

# Classes IOS – Swift

Parte 1

X-Code

Prof. Agesandro Scarpioni



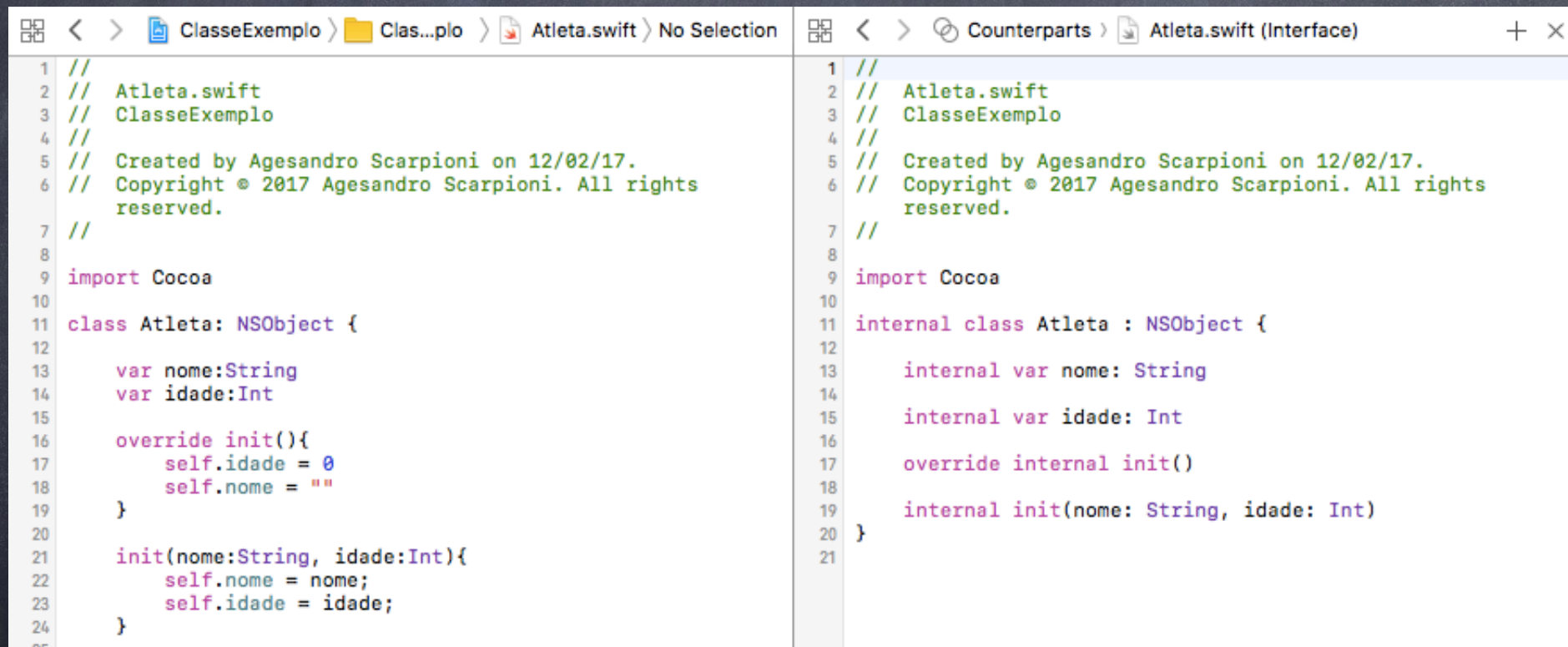
# Classes

- Diferente do Objective-C, as classes em swift possuem apenas um arquivo com a extensão .swift.
- Os arquivos das classes Swift até eram separados como (header file) .h e (messages) .m (2014, 2015, 2016), no entanto não apareciam para o usuário de forma tão explícita como era com o Obj-C.



# Classes

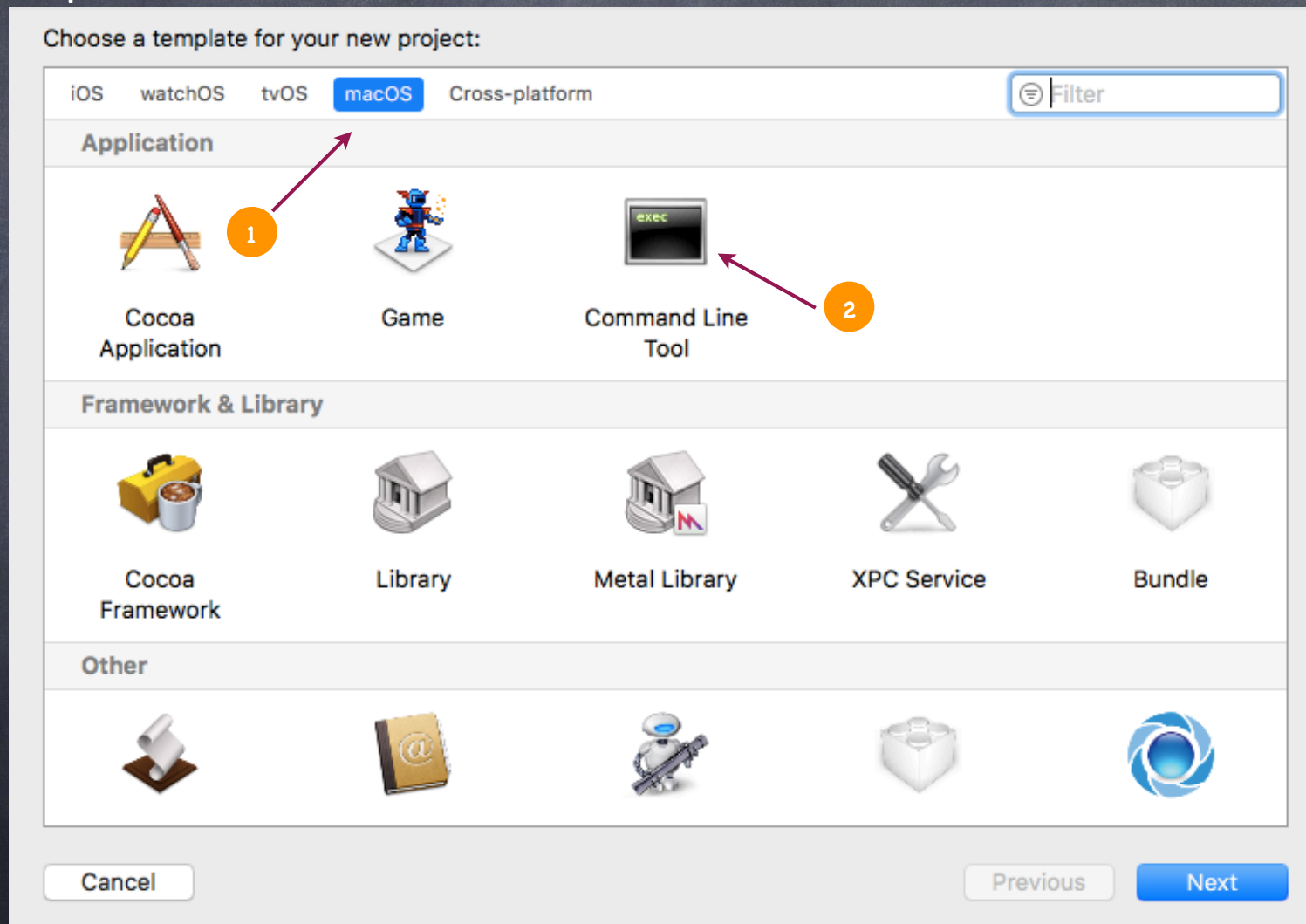
- Atualmente (2017 XCode 8) os arquivos .h e .m no Swift continuam não aparecendo para o usuário, porém são conhecidos como arquivo.swift e arquivos.swift(interface).



Obs: Veja os dois lados da imagem acima, o lado direito é a interface como o antigo .h do Obj-C e o lado esquerdo o antigo arquivo .m.

# X-Code

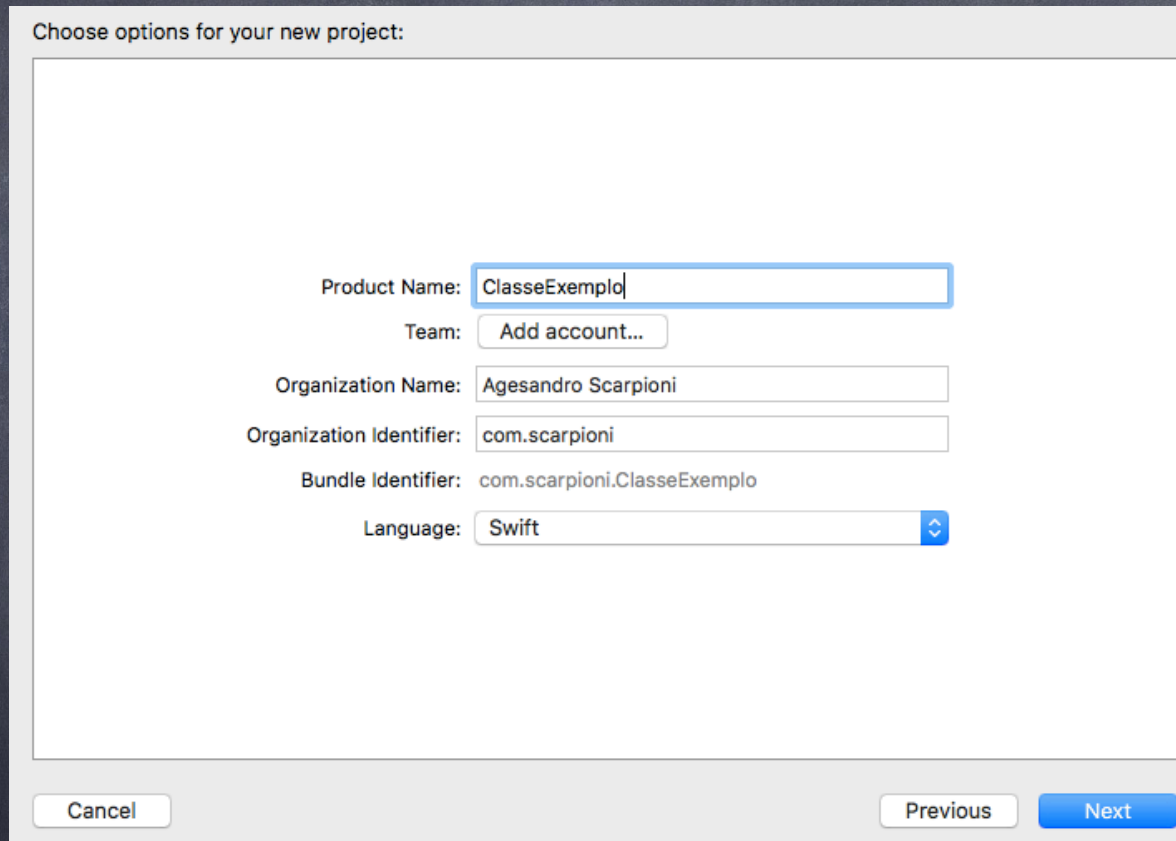
- Abra o Xcode e clique em: File --> New --> Project --> Escolha macOS, depois dê um duplo click em Command Line Tool (2)





# X-Code

- Informe o nome da aplicação em Product Name, preencha o Organization Name com seu nome, escolha Swift na linguagem e clique em Next.



The screenshot shows the 'Choose options for your new project' dialog box in Xcode. The dialog has a title bar and a main content area with several input fields and buttons. The fields are: 'Product Name' with the text 'ClasseExemplo', 'Team' with a button 'Add account...', 'Organization Name' with the text 'Agesandro Scarpioni', 'Organization Identifier' with the text 'com.scarpioni', 'Bundle Identifier' with the text 'com.scarpioni.ClasseExemplo', and 'Language' with a dropdown menu showing 'Swift'. At the bottom, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Choose options for your new project:

Product Name: ClasseExemplo

Team: Add account...

Organization Name: Agesandro Scarpioni

Organization Identifier: com.scarpioni

Bundle Identifier: com.scarpioni.ClasseExemplo

Language: Swift

Cancel Previous Next

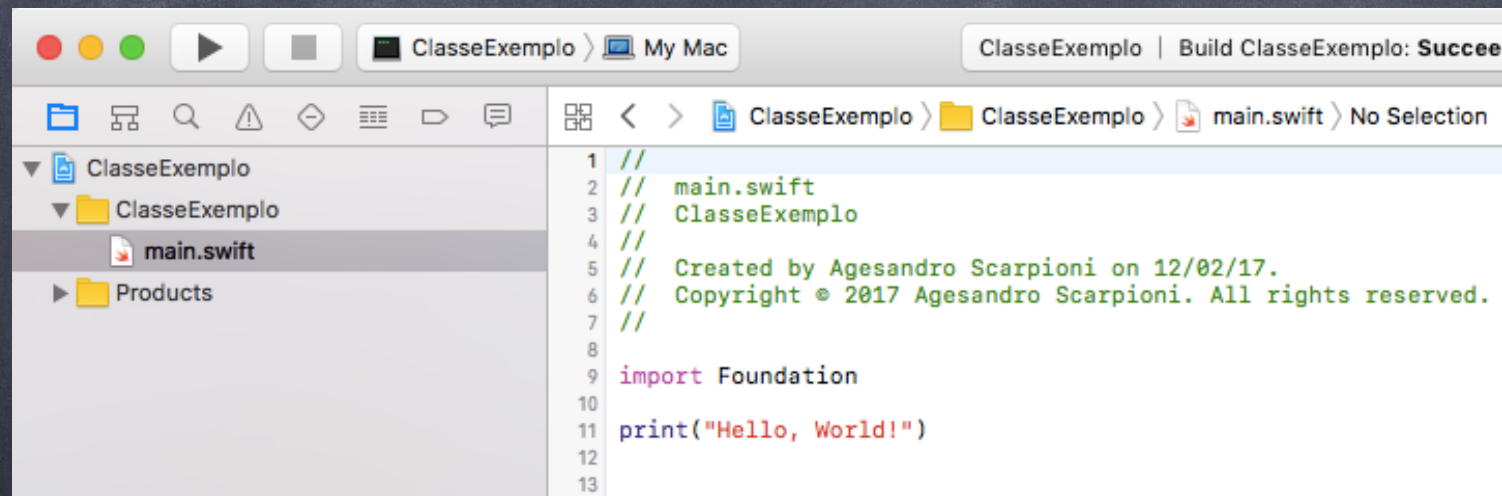
- Na tela seguinte salve na pasta padrão, clicando em Create.

**OBS:** O Organization Identifier é semelhante ao NameSpace no Visual Studio ou o Package no Java, Organization Name informe seu nome ou o nome da empresa.



# X-Code

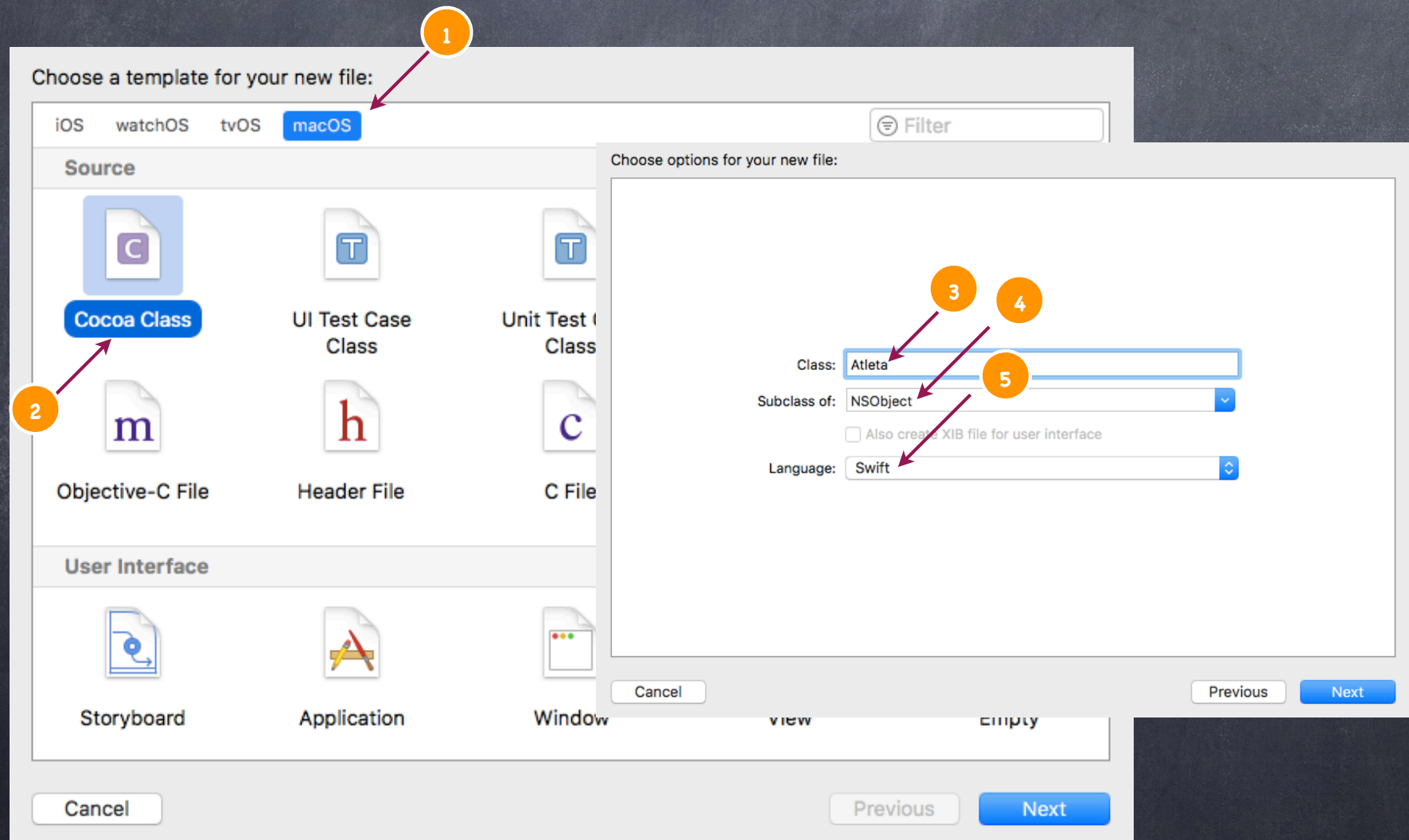
- No canto à esquerda selecione a pasta main abaixo do nome do projeto e do lado direito REMOVA a linha: `NSLog(@"Hello, World!");`
- OK, Vamos por a mão na massa e fazer nossos testes.





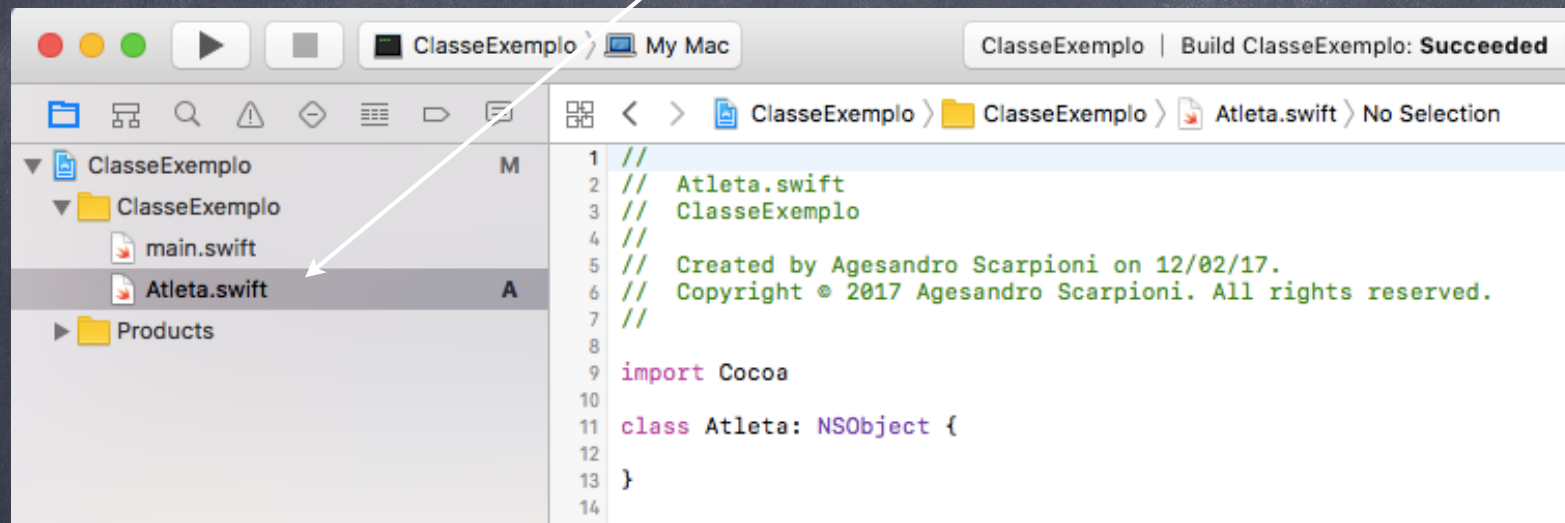
# X-Code

- Os passos abaixo ilustram como devemos criar a classe dentro de nosso projeto. Clicando em File → New File → macOS → double clique em Cocoa Class



# Classes

Arquivo .swift





## Atleta.swift

- É normal após digitarmos as variáveis aparecer a marcação em vermelho, isso ocorre porque não iniciamos as variáveis, que aqui funcionam como atributos getters e setters.

```
1 //  
2 // Atleta.swift  
3 // ClasseExemplo  
4 //  
5 // Created by Agesandro Scarpioni on 12/02/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import Cocoa  
10  
11 class Atleta: NSObject {  
12  
13     var nome:String  
14     var idade:Int  
15  
16 }  
17 |
```

**OBS:** Veja como é declarada uma herança no Obj-C, os ":" após o nome da classe "Atleta" indica que Atleta é filha de NSObject e como no Java, só é possível herdar de uma única classe.



# Construtores

## Atleta.swift

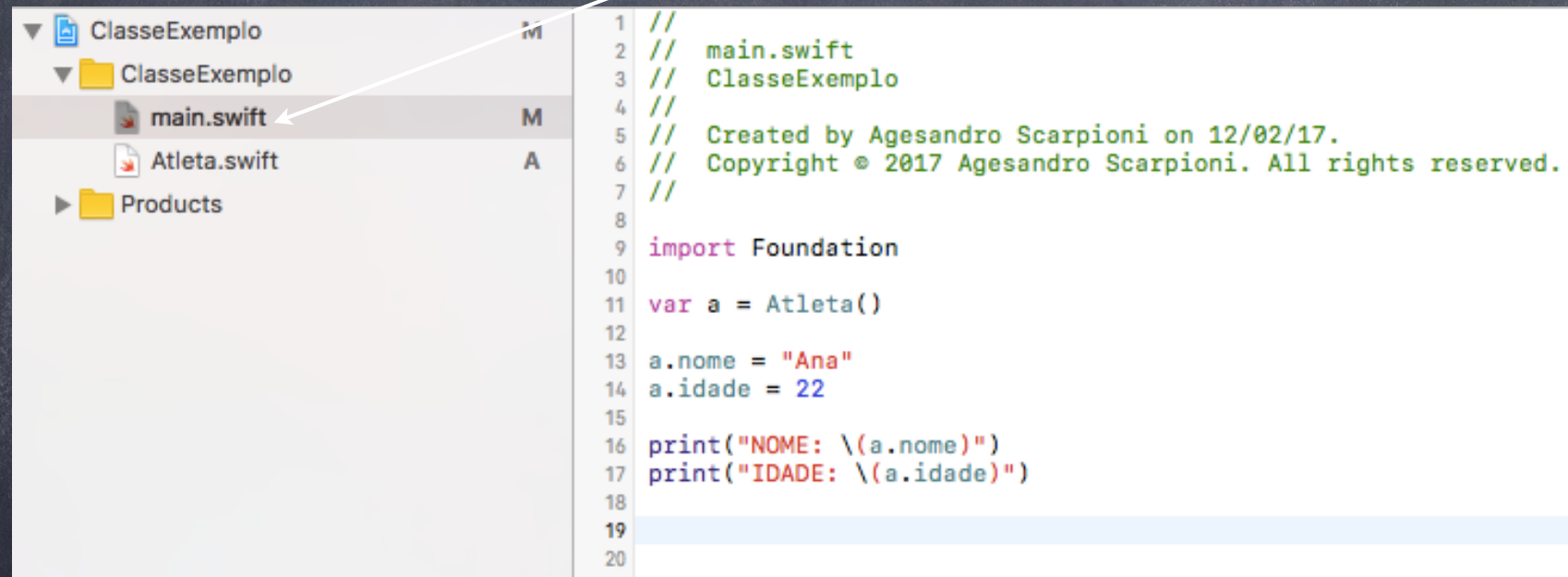
```
1 //  
2 // Atleta.swift  
3 // ClasseExemplo  
4 //  
5 // Created by Agesandro Scarpioni on 12/02/17.  
6 // Copyright © 2017 Agesandro Scarpioni. All rights reserved.  
7 //  
8  
9 import Cocoa  
10  
11 class Atleta: NSObject {  
12     var nome:String  
13     var idade:Int  
14  
15     override init(){ 1  
16         self.idade = 0  
17         self.nome = ""  
18     }  
19  
20     init(nome:String, idade:Int){ 2  
21         self.nome = nome;  
22         self.idade = idade;  
23     }  
24 }  
25  
26 }  
27
```

**OBS:** No exemplo acima temos dois construtores, um padrão (1) , outro com parâmetros (2).



# Classes – Atributos

## Implementando o Main



**OBS:** Foi criado um objeto *a*, instancia da classe *Atleta*, foi carregado alguns dados nos atributos (set), e em seguida foi feito um pequeno teste para exibir esses atributos (get).



# Classes – Atributos

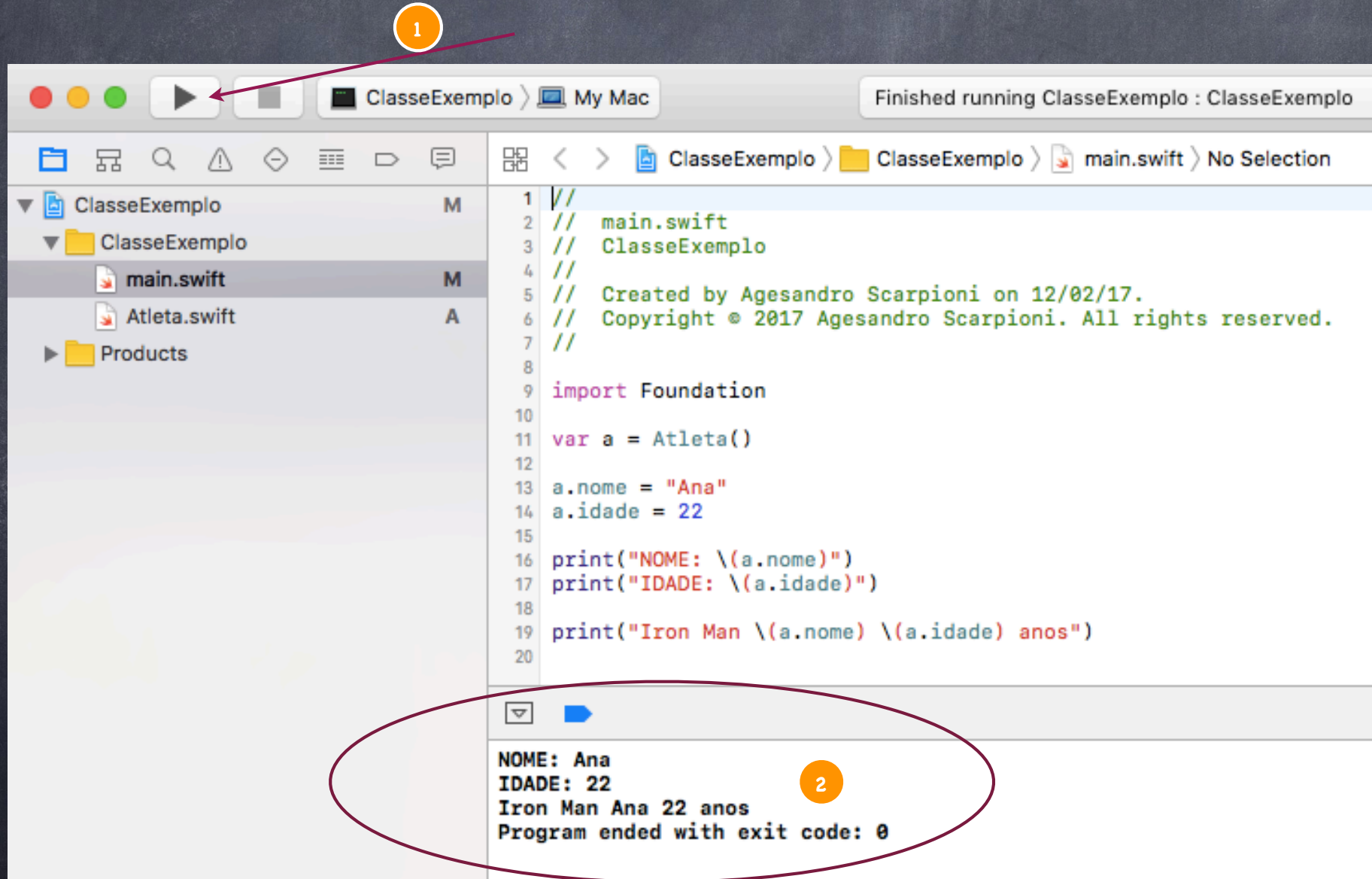
## Implementando o Main

```
1 //  
2 //  main.swift  
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  
10  
11 var a = Atleta()  
12  
13 a.nome = "Ana"  
14 a.idade = 22  
15  
16 print("NOME: \(a.nome)")  
17 print("IDADE: \(a.idade)")  
18  
19 print("Iron Man \(a.nome) \(a.idade) anos")  
20
```



# Classes – Atributos

Executando – Command + R





# Construtor

- Na linha (21) foi criado o objeto a2 com o construtor com parâmetros, logo em seguida foi dado o display para exibir os dados.

```
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
10
11 var a = Atleta()
12
13 a.nome = "Ana"
14 a.idade = 22
15
16 print("NOME: \(a.nome)")
17 print("IDADE: \(a.idade)")
18
19 print("Iron Man \(a.nome) \(a.idade) anos")
20
21 var a2 = Atleta(nome: "Carlos Gomes", idade: 25)
22 print("Iron Man \(a2.nome) \(a2.idade) anos")
23
24
25
```

NOME: Ana  
IDADE: 22  
Iron Man Ana 22 anos  
Iron Man Carlos Gomes 25 anos  
Program ended with exit code: 0



# Classes – Atributos

## Informações importantes

- O gerenciamento de memória dos objetos no Obj-C e no Swift é chamado de contador de referências (Reference Counting), o método `release` é o destrutor do objeto e sempre é chamado quando o contador de referência do objeto chega em zero. Nós não o chamamos porque iniciamos o projeto com este gerenciamento automático (Desde o Xcode5 isto é padrão), não precisamos mais nos preocupar em liberar o objeto da memória. Caso contrário seria necessário chamar o método destrutor para o objeto criado. Em nossa classe `Atleta` seria: `[a release];`.



# Classes – Atributos

## Linha do Tempo

- Se você desabilitar o gerenciamento automático de memória, sempre deverá chamar o método release quando chamar o alloc para criar um objeto, não vamos nos preocupar com isso no Xcode 6 vamos deixar esse gerenciamento automático, hoje ele é Default para a forma automática.
- No Xcode 4 você tinha que selecionar o gerenciamento automático de memória marcando um checkbox com a frase "Use Automatic Reference Counting)", já no Xcode 5 ou posterior o ARC já vem habilitado.



# Classes – Atributos

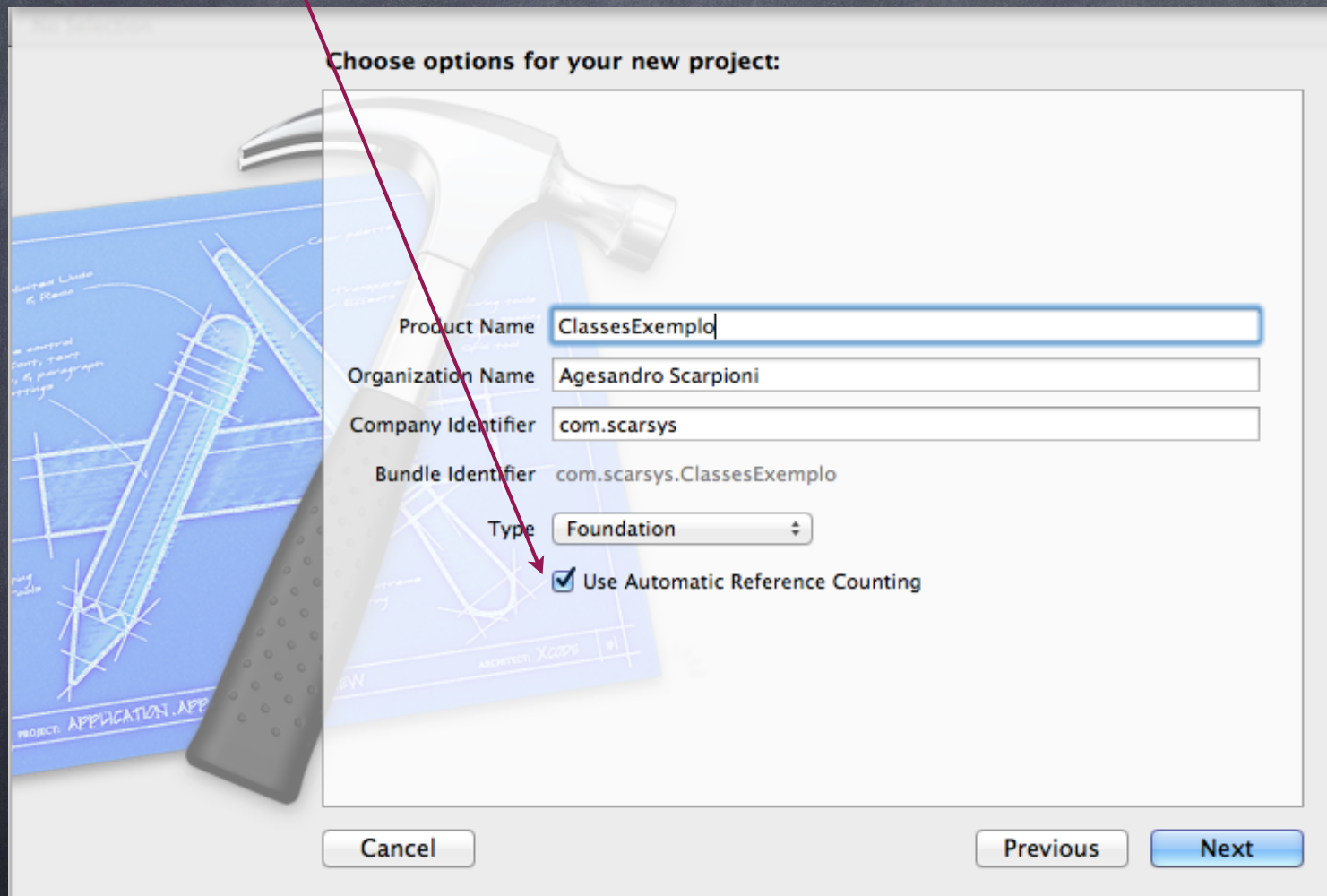
## Linha do Tempo

- Nas versões mais velhas, ou seja anteriores ao Xcode 4, nós tínhamos que gerenciar a memória de forma manual.
- Na versão 4 do Xcode nós tínhamos que definir que iríamos trabalhar com o gerenciamento de memória automática (veja o checkbox marcado no próximo slide).
- Desde a versão 5 ou posterior do Xcode o gerenciamento automático de memória está habilitado por padrão.



# Curiosidade

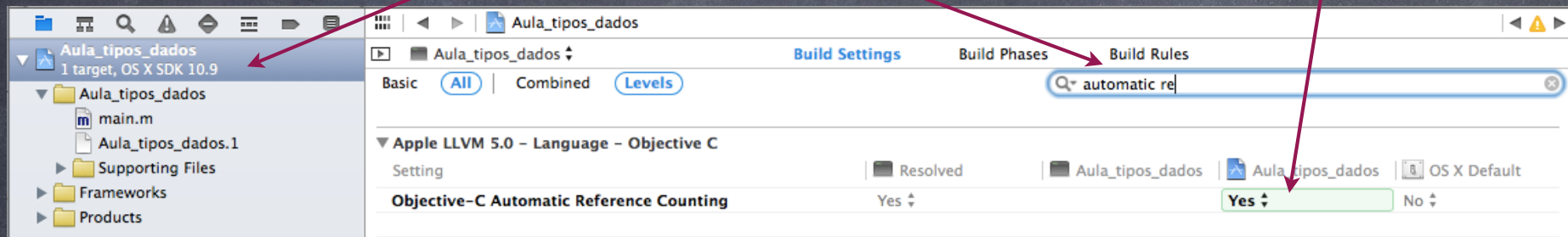
Como era o check box para utilizar o ARC no Xcode 4.



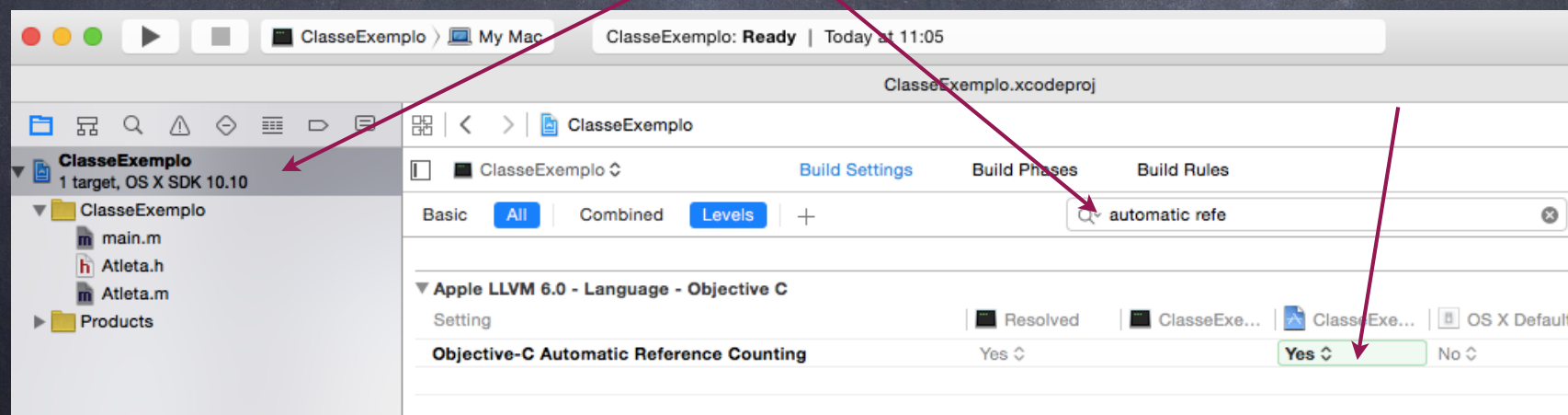


# Classes

Onde ficava o ARC no Xcode 5



Onde fica o ARC no Xcode 6 ou posterior





# Prática

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

- Crie um projeto novo e crie uma classe chamada Enfermeira.
- Esta classe deve possuir atributos do tipo String, float, bool e int.
- No Main instancie a classe, passe algumas informações para os atributos do objeto e exiba o resultado com NSLog, se você quiser ao invés de chamar os getters na linha do NSLog você pode criar novas variáveis, passar os getters para as variáveis e usá-las no NSLog.



# Próxima aula

- Métodos do tipo void e function com parâmetros, métodos de instância e métodos de classe.