FIAP

Classes IOS - ObjC

Parte 4 - @property

X-Code

Prof. Agesandro Scarpioni

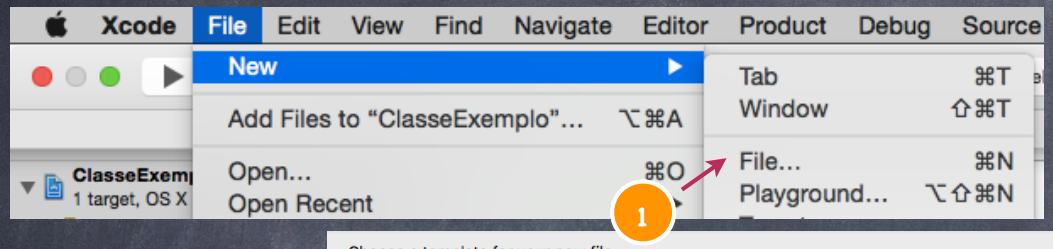
FIAP

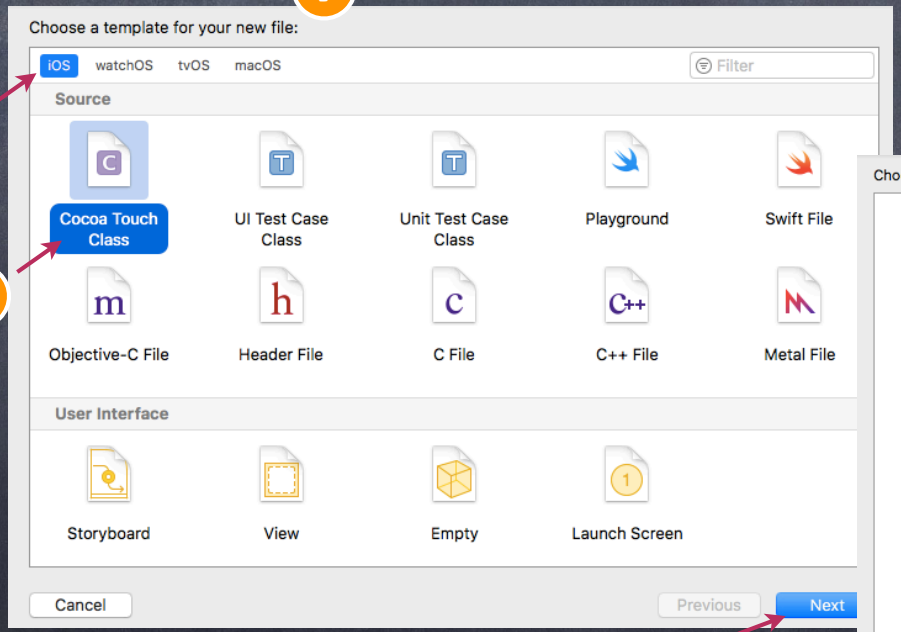
@Property

- Ainda em nosso projeto Atleta, inclua a classe MassagemAtleta.

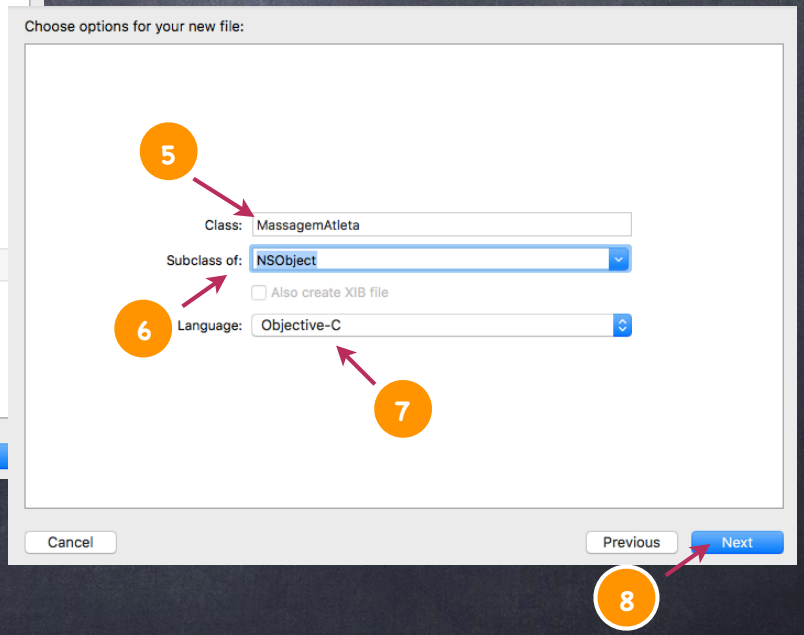
@property

- Vamos criar uma classe "MassagemAtleta". Clique em File->New->File-> IOS -> Source —> Cocoa Touch Class-> Next.





- Nomeie a classe MassagemAtleta como subclasse de NSObject



Dica: Esta classe será responsável por receber um atleta e fazer uma massagem para o mesmo.

@property

FIAP

- No MassagemAtleta.h, faça o import da classe Atleta.h e coloque as { } (1).
- Crie um método (void) massagearAtleta (2).
- Crie um @property (3) como é mostrado na imagem abaixo:

```
1  //
2  //  MassagemAtleta.h
3  //  ClasseExemplo
4  //
5  //  Created by Agesandro Scarpioni on 12/02/17.
6  //  Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11
12 @interface MassagemAtleta : NSObject {
13
14 }
15
16 -(void)massagearAtleta;
17
18 @property (nonatomic,retain) Atleta *atleta;
19
20 @end
```

1
2
3

@property

FIAP

- A palavra nonatomic indica que essa propriedade pode ser utilizada sem a necessidade de sincronizar as threads e portanto o acesso não é thread-safe. Como no iOS só existe um thread por aplicação, é obrigatório utilizar o nonatomic. Para desenvolver para Mac OS X teríamos também a opção atomic, que indica que o método é thread-safe e poderia ser utilizado de forma concorrente por diversas threads ao mesmo tempo.

- A palavra retain informa ao compilador para fazer o gerenciamento do retain e release automaticamente, para tirar essa responsabilidade do desenvolvedor.

```
1  //
2  //  MassagemAtleta.h
3  //  ClasseExemplo
4  //
5  //  Created by Agesandro Scarpioni on 12/02/17.
6  //  Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11
12 @interface MassagemAtleta : NSObject {
13
14 }
15
16 -(void)massagearAtleta;
17
18 @property (nonatomic,retain) Atleta *atleta;
19
20 @end
21
```

OBS: Para que os gets e sets sejam gerados automaticamente precisamos do @synthesize no arquivo MassagemAtleta.m.

@synthesize

- Informando a notação @synthesize para que de forma "sintetizada", automática, em tempo de execução, seja criado os getters e setters, depois implemente o método massagemAtleta.

```
1  //
2  //  MassagemAtleta.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 "MassagemAtleta.h"
10
11
12 @implementation MassagemAtleta
13
14 @synthesize atleta;
15
16 -(void)massagearAtleta{
17     NSLog(@"Massageando o atleta %@ %d", [atleta getNome], [atleta getIdade] );
18 }
19
20
21 @end
22
```

Dica: Quando você possuir várias propriedades (nome, telefone, cidade, bairro), você pode colocar apenas um @synthesize e os campos separados por vírgula ex.: @synthesize nome, telefone, cidade, bairro;

@property @synthesize

FIAP

- Vamos testar, no arquivo main.m faça o import da classe MassagemAtleta.

```
1  //
2  //  main.m
3  //  ClasseExemplo
4  //
5  //  Created by Agesandro Scarpioni on 05/02/17.
6  //  Copyright © 2017 Agesandro Scarpioni. All rights reserved.
7  //
8
9  #import <Foundation/Foundation.h>
10 #import "Atleta.h"
11 #import "MassagemAtleta.h"
```

- Ainda no main.m vamos criar um objeto da classe MassagemAtleta.

```
26          NSLog(@"%@", [a2 calcularSeuRendimentonaAguawithTempoemHoras:1.5 andMetros:8000]);
27
28          MassagemAtleta *m = [[MassagemAtleta alloc] init];
29          [m setAtleta:a]; //este método foi gerado automaticamente pelas notações @property e @synthesize
30          [m massagearAtleta];
31      }
32      return 0;
33  }
```

- Execute command + run e veja o resultado.

```
2017-02-12 10:02:03.659600 ClasseExemplo[1036:44662] Iron Man José da Silva 25 anos
2017-02-12 10:02:03.659873 ClasseExemplo[1036:44662] O imc de José da Silva é 23.99
2017-02-12 10:02:03.659999 ClasseExemplo[1036:44662] O meu rendimento na água é 5333.33 metros por hora
2017-02-12 10:02:03.660020 ClasseExemplo[1036:44662] Iron Man Maria Mendes 18 anos
2017-02-12 10:02:03.660077 ClasseExemplo[1036:44662] O imc de Maria Mendes é 23.99
2017-02-12 10:02:03.660113 ClasseExemplo[1036:44662] O meu rendimento na água é 5333.33 metros por hora
2017-02-12 10:02:03.660134 ClasseExemplo[1036:44662] Massageando o atleta José da Silva 25
Program ended with exit code: 0
```

+ sobre @property

- Atributos de propriedades disponíveis.

Categoria	Atributos de exemplos	Descrição
Nomeclatura de métodos	setter, getter	Permite que o desenvolvedor substitua o nome dos métodos gerados.
Permissão de gravação	readonly, readwrite	Permite que o desenvolvedor especifique que uma propriedade é apenas para leitura (não tem um método setter).
Semântica de setters	assign, retain, copy	Permite que o desenvolvedor controle o gerenciamento de memória dos valores das propriedades.
Thready safety	nonatomic	Permite que o desenvolvedor controle o gerenciamento de memória dos valores das propriedades.

- Segue a explicação de cada categoria nos próximos slides

FIAP

+ sobre @property

NOMECLATURA DE MÉTODOS:

- Por default para um propriedade nome, o compilador produz um método getter também chamado nome, e um método setter, setNome, Podemos substituir esses nomes especificando explicitamente alternativas por meio dos atributos getter e setter opcionais. Veja que no exemplo abaixo temos um getter, isAtivo, e um setter, setAtivo, estes tipos são comumente utilizados em um tipo de dado BOOL.

```
@property (getter=isSelected) BOOL selected;

@property (getter=isSelected setter=setingSelect:) BOOL selected;

@property (nonatomic, getter=isSelected setter=setingSelect:) BOOL selected;
```

FIAP

+ sobre @property

PERMISSÃO DE GRAVAÇÃO:

- Por default toda propriedade possui um método getter e um método setter, quando adicionamos o atributo readonly na declaração da propriedade, esta propreidade só pode ser lida, ou seja, não possui, setter. Para atribuirmos um valor a este atributo somente via código diretamente à variável de instância. A opção readwrite é a (default) e pode ser chamada explicitamente.

```
@property (readonly) NSString* categoriaAtleta;
```

+ sobre @property

SEMÂNTICA DE SETTERS:

- Esses atributos permitem que você especifique como um método setter lidará com o gerenciamento de memória. São 3 opções:
  - Assign - O setter utiliza uma simples instrução de atribuição. (opção default). A atribuição da propriedade é tratada como se fosse uma simples variável.
  - Copy - Uma cópia do novo valor é feita e guardada. É feita uma duplicata do valor fornecido.
  - Retain - O setter chama retain para o novo valor e release para o valor antigo. O retain fornece uma funcionalidade intermediária entre as duas outras opções.

FIAP

+ sobre @property

THREAD SAFETY

- Por default, as propriedades são atômicas. Isso significa que, em um ambiente multithread, a linguagem Objective-C garante que o valor obtido por meio de um getter, ou definido por um setter, seja sempre consistente e que não possa ser corrompido pelo acesso concorrente de outas threads.

- Essa proteção pode afetar o desempenho, pelo motivo de tempo para se obter as proteções necessárias no acesso concorrente, se a sua propriedade não for acessada por várias threads você pode invalidar essa proteção utilizando o atributo nonatomic, isso pode resultar em um leve ganho de desempenho caso a sua propriedade seja muito acessada.

```
@property (nonatomic) int idade;
```

FIAP

+ sobre @property

- Uso do acesso ao setter com os padrões objeto.propriedade e [objeto propriedade].

```
c.cidade=@"São Paulo";
c.cep=@"03022-030";
```

```
[c setEndereco:@"Av. Paulista"];
[c setNumero:900];
```

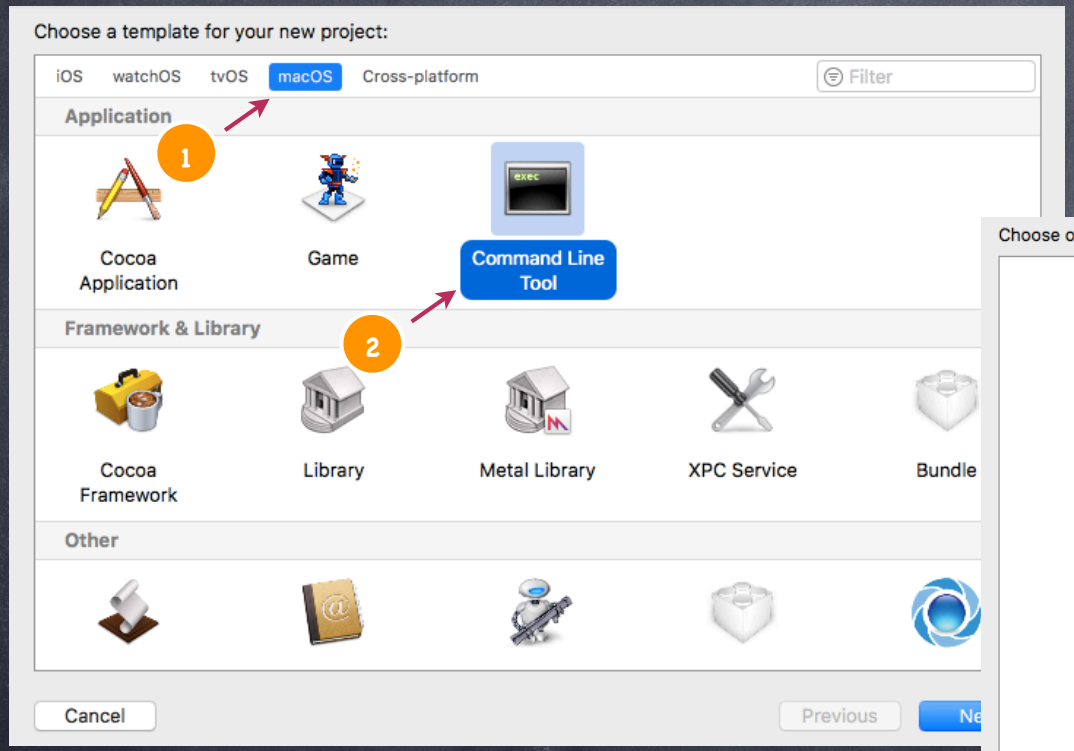
- Uso do acesso ao getter com os padrões objeto.propriedade e [objeto propriedade].

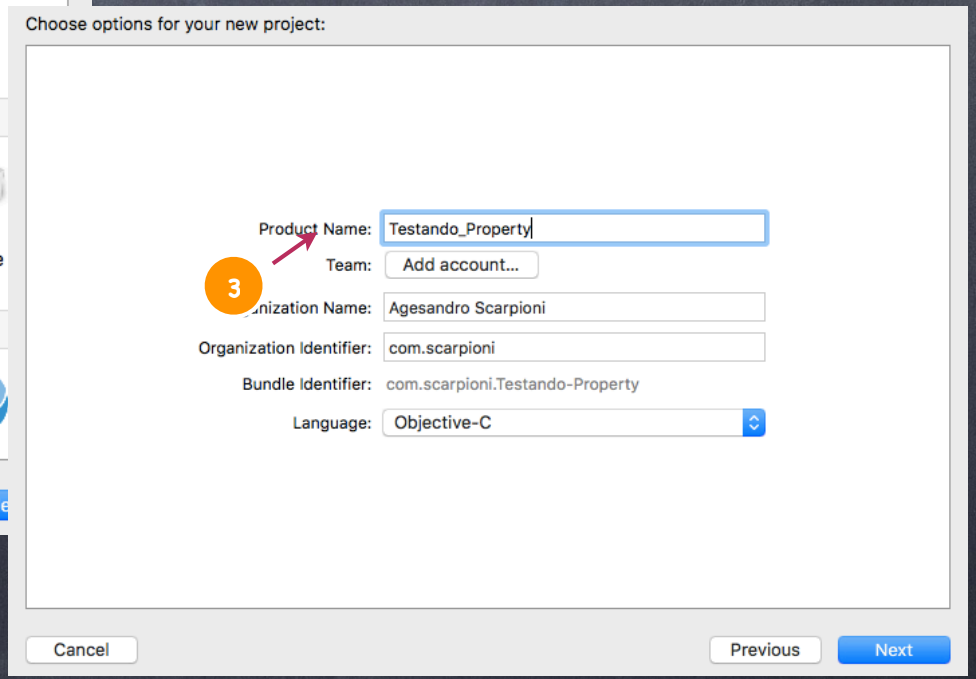
```
NSLog(@"O bairro é %@ na cidade de %@ ", c.bairro, c.cidade);
```

```
NSLog(@"O nome atribuído é %@, endereço %@ , %d", [c nome], [c endereco], [c numero]);
```

Prática @property

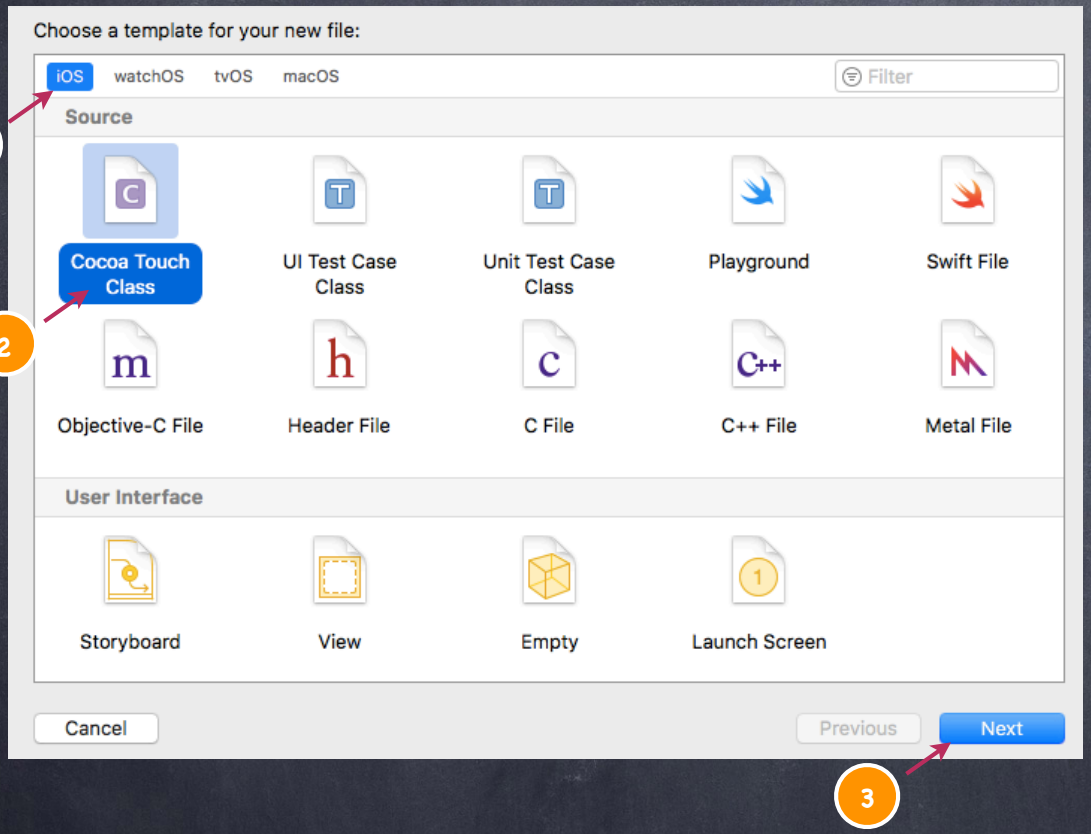
- Vamos testar a notação @property e @synthesize, crie um projeto do tipo MAC OSx (1) -> Command Line Tool (2), chamado: Testando_property (3).

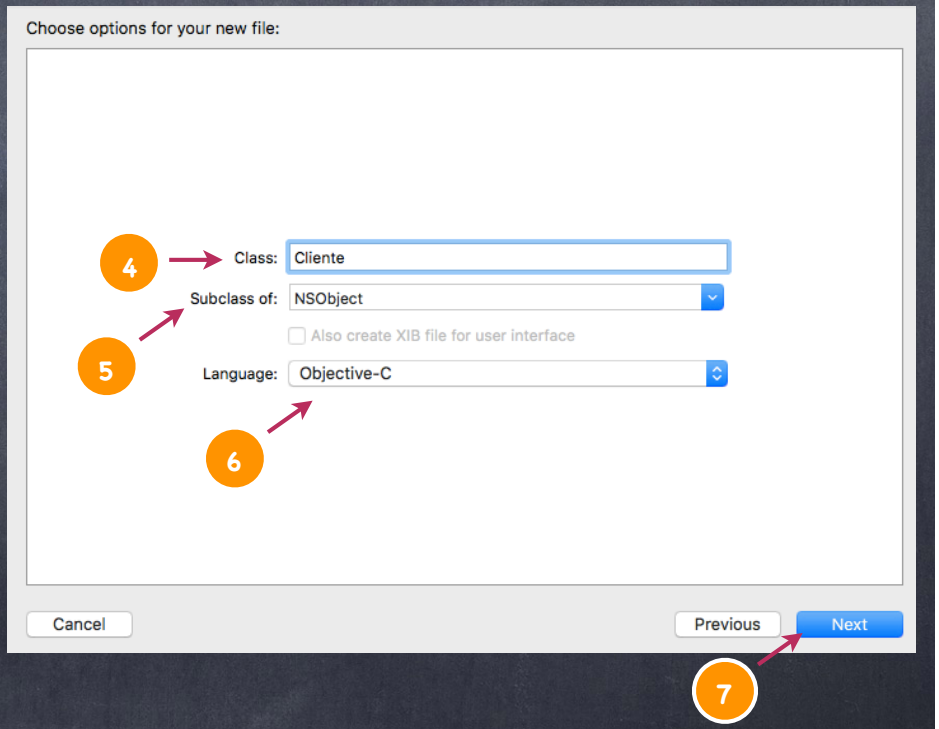




Prática @property

- Inclua uma nova classe (command + N) (1), (2), (3), chamada: Cliente (4), filha de NSObject (5), linguagem Objective-C (6).





FIAP

Prática @property

- Em Cliente.h utilizando @property declare os atributos (getter/setter) código, nome, endereço, número, complemento, bairro, cidade, cep e status. Todos devem receber string exceto código e número que devem ser inteiros.

- Crie um atributo ativo do tipo BOOL, utilizando nomeclatura de métodos para que o getter seja isAtivo e o setter setandoAtivo (normalmente não trocamos nome de set, caso precise este é o caminho).

- Monte um atributo do tipo readonly por exemplo categoria (carta para moto, carro, etc), ou seja, não terá setter, portanto precisamos alterar seu conteúdo via código por algum método diretamente na variável de instância.

- No main.m crie uma instância do seu cliente (alloc init) e faça o uso das formas de atribuição com [objeto propriedade:valor] e objeto.propriedade=valor, faça o mesmo para exibir o conteúdo desses atributos utilizando o padrão [objeto propriedade] e objeto.propriedade.

FIAP

Prática 2

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

- Ainda no projeto da classe Cliente (auto escola) criado anteriormente onde criamos nossos getters e setters em tempo de execução, desenvolva um método e faça o uso de lançamento de excessões. No main.m faça uso desse método.
- Crie um protocolo (exemplo ExamePratico pois sabemos que a forma de conduzir carro, caminhão ou moto é diferente) para usar com sua classe clientes da auto escola e implementando na classe Clientes.m. No main.m demonstre esse método.

FIAP

Próxima aula

- Exceptions e Protocolos