

DESENVOLVIMENTO PARA IOS – APPLE 01

IOS SDK 9

Professor: Pedro Henrique
prof.pedrohenrique.iossdk@gmail.com

AGENDA

- Pré-requisitos (o que você precisa saber antes);
- Tarefas e avaliação;
- Objetivos do dia
- Visão geral do iOS;
- Objective-C
 - Conceitos básicos

PRÉ-REQUISITOS

- Programação orientada a objetos
 - Classe
 - Instância
 - Mensagem
 - Método
 - Variável de Instância
 - Subclasse e superclasse
 - Ponteiros
- Um pouco de experiência

TAREFAS E AVALIAÇÃO

- Tarefas em sala;
- Desafios semanais;
- Para estudar em casa:
 - <https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
- Aplicativo KNOOWL (podemos mudar o nome).

OBJETIVOS DO DIA

- Entender a sintaxe básica
 - **@property**
 - Declaração e chamada de métodos

VISÃO GERAL DO IOS

- Core
- Core Services
- Media
- Cocoa Touch
- Ferramentas
 - XCode
 - Instruments
- Objective-C e Swift
- Frameworks
 - Foundation
 - UIKit
 - Core Data
 - Core Location, etc...

OBJECTIVE-C

- Um aditivo sobre a linguagem C;
 - Adiciona nova sintaxe para classes, métodos e etc.
- Conceito importante do dia: propriedades
 - Parecido com C#, sem declaração explícita de getter e setter (mas, assim como no C#, eles existem!)
- Vamos adentrar agora na parte legal, mas não se assustem com a sintaxe!

OBJECTIVE-C

CARTA.H

- Declarações públicas

CARTA.M

- Implementação privada

- Por que um arquivo .h e outro arquivo .m?
 - Ver link: http://pt.wikipedia.org/wiki/Arquivo_cabeçalho

CARTA.H

Notação de herança

Superclasse

```
11 @interface Carta : NSObject
```

Nome da sua
classe

NSObject é a classe raiz. Vai estar presente na hierarquia de todas as classes do iOS SDK, assim como nas classes que você criar.

```
20 @end
```

CARTA.M

```
11 @implementation Carta
```

```
12
```

```
13
```

```
14
```

```
15
```

```
16
```

```
17
```

```
18
```

```
19
```

```
20
```

```
@end
```

A superclasse não
precisa ser declarada!

CARTA.H

```
9  #import <Foundation/NSObject.h>
10
11 @interface Carta : NSObject
12
13 |
14
15
16
17
18
19
20 @end
```

Arquivo header da superclasse

CARTA.H

```
9  #import <Foundation/Foundation.h>
10
11  @interface Carta : NSObject
12
13
14
15
16
17
18
19
20  @end
```

No iOS, quando a superclasse é uma classe do sistema, devemos importar o respectivo framework que inclui a superclasse. Nesse caso, é o framework **Foundation** que contém objetos básicos não visuais, dentre eles o **NSObject**.

CARTA.H

Nova notação para importar o framework inteiro, a partir do iOS 7, que é retro compatível, porém pouco usual.

```
9 @import Foundation;  
10  
11 @interface Carta : NSObject  
12  
13  
14  
15  
16  
17  
18  
19  
20 @end
```

CARTA.M

O arquivo de implementação precisa necessariamente importar o arquivo de declaração (header)

```
9  #import "Carta.h"  
10  
11  @implementation Carta  
12  
13  
14  
15  
16  
17  
18  
19  
20  @end  
21
```


CARTA.M

Este bloco é opcional

```
9  #import "Carta.h"
10
11  @interface Carta()
12      //declarações privadas vão aqui
13  @end
14
15  @implementation Carta
16
17  @end
18
```


PROPRIEDADE (CARTA.H)

Não se acessa variável de instância diretamente. Usa-se **@property**, que declara métodos *get* e *set* pelos quais o acesso à variável é feito, tanto dentro quanto fora da classe.

```
11 @interface Carta : NSObject
12
13 @property (strong) NSString *conteudo;
14
```

Esta propriedade em particular é um ponteiro. Especificamente um ponteiro para um objeto do tipo **NSString**.

PROPRIEDADE (CARTA.H)

TODOS os **objetos** ficam na memória heap (ponteiros para lá). Por isso não pode existir uma propriedade do tipo “NSString variavel” (o correto é “NSString *variavel”)

```
11 @interface Carta : NSObject
12
13 @property (strong) NSString *conteudo;
14
```

Como esta propriedade está declarada no arquivo header, ela é pública e pode ser acessada por outros objetos.

PROPRIEDADE (CARTA.H)

```
13 @property (strong) NSString *conteudo;
```

- **strong** faz com que o objeto para o qual este ponteiro aponta seja mantido na memória até que o ponteiro seja setado para **nil**
 - Cuidado: o objeto vai permanecer na memória enquanto houver outros ponteiros **strong** para ele

PROPRIEDADE (CARTA.H)

13

```
@property (strong) NSString *conteudo;
```

- **weak**, em contrapartida, quer dizer que se num dado momento não existirem ponteiros **strong** para o objeto, o sistema operacional pode limpar aquela área de memória e deixar o(s) ponteiro(s) **weak** com valor **nil**.

PROPRIEDADE (CARTA.H)

```
13 @property (strong, nonatomic) NSString *conteudo;
```

- **nonatomic** quer dizer que o acesso a essa propriedade não é “thread-safe”. Para o iOS, sempre é recomendado o **nonatomic** para ponteiros de objeto.
- **atomic** faz justamente o contrário. **Não é recomendado usar no iOS** porque gera código “thread-safe” (só permite acesso à propriedade a uma thread por vez), o que pode até mesmo provocar travamentos na interface de usuário.

PROPRIEDADE (CARTA.M)

```
11 @implementation Carta
12
13 @synthesize conteudo;|
14
15 @end
```

- No arquivo de implementação, basta adicionar a linha destaca à esquerda para que sejam gerados métodos *getter* e *setter* padrão.

PROPRIEDADE (CARTA.M)

```
11 @implementation Carta
12
13 @synthesize conteudo = _conteudo;
14
15 - (NSString *) conteudo {
16     return _conteudo;
17 }
18 - (void) setConteudo:(NSString *) conteudo {
19     _conteudo = conteudo;
20 }
```

Quando se opta por implementar os métodos *get* e *set*, **_conteudo** passa a ser o nome da variável para acesso dentro da classe Carta. Pode ser qualquer outro nome, mas o padrão é que seja o nome da **@property** com “_” como prefixo.

PROPRIEDADE (CARTA.H)

Perceba que aqui não tem **strong** ou **weak**. Tipos primitivos não são armazenados na memória heap, então não é necessário especificar como eles devem ser armazenados nela.

```
1 carta : NSObject
2
3 @property (strong, nonatomic) NSString *conteudo;
4
5 @property (nonatomic) BOOL escolhida;
6 @property (nonatomic) BOOL combinada;
7
8 @end
9
```

Não são
ponteiros!

- Propriedades podem ser de qualquer tipo da linguagem C, incluindo **int**, **float** e até mesmo structs. **BOOL** é um typedef Objective-C, já que o C não tem o tipo boolean.

PROPRIEDADE (CARTA.H)

```
#import <Foundation/Foundation.h>

@interface Carta : NSObject

@property (strong, nonatomic) NSString *conteudo;

@property (nonatomic, getter=isEscolhida) BOOL escolhida;
@property (nonatomic, setter=setarCombinada:) BOOL combinada;

@end
```

No setter é preciso colocar os ":". Isso indica que o método espera um parâmetro do mesmo tipo da propriedade.

MÉTODO (CARTA.H)

```
9 #import <Foundation/Foundation.h>
10
11 @interface Carta : NSObject
12
13 @property (strong, nonatomic) NSString *conteudo;
14
15 @property (nonatomic) BOOL escolhida;
16 @property (nonatomic) BOOL combinada;
17
18 - (int) combinar: (Carta *) outraCarta;
19
20 @end
```

Nome do método

Tipo de dado do argumento.
Perceba que é um ponteiro!

Tipo de dado do retorno

Nome da variável local do
parâmetro

MÉTODO (CARTA.

Sintaxe de implementação do método que foi declarado no arquivo .h

Acessando uma propriedade via getter

```
15 - (int) combinar:(Carta *)outraCarta {  
16  
17     int resultado = 0;  
18  
19     NSString *meuConteudo = self.conteudo;  
20     NSString *conteudoDaOutraCarta = outraCarta.conteudo;  
21  
22     if ([meuConteudo isEqualToString:conteudoDaOutraCarta]) {  
23         resultado = 1;  
24     }  
25  
26  
27  
28 }
```

Esta é uma chamada ao método

```
105 - (BOOL)isEqualToString:(NSString *)aString;  
da classe NSString
```

MÉTODO (CARTA.M)

```
15 - (int) combinar:(NSArray *)outrasCartas {  
16  
17     int resultado = 0;  
18  
19     for (Carta *umaCarta in outrasCartas) {  
20  
21         if ([self.conteudo isEqualToString:umaCarta.conteudo]) {  
22             resultado += 1;  
23         }  
24     }  
25  
26     return resultado;  
27  
28 }
```


CONCEITOS RELACIONADOS AOS MÉTODOS

- Troca de mensagens
 - Origem na linguagem Smalltalk-80
- Exemplo:
 - [**texto** **procurarLetra:** **@ "a"** **aPartirDe:** **INICIO**]
- No exemplo acima:
 - texto – objeto que recebe a mensagem
 - Qual é a mensagem?
 - “procurarLetra:aPartirDe:”