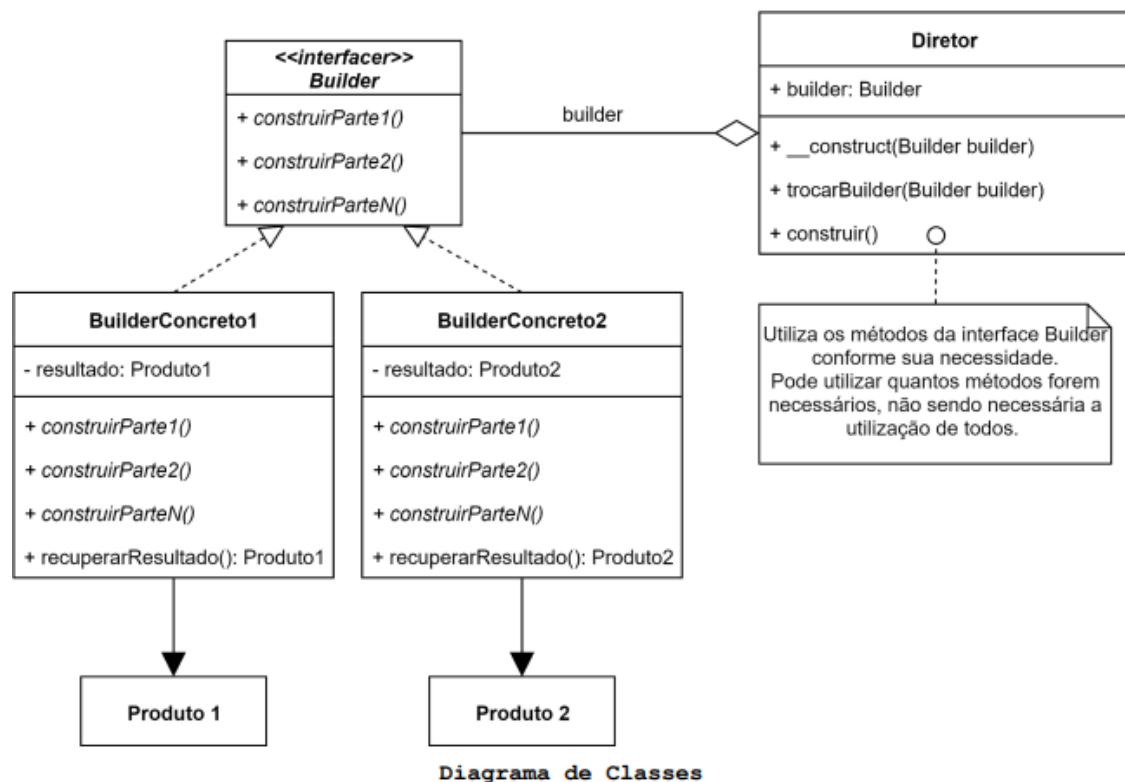
Diagrama de classes utilizando o padrão Builder completo

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BuilderSolucao.PDF
8 {
9     5 referências
10     public class GeradorPDFDirector {
11
12         private BuilderInterfaceGeradorPDF builderGeradorPDF;
13
14         2 referências
15         public GeradorPDFDirector(BuilderInterfaceGeradorPDF builder)
16         {
17             this.builderGeradorPDF = builder;
18         }
19
20         0 referências
21         public void setBuilderGeradorPDF(BuilderInterfaceGeradorPDF builder)
22         {
23             this.builderGeradorPDF = builder;
24         }
25
26         1 referência
27         public void criarGeradorPDF()
28         {
29             this.builderGeradorPDF.setPageConfiguration();
30             this.builderGeradorPDF.setMargin();
31             this.builderGeradorPDF.setHeader();
32             this.builderGeradorPDF.setFooter();
33         }
34     }
```

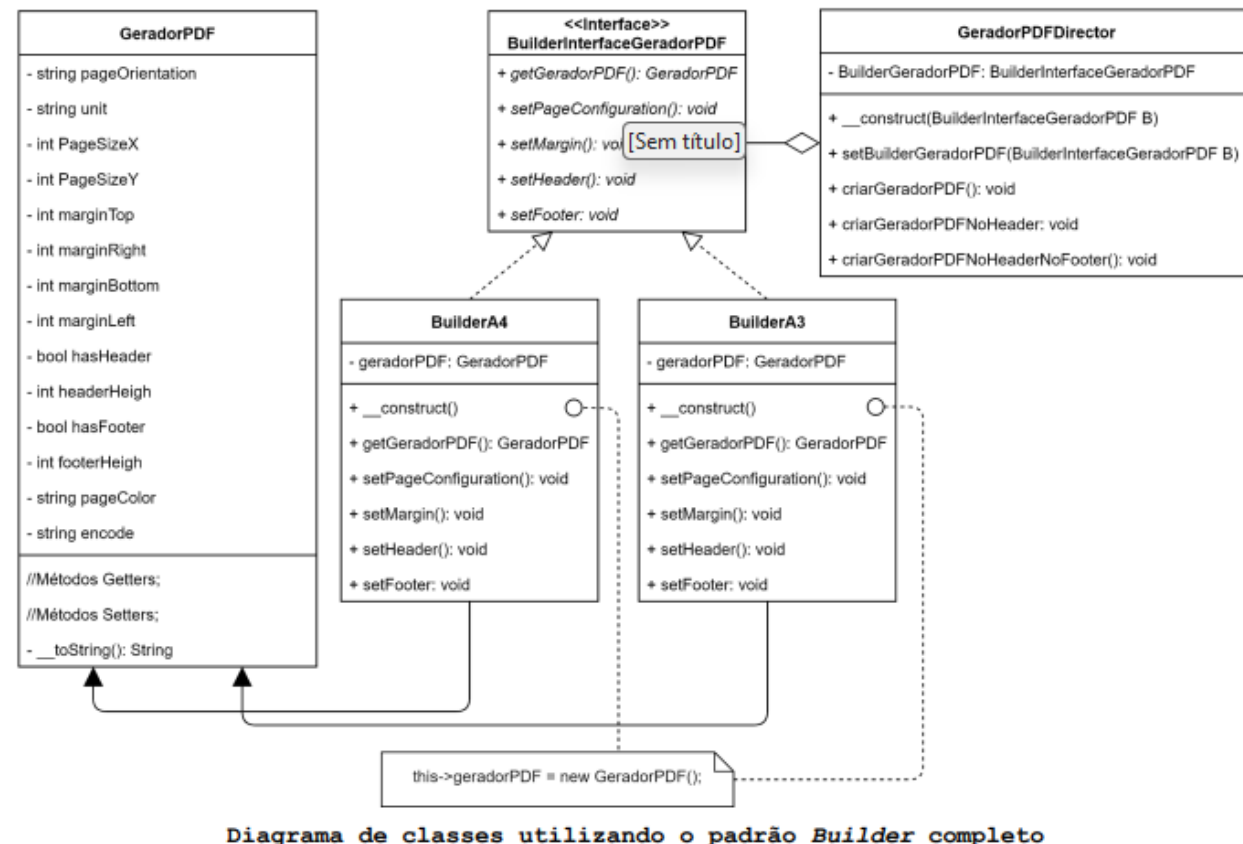
```
31
32
33     0 referências
34     public void criarGeradorPDFNotHeader()
35     {
36         this.builderGeradorPDF.setPageConfiguration();
37         this.builderGeradorPDF.setMargin();
38         this.builderGeradorPDF.setFooter();
39     }
40
41     0 referências
42     public void criarGeradorPDFNotFooter()
43     {
44         this.builderGeradorPDF.setPageConfiguration();
45         this.builderGeradorPDF.setMargin();
46         this.builderGeradorPDF.setHeader();
47     }
48
49     1 referência
50     public void criarGeradorPDFNotHeaderNotFooter()
51     {
52         this.builderGeradorPDF.setPageConfiguration();
53         this.builderGeradorPDF.setMargin();
54     }
55 }
```

Solução 'BuilderSolucao' (1 de 1 projeto)

- ▲ C# BuilderSolucao
 - ▷ Dependências
 - ▲ PDF
 - ▷ C# BuilderA3.cs
 - ▷ C# BuilderA4.cs
 - ▷ C# BuilderInterfaceGeradorPDF.cs
 - ▷ C# GeradorPDF.cs
 - ▷ C# GeradorPDFDirector.cs
 - ▷ C# Program.cs

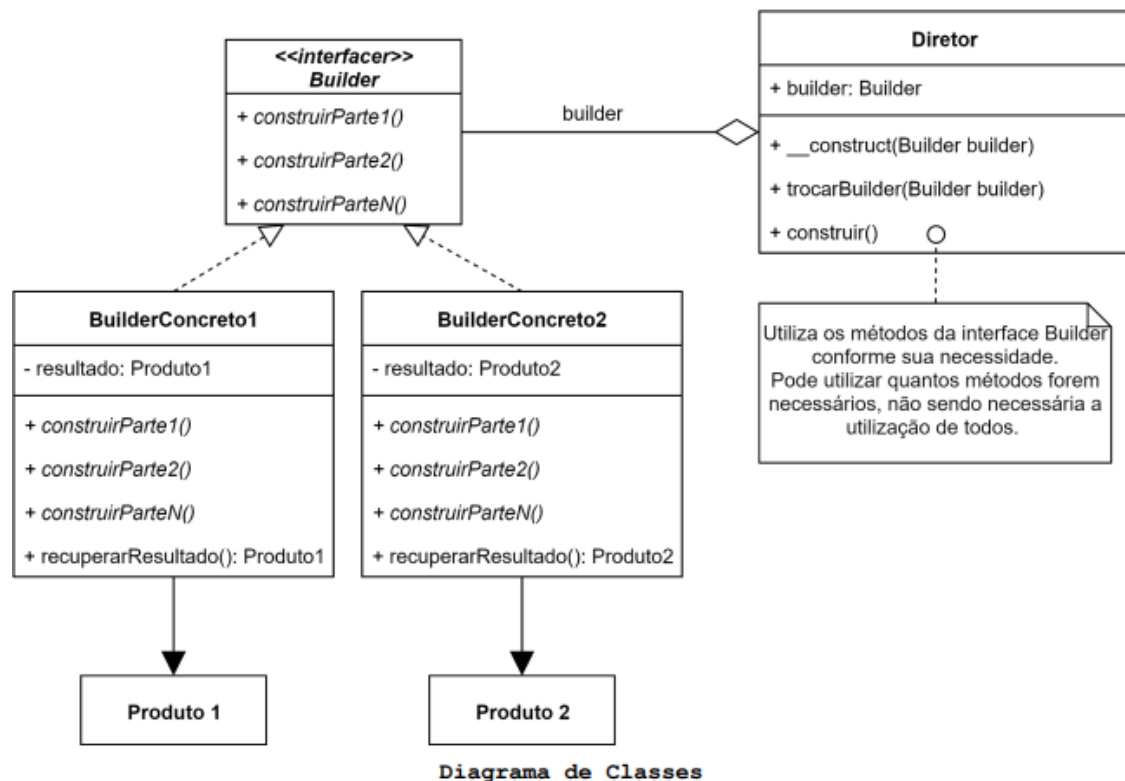


Modelo Genérico do Builder

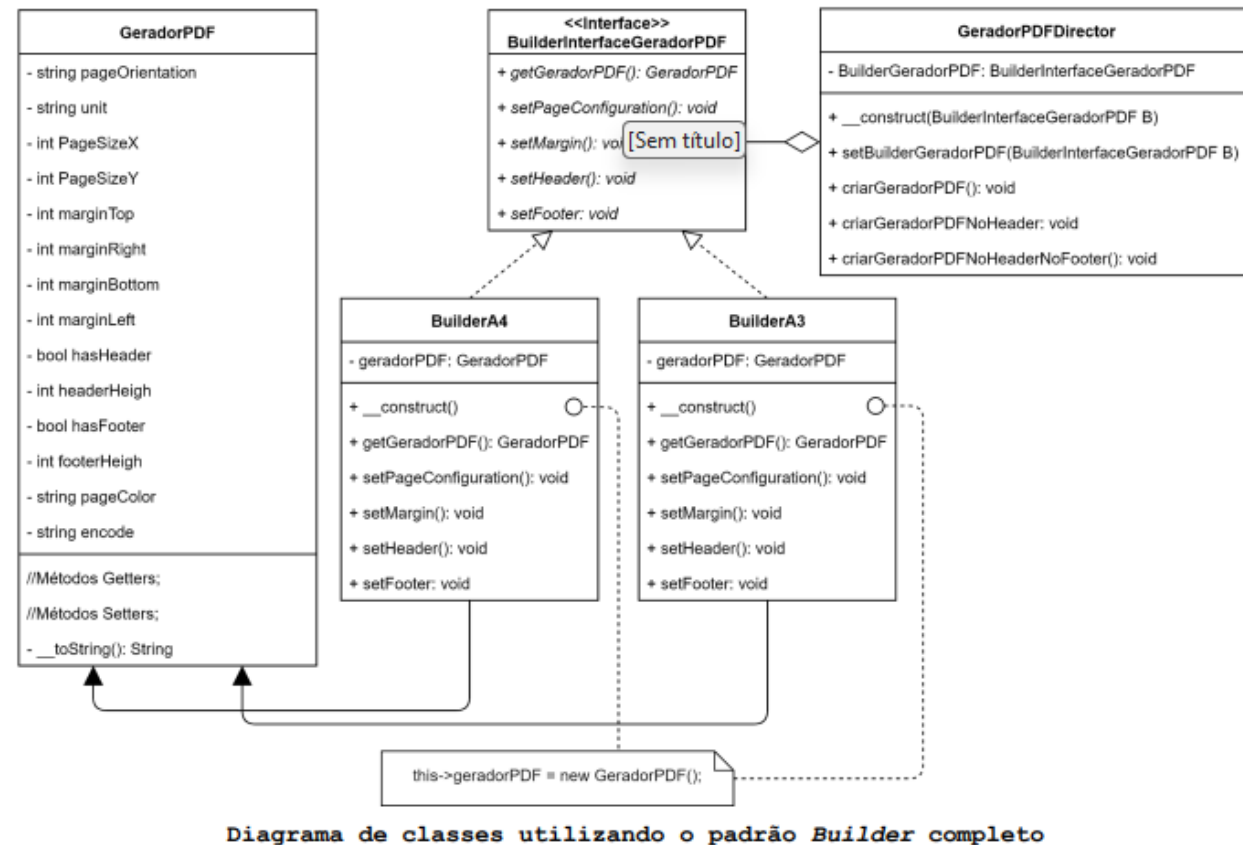


Modelo Aplicado do Builder

Modelo Genérico do Builder



Modelo Aplicado do Builder



É importante dizer que Builders distintos não necessariamente precisam criar objetos de uma mesma classe mudando apenas seus atributos. Eles podem criar objetos de classes diferentes e tais classes nem precisam pertencer a mesma hierarquia de classes. O único requisito é que o processo de criação seja o mesmo.

```
PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HeaderHeight = 15
FooterHeight = 15
PageColor = #++++++
Encode = UTF-8
```

```
PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HeaderHeight = 0
FooterHeight = 0
PageColor = #++++++
Encode = UTF-8
```

```
1 using BuilderSolucao.PDF;
2
3 BuilderA4 builderA4 = new BuilderA4();
4 GeradorPDFDirector director = new GeradorPDFDirector(builderA4);
5 director.criarGeradorPDF();
6 GeradorPDF pdfA4 = builderA4.getGeradorPDF();
7 Console.WriteLine(pdfA4.ToString());
8
9 BuilderA3 builderA3 = new BuilderA3();
10 GeradorPDFDirector director2 = new GeradorPDFDirector(builderA3);
11 director2.criarGeradorPDFNotHeaderNotFooter();
12 GeradorPDF pdfA3 = builderA3.getGeradorPDF();
13 Console.WriteLine(pdfA3.ToString());
```

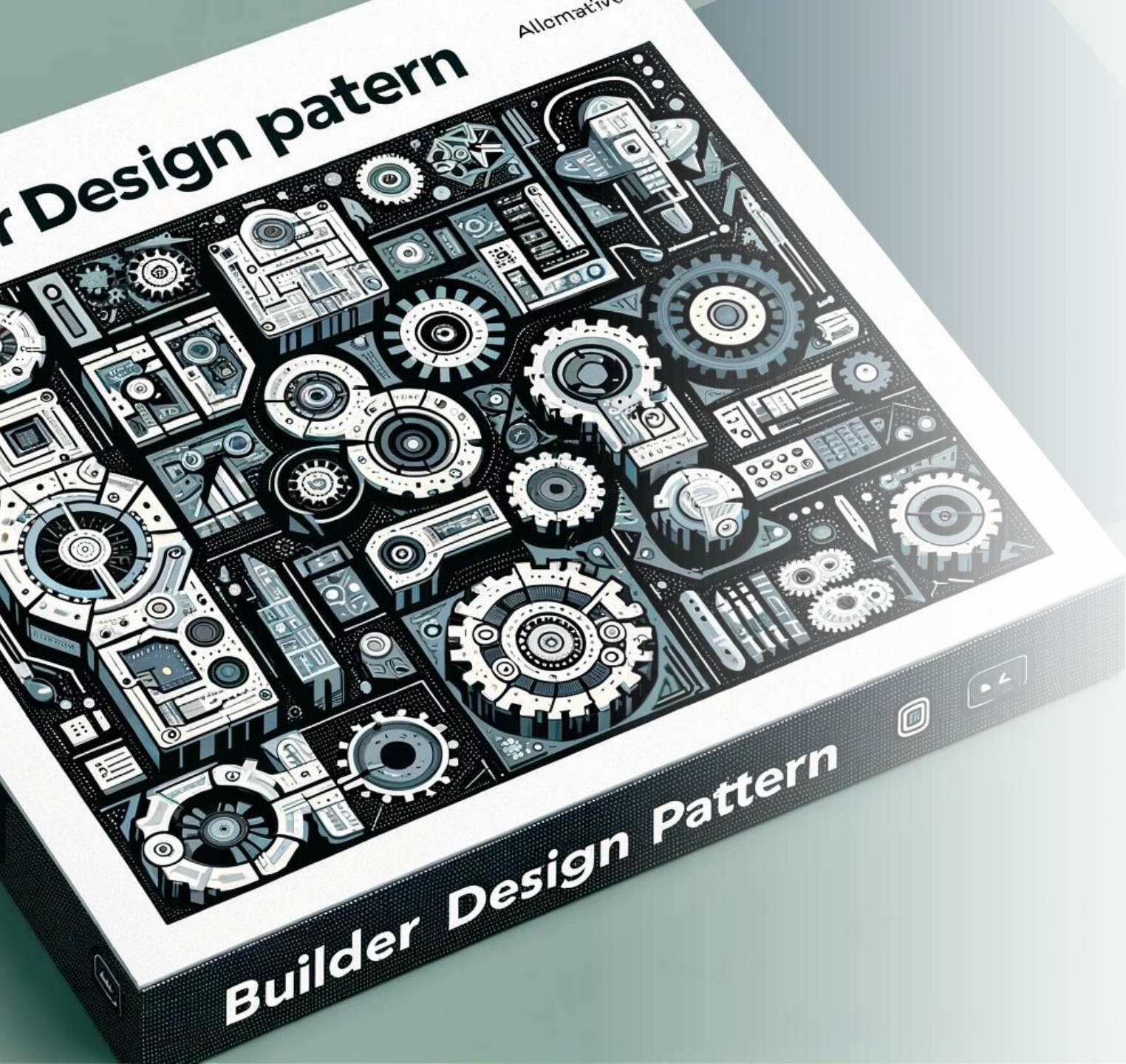
- Na classe GeradorPDFDirector podemos deixa-la pré-configuradas diversas representações de um objeto que será criado por um Builder.
- Na classe implementada temos:
 - Objeto totalmente configurado
 - Objeto sem cabeçalho
 - Objeto sem rodapé
 - Objeto sem cabeçalho e rodapé

```
PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HeaderHeight = 15
FooterHeight = 15
PageColor = #++++++
Encode = UTF-8
```

```
PageOrientation = portrait
Unit = mm
PageSizeX = 297
PageSizeY = 420
MarginTop = 30
MarginRight = 20
MarginBottom = 30
MarginLeft = 20
HeaderHeight = 0
FooterHeight = 0
PageColor = #++++++
Encode = UTF-8
```

```
1 using BuilderSolucao.PDF;
2
3 BuilderA4 builderA4 = new BuilderA4();
4 GeradorPDFDirector director = new GeradorPDFDirector(builderA4);
5 director.criarGeradorPDF();
6 GeradorPDF pdfA4 = builderA4.getGeradorPDF();
7 Console.WriteLine(pdfA4.ToString());
8
9 BuilderA3 builderA3 = new BuilderA3();
10 GeradorPDFDirector director2 = new GeradorPDFDirector(builderA3);
11 director2.criarGeradorPDFNotHeaderNotFooter();
12 GeradorPDF pdfA3 = builderA3.getGeradorPDF();
13 Console.WriteLine(pdfA3.ToString());
```

- O Cliente delega a chamada dos métodos do Builder para o diretor (GeradorPDFDirector).

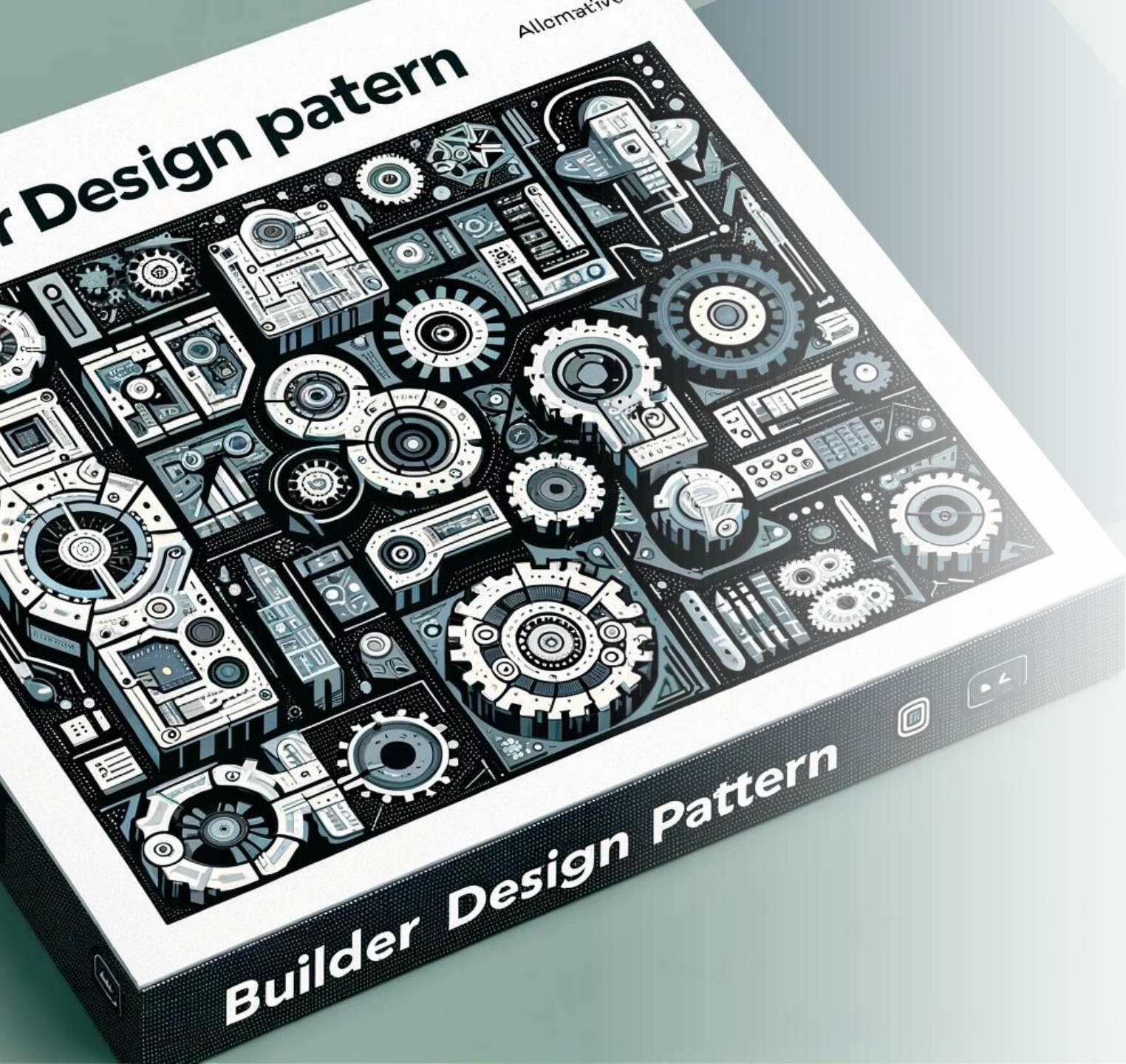


Builder

Implementação Java

Padrões de Projeto Criacional I

Prof. Me Jefferson Passerini



Builder

Consequências

Padrões de Projeto Criacional I

Prof. Me Jefferson Passerini

- Permite variar a representação interna de um produto.
 - O objeto Builder fornece ao Diretor uma interface abstrata para construir o produto.
 - Tal interface permite que o Builder oculte a representação e a estrutura interna do produto, e também oculta como o produto é montado.
 - O produto é construído por meio de uma interface abstrata, então, para alterar a representação interna do produto **basta trocar o Builder do Diretor.**

- Isola o código de construção e representação.
 - O padrão Builder aprimora a modularidade, encapsulando a maneira a qual um objeto complexo é construído e representado.
 - Os clientes não precisam ter conhecimento sobre como as classes definem a estrutura interna do produto.
 - As classes de produto ficam encapsuladas nos Builders concretos e não aparecem na interface Builder.
 - O código de criação dos produtos são escritos apenas uma vez, assim diferentes diretores podem compartilhar o mesmo código de criação.

- Oferece um controle mais fino sobre o processo de construção.
 - Enquanto padrões criacionais constroem produtos de uma só vez, o padrão Builder constrói o Produto passo a passo, sob o controle do Diretor.
 - Somente quando o Produto é concluído, o Diretor o recupera do BuilderConcreto.
 - Portanto, a interface de Builder reflete o processo de construção do produto mais do que outros padrões criacionais.
 - Isso permite um controle mais preciso sobre o processo de criação e, conseqüentemente, a estrutura interna do produto resultante.