

Classes IOS – ObjC

Parte 2

X-Code

Prof. Agesandro Scarpioni

Classes – Métodos

- No Objective-C, criar métodos com muitos parâmetros e passar estes parâmetros para os métodos é bem diferente de outras linguagens. Vamos ver 3 exemplos:

- Java

- `public void CalcularImcComPesoeAltura(float peso, float altura) { código aqui };`

- VB

- `Public Sub CalcularImcComPesoeAltura(Byval peso as Single, Byval altura as Single) (código aqui) End Sub`

- Objective-C

- `-(void) calcularImcComPeso:(float) peso eAltura:(float) altura { código aqui }`

Classes – Métodos

- Em nosso exemplo de classes no Objective-C o método tem o nome `calcularImcComPeso e Altura`, veja que o nome foi adaptado para receber dois parâmetros, pois, teremos que passar os parâmetros entre o nome do método, sendo que cada parâmetro entrará próximo a sua respectiva parte do nome do método, os dois pontos indica a atribuição. Veja a chamada abaixo:
- `[a calcularImcComPeso: 83.8 eAltura:1.87];`

Classes – Métodos

- No header file (.h), declaramos o método:

```
1 //  
2 // Atleta.h  
3 // ClasseExemplo  
4 //  
5 // Created by agesandro scarpioni on 08/03/15.  
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 #import <Foundation/Foundation.h>  
10  
11 @interface Atleta : NSObject {  
12     NSString *nome;  
13     int idade;  
14  
15 }  
16  
17 -(void) setName: (NSString *)_nome;  
18 -(NSString *) getName;  
19 -(void) setIdade: (int) _idade;  
20 -(int) getIdade;  
21  
22 -(void) calcularImcComPeso:(float) peso eAltura:(float) altura;  
23  
24 @end  
25
```


Classes – Métodos

- No (.m), vamos implementar o método:

```

1 //
2 // Atleta.m
3 // ClassesExemplo
4 //
5 // Created by agesandro scarpioni on 20/04/13.
6 // Copyright (c) 2013 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import "Atleta.h"
10
11 @implementation Atleta
12 - (void) setName:(NSString *)_nome{
13     nome=_nome;
14 }
15
16 - (NSString *) getName{
17     return nome;
18 }
19
20 - (void) setIdade: (int)_idade{
21     idade=_idade;
22 }
23
24 - (int) getIdade{
25     return idade;
26 }
27
28
29 - (void) calcularImcComPeso:(float) peso eAltura:(float) altura{
30     float imc;
31     imc = peso / (altura * altura);
32     NSLog(@" 0 IMC de %@ é %f", self.getName, imc);
33 }
34
35 @end

```

```

28
29 - (void) calcularImcComPeso:(float) peso eAltura:(float) altura{
30     float imc;
31     imc = peso / (altura * altura);
32     NSLog(@" 0 IMC de %@ é %f", self.getName, imc);
33 }
34

```

DICAS: 1) Para ganhar tempo copie as assinaturas dos métodos no .h, cole no .m retire o ";" e coloque as chaves { }; depois basta implementar. 2) Self funciona como o this do Java. 3) Caso prefira use pow para trabalhar com potência → `imc = peso / pow(altura,2);`

Classes – Métodos

- Para apresentar o resultado com 2 casas decimais, formate o resultado como %0.2f

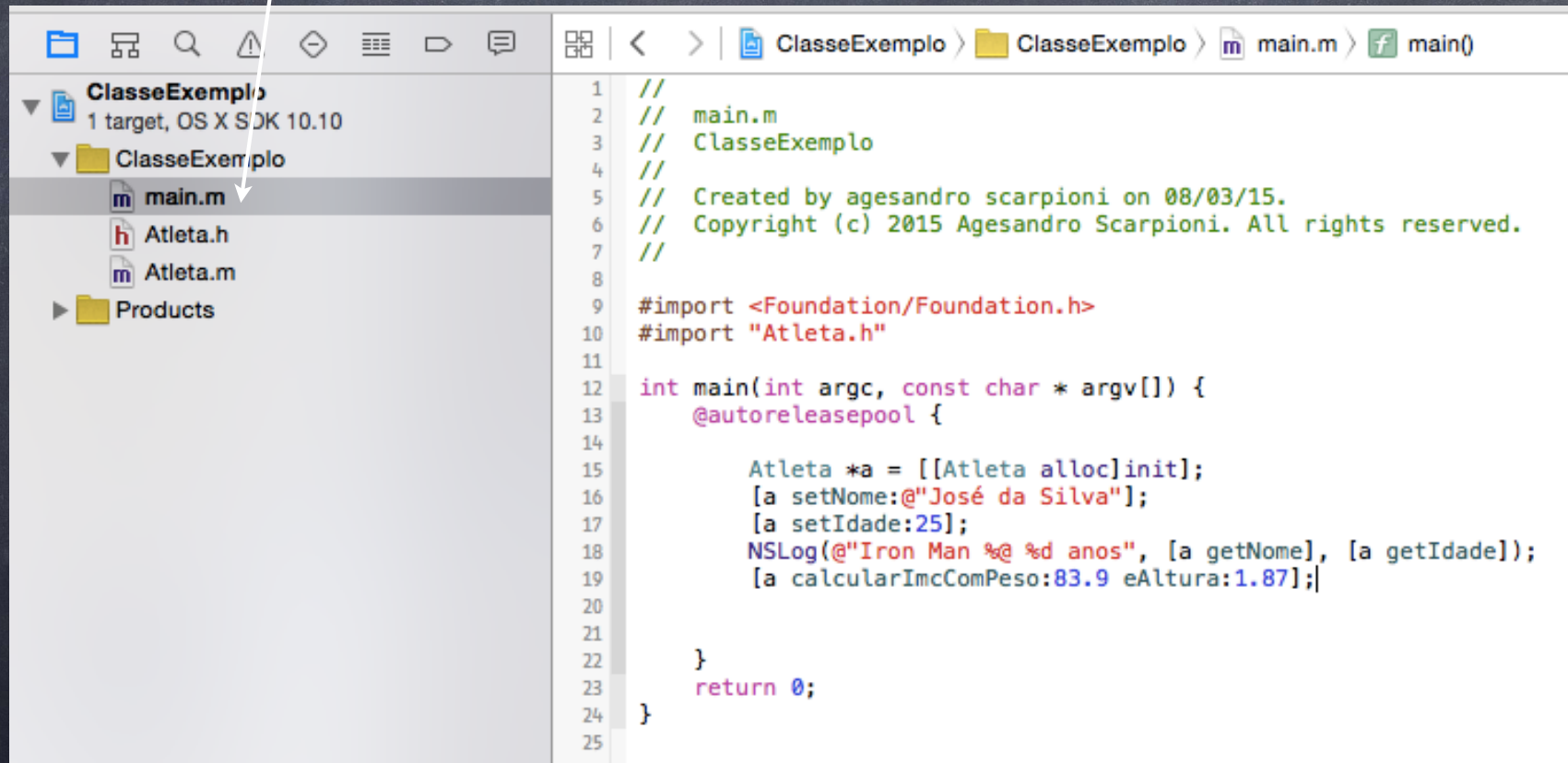
```
32 NSLog(@" 0 IMC de %@ é %0.2f", self.getNome, imc);
```

- Caso prefira, ao fazer a conta utilizando (altura*altura) use o comando pow. Você também pode acessar o get utilizando a notação com [].

```
-(void)CalcularImcComPeso:(float)peso eAltura:(float)altura{  
    float imc;  
    imc = peso/pow(altura,2);  
    NSLog(@"0 atleta %@ tem imc de %0.2f",[self getNome],imc);  
}
```


Classes – Métodos

- No Main vamos chamar o método calcularImcComPesoeAltura



```
1 //
2 //  main.m
3 //  ClasseExemplo
4 //
5 //  Created by agesandro scarpioni on 08/03/15.
6 //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11
12 int main(int argc, const char * argv[]) {
13     @autoreleasepool {
14
15         Atleta *a = [[Atleta alloc] init];
16         [a setName:@"José da Silva"];
17         [a setIdade:25];
18         NSLog(@"Iron Man %@ %d anos", [a getName], [a getIdade]);
19         [a calcularImcComPeso:83.9 eAltura:1.87];
20
21     }
22     return 0;
23 }
24
25
```


Classes – Métodos

- É uma boa prática substituir o “com” por “with” e o “e” por “and” nos arquivos .h, .m e Main.
- Assim o método passaria a se chamar `calcularImcwithPesoandAltura` ao invés de `calcularImccomPesoeAltura`.
- Faremos isto no próximo método.

Classes – Métodos

- No header file (.h), declaramos o método `calcularseuRendimentoNaAguawithTempoemHorasandMetros`.
- Diferente do primeiro método que era um (void) Java, (sub) VB, este será do tipo função, ou seja, deve retornar algo, neste caso um `NSString`.

Classes – Métodos

```
1 //
2 //  Atleta.h
3 //  ClasseExemplo
4 //
5 //  Created by agesandro scarpioni on 08/03/15.
6 //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10
11 @interface Atleta : NSObject {
12     NSString *nome;
13     int idade;
14
15 }
16
17 -(void) setName: (NSString *)_nome;
18 -(NSString *) getName;
19 -(void) setIdade: (int) _idade;
20 -(int) getIdade;
21
22 -(void) calcularImcComPeso:(float) peso eAltura:(float) altura;
23 -(NSString *) calcularSeuRendimentonaAguawithTempoemHoras:(float) horas andMetros:(float) metros;
24
25 @end
26
```

Dica: Como NSString é um objeto ele precisa ser declarados com * indicando que é um ponteiro.

Classes – Métodos

- No (.m) vamos implementar o método chamado `calcularSeuRendimentoNaAguawithTempoemHorasandMetros` que retornará um `NSString`.
- Como nossa função retorna um `NSString` e existe um cálculo com `float`, não podemos retornar um `float` concatenado com texto, por este motivo foi utilizado um `stringWithFormat`

```
34  
35 -(NSString *) calcularSeuRendimentoNaAguawithTempoemHoras:(float) horas andMetros:(float) metros{  
36     float resultado;  
37     resultado = metros / horas;  
38     NSString *texto = [NSString stringWithFormat:@"0 meu rendimento na água é %0.2f metros por hora", resultado];  
39     return texto;  
40 }  
41
```

OBS: Veja que formatamos o resultado com duas casas decimais `%0.2f`

Classes – Métodos

- No Main, vamos chamar o método calcularSeuRendimentonaAguawithTempoemHorasandMetros.

```
1 //
2 //  main.m
3 //  ClasseExemplo
4 //
5 //  Created by agesandro scarpioni on 08/03/15.
6 //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11
12 int main(int argc, const char * argv[]) {
13     @autoreleasepool {
14
15         Atleta *a = [[Atleta alloc] init];
16         [a setName:@"José da Silva"];
17         [a setIdade:25];
18         NSLog(@"Iron Man %@ %d anos", [a getName], [a getIdade]);
19         [a calcularImcComPeso:83.9 eAltura:1.87];
20         NSLog(@"%@", [a calcularSeuRendimentonaAguawithTempoemHoras:1.5 andMetros:8000]);
21
22     }
23     return 0;
24 }
25
26
```


Construtores

- Uma característica da linguagem OBJ-C é que ela não possui os construtores tradicionais como no VB utilizando `Public Sub New()` ou no Java utilizando o mesmo nome da classe, o método estático `alloc` e `init` fazem isto como já havíamos falado no conjunto de slide anterior, veja a imagem abaixo:

```
Atleta *a = [[Atleta alloc] init];  
[a setName:@"José da Silva"];  
[a setIdade:25];
```


Construtores

- No exemplo abaixo criamos o objeto e passamos alguns dados para o mesmo:

```
Atleta *a = [[Atleta alloc] init];  
[a setName:@"José da Silva"];  
[a setIdade:25];
```

- Mas poderíamos criar e já atribuir os dados em 1 linha

```
Atleta *a = [[Atleta alloc] initWithNome:@"José da Silva" eIdade:25];
```

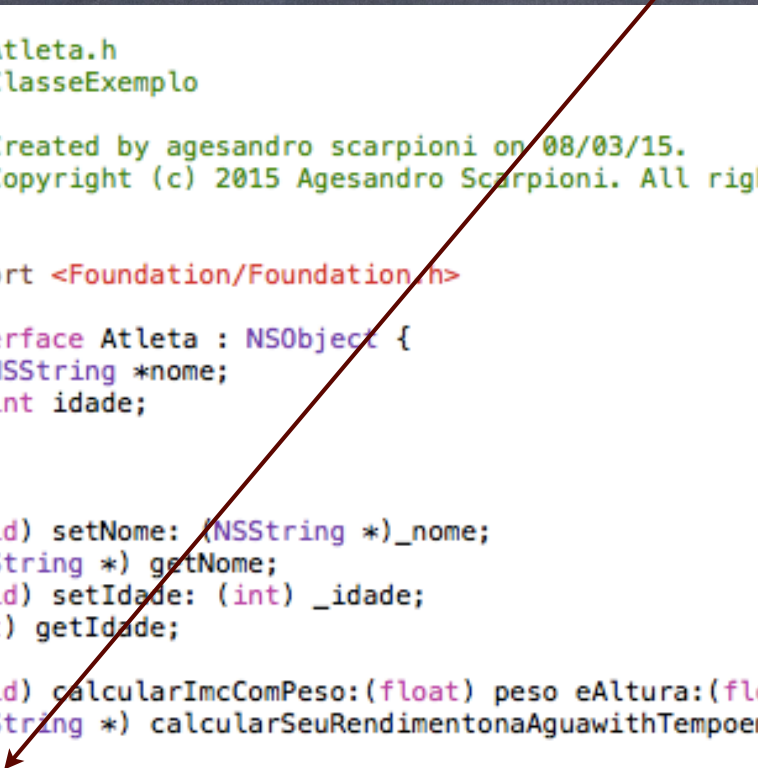
- Foi escrito "Com" e "E" em português, mas nas classes oficiais da Apple você vai encontrar os construtores como "With" e "And", portanto vamos seguir este padrão.

```
Atleta *a = [[Atleta alloc] initWithNome:@"José da Silva" AndIdade:25];
```


Construtores

- Precisamos declarar este método construtor no atleta.h

```
1 //
2 //  Atleta.h
3 //  ClasseExemplo
4 //
5 //  Created by agesandro scarpioni on 08/03/15.
6 //  Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10
11 @interface Atleta : NSObject {
12     NSString *nome;
13     int idade;
14 }
15
16 -(void) setName: (NSString *)_nome;
17 -(NSString *) getName;
18 -(void) setIdade: (int) _idade;
19 -(int) getIdade;
20
21 -(void) calcularImcComPeso:(float) peso eAltura:(float) altura;
22 -(NSString *) calcularSeuRendimentonaAguawithTempoemHoras:(float) horas andMetros:(float) metros;
23
24
25 -(Atleta *) initWithNome:(NSString *)_nome andIdade:(int)_idade;
26
27
28 @end
29
```



Construtores

- Precisamos implementar o método no atleta.m, lembre-se que podemos copiar a declaração do .h colar dentro da área @ implementation do .m retirar o ";" e colocar as chaves { }.

```
42 -(Atleta *) initWithNome:(NSString *)_nome andIdade:(int)_idade {  
43     self= [super init];  
44     if (self) { // Se a inicialização foi ok  
45         [self setName: _nome];  
46         [self setIdade: _idade];  
47     }  
48     return self;  
49 }
```

- O Self é parecido com o this. do Java, também podemos chamar métodos locais dentro da classe como por exemplo [self calcularXYZ].

Construtores

- No Main vamos criar um outro objeto com outro atleta, utilizando nosso novo método construtor.

```
1 //
2 // main.m
3 // ClasseExemplo
4 //
5 // Created by agesandro scarpioni on 08/03/15.
6 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11
12 int main(int argc, const char * argv[]) {
13     @autoreleasepool {
14
15         Atleta *a = [[Atleta alloc] init];
16         [a setName:@"José da Silva"];
17         [a setIdade:25];
18         NSLog(@"Iron Man %@ %d anos", [a getName], [a getIdade]);
19         [a calcularImcComPeso:83.9 eAltura:1.87];
20         NSLog(@"%@", [a calcularSeuRendimentonaAguawithTempoemHoras:1.5 andMetros:8000]);
21
22         Atleta *a2 = [[Atleta alloc] initWithNome:@"Maria Mendes" andIdade:18];
23         NSLog(@"Iron Man %@ %d anos", [a2 getName], [a2 getIdade]);
24         [a2 calcularImcComPeso:83.9 eAltura:1.87];
25         NSLog(@"%@", [a2 calcularSeuRendimentonaAguawithTempoemHoras:1.5 andMetros:8000]);
26
27     }
28     return 0;
29 }
30
31 }
```


Construtores

```

1 // main.m
2 // ClasseExemplo
3 //
4 // Created by agesandro scarpioni on 08/03/15.
5 // Copyright (c) 2015 Agesandro Scarpioni. All rights reserved.
6 //
7 //
8 #import <Foundation/Foundation.h>
9 #import "Atleta.h"
10
11
12 int main(int argc, const char * argv[]) {
13     @autoreleasepool {
14         Atleta *a = [[Atleta alloc] init];
15         [a setName:@"José da Silva"];
16         [a setIdade:25];
17         NSLog(@"Iron Man %d anos", [a getName], [a getIdade]);
18         [a calcularImcComPeso:83.9 eAltura:1.87];
19         NSLog(@"%d", [a calcularSeuRendimentoNaAguawithTempoemHoras:1.5 andMetros:8000]);
20
21         Atleta *a2 = [[Atleta alloc] initWithNome:@"Maria Mendes" andIdade:18];
22         NSLog(@"Iron Man %d anos", [a2 getName], [a2 getIdade]);
23         [a2 calcularImcComPeso:83.9 eAltura:1.87];
24         NSLog(@"%d", [a2 calcularSeuRendimentoNaAguawithTempoemHoras:1.5 andMetros:8000]);
25
26     }
27     return 0;
28 }

```

Output (highlighted in red circle):

```

2015-03-08 12:37:32.912 ClasseExemplo[1154:57058] Iron Man José da Silva 25 anos
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O IMC de José da Silva é 23.99
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O meu rendimento na água é 5333.33 metros por hora
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] Iron Man Maria Mendes 18 anos
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O IMC de Maria Mendes é 23.99
2015-03-08 12:37:32.915 ClasseExemplo[1154:57058] O meu rendimento na água é 5333.33 metros por hora
Program ended with exit code: 0

```

No Selection

```

2015-03-08 12:37:32.912 ClasseExemplo[1154:57058] Iron Man José da Silva 25 anos
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O IMC de José da Silva é 23.99
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O meu rendimento na água é 5333.33 metros por hora
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] Iron Man Maria Mendes 18 anos
2015-03-08 12:37:32.914 ClasseExemplo[1154:57058] O IMC de Maria Mendes é 23.99
2015-03-08 12:37:32.915 ClasseExemplo[1154:57058] O meu rendimento na água é 5333.33 metros por hora
Program ended with exit code: 0

```


Prática

Criação de um programa para testarmos todos os conceitos deste tópico.

- Ainda no projeto da classe Enfermeira criado anteriormente, desenvolva um método do tipo void e três métodos que retornem respectivamente um bool , um NSString e um int.
- Como dica, vocês podem criar um método que entre com a temperatura do corpo e retorne um booleano para indicar que o paciente está com febre, ou ainda um método que receba dois parâmetros e retorne a temperatura da água para um banho em graus celsius. Faça uso do método construtor com pelo menos 3 atributos.

Próxima aula

👁 @property.