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/Conce ptual/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
 - UlKit
 - 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

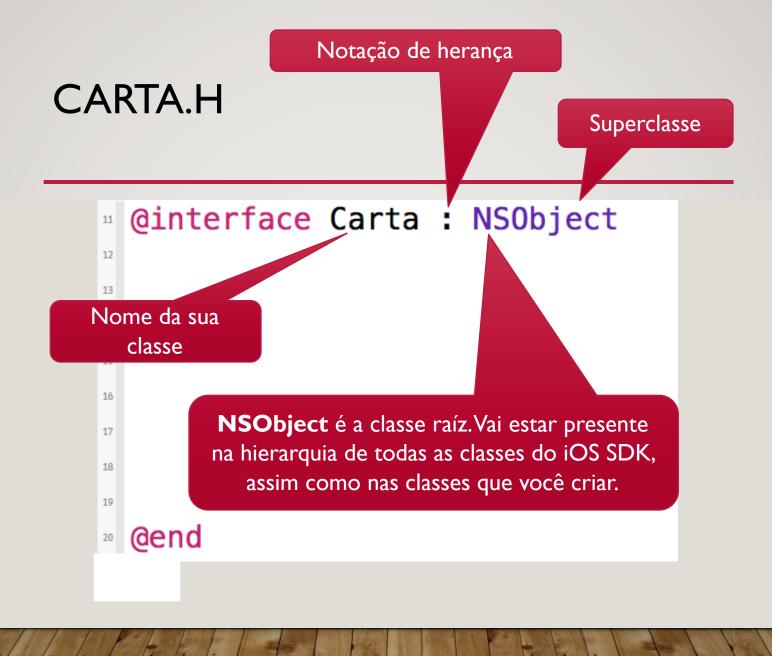
CARTA.H

Declarações públicas

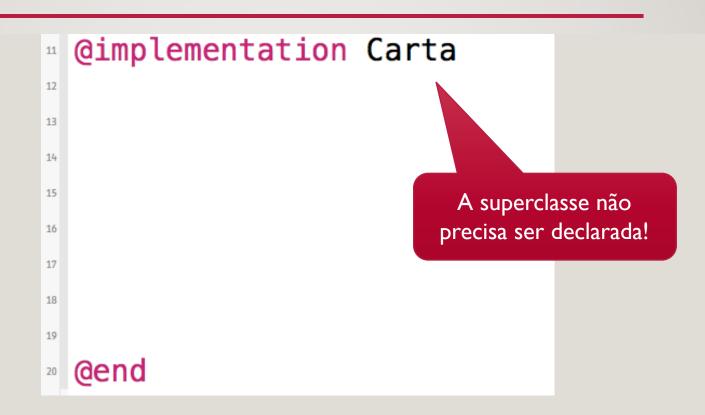
CARTA.M

Implementação privada

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



CARTA.M



CARTA.H

```
#import <Foundation/NSObject.h>
 @interface Carta: NSObject
12
13
15
                     Arquivo header da superclasse
16
17
18
19
 @end
```

CARTA.H

```
#import <Foundation/Foundation.h>
 @interface Carta : NSObject
12
13
14
15
                No iOS, quando a superclasse é uma classe do
16
             sistema, devemos importar o respectivo framework
17
              que inclui a superclasse. Nesse caso, é o framework
18
             Foundation que contém objetos básicos não visuais,
19
                         dentre eles o NSObject.
 @end
```

CARTA.H

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

```
@import Foundation;
 @interface Carta : NSObject
15
 @end
```

CARTA.M

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

CARTA.M

Este bloco é opcional

```
#import "Carta.h"
@interface Carta()
     //declarações privadas vão aqui
13 @end
@implementation Carta
16
 @end
```

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.

```
dinteri Carta: NSObject
deproperty (strong) NSString *conteudo;
deproperty
```

Esta propriedade em particular é um <u>ponteiro</u>. Especificamente um ponteiro para um objeto do tipo **NSString**.

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")

```
dinterface Carta: NSObject
deproperty (strong) NSString *conteudo;
deproperty
```

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

- @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

- @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.

- @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.

```
@implementation Carta

@synthesize conteudo;

@end
```

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

```
dimplementation Carta

description

conteudo

description

conteudo

conteudo

description

conteudo

conteudo

description

conteudo

description

descript
```

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.

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.

```
@proper rong, nonatomic) NSString *conteudo;

@property (nonatomic) BOOL escolhida;
@property (nonatomic) BOOL combinada;

@end

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.

```
#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)

```
Tipo de dado do argumento.

Perceba que é um ponteiro!

Nome do método : NSObject

Perceba que é um ponteiro!

Nome do método : NSObject

Perceba que é um ponteiro!

String *conteudo;

Gproperty onatomic) B0/ escolhida;
Gproperty nonatomic) B L combinada;

- (int) combinar: (Carta *) outraCarta;

Qend
```

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

MÉTODO (CARTA.M)

```
(int) combinar:(NSArray *)outrasCartas {
16
      int resultado = 0;
18
      for (Carta *umaCarta in outrasCartas) {
20
          if ([self.conteudo isEqualToString:umaCarta.conteudo]) {
21
               resultado += 1;
23
24
25
      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:"