# Livro do Educador

## Desenvolvimento de Aplicativos Móveis

### iOS

# Unidade 2

## Aula 1 – Swift – Parte 1

Educador, nesta unidade, o intuito é apresentar o básico da linguagem Swift ao aluno. O aluno aprenderá o que é a linguagem Swift, suas características, alguns tipos básicos, arrays, Controle de fluxo, condicionais, loops, métodos e tudo mais num primeiro momento, o que pode ser traçado um paralelo com a linguagem Java. Num segundo momento, serão apresentadas algumas peculiaridades da linguagem Swift, como o Initializer, Casting, Structs, Protocol, Closures e algumas funções mais avançadas.

Esta aula é muito importante para o decorrer do curso, é essencial que o aluno conheça todo o conteúdo.

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Descobrir a linguagem Swift, seus usos e características;
* Aprender sobre a sintaxe básica da linguagem Swift;
* Fixar os conceitos de Tipos Básicos, Arrays, Controle de Fluxo, Condicionais, Loops, Métodos, e mais.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma e apresentação da TDP.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* O que é Swift?;
* Tipos Básicos;
* Controle de Fluxo;
* Funções e Métodos.

### PONTOS IMPORTANTES

Esta aula é muito semelhante as primeiras aulas de POO, por isso é importante que se faça uma associação da linguagem Swift com a Java, pois será mais fácil a assimilação do aluno.

### 1.1 O que é Swift?

Educador, iremos iniciar as explicações sobre a linguagem Swift, começando por diferenciá-la de outras linguagens de programação. No curso o aluno verá que a linguagem Swift é uma linguagem bastante moderna e eficiente, pois ela possui o melhor das linguagens baseadas em C, muito mais aprimorada, objetiva e rápida. Ela é uma linguagem muito fácil de escrever e interpretar e muito mais estável e segura. A linguagem Swift é muito moderna e ainda está em constante aprimoramento, hoje, temos a versão Swift 3 como a mais avançada.

#### Características

Educador, algumas características interessantes da linguagem Swift devem ser apresentadas ao aluno, como os “tipos” que tornam o código mais limpo e menos propenso a erros, também há modos de eliminar cabeçalhos e fornecer “namespaces”, que são análogos aos pacotes Java, ou seja, é um diretório que separa um conjunto comum de classes do projeto. Outra característica que facilita a digitação e diminui a incidência de erros, é a não necessidade de inserção de ponto e vírgula “ ; ” no final dos códigos. A linguagem Swift também traz bastante parâmetros de outras linguagens C, só que numa sintaxe mais limpa e objetiva.

No decorrer do curso veremos outras características. Agora, auxilie o aluno a abrir um Playground e siga os exemplos seguintes.

### 1.2 – Tipos Básicos

Educador, explique a diferença na digitação entre constante e variável. Explique também que não é necessário definir o tipo de variável ou constante, pois é definida por padrão como int, e ao definir um tipo apara uma variável ou constante, não é possível mais alterá-lo. Uma dica é usar o atalho “Option + click” sobre o nome de uma constante ou variável, para ver seu tipo inferido.

Mostre a forma de conversão explícita de valores, como por exemplo de “int” para “String”.

Ou seja:

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| var explicitDouble: Double = 70 | Indica que um valor é variável |
| let explicitDouble: Double = 70 | Indica que um valor é constante |
| let explicitDouble: Double = 70 | Definição explicita do tipo da constante como Double |
| let width = 94  let widthLabel = “Any text” + String(width) | Conversão da constate width em int para String |
| let apples = 3  let appleSummary = "I have \(apples) apples." | Conversão da constante apples em int para String utilizando “\(cte)” |

#### Opcionais

Eduacador, defina a função dos optionals como marcadores à serem usados para trabalhar com valores que podem estar ausentes e tentativas de conversões, utilizando “?” para marcar um valor como “optional” e “nil” para indicar o valor como ausente.

Explique também que em Swift, os valores de uma constante/variável podem ser embrulhados e desembrulhados através de código especial.

Ou seja:

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| let optionalInt: Int? = 9 | Marca um valor como opcional |
| let optionalInt: Int? = nil | Indica que o valor está ausente |
| var actualInt: Int = optionalInt! | Utilizado para desembrulhar uma variável |

* + 1. **Arrays**

Também conhecido como matriz, é um tipo de dado que mantém uma coleção ordenada de itens. Podem ser criados utilizando [] e também podem ser vazios.

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| var ratingList = ["Poor", "Fine", "Good", "Excellent"] | Uma lista ordenada de dados entre [] (Array). |
| // Cria um array vazio | Comentário de uma só linha |
| /\* ... \*/ | Comentário de várias linhas |

### 1.3 – Controle de fluxo

Educador, daqui em diante, você irá ministrar sobre indicações condicionais e Loops.

#### 1.3.1 – If, else

Educador, como em POO, “if” verifica se uma condição é verdadeira, se for, seu pedaço de código será executado, se não, o seu pedaço de código será ignorado. Já o else é utilizado para atribuir uma ação caso a condição retorne falsa. O else if é utilizada para avaliarmos mais condições em casos mais complexos. Na linguagem Swift, os condicionais têm sua estrutura com pequenas diferenças da linguagem Java, vide exemplos.

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| let number = 23  if number < 10 {} | Cláusula que verifica se uma condição é verificada |
| let number = 23  if number < 10 {} | Condição a verificar |
| let number = 23  if number < 10 {} | Elemento a ser verificado, no caso |
| let number = 23  if number < 10 { comando } | Código a ser executado se a condição verdadeira |
| let number = 23  if number < 10 { }  else if number > 100 { comando } | Encadeamento de condições a ser avaliada antes de ser executado o código para a condição falsa |
| let number = 23  if number < 10 { }  else if number > 100 { }  else {  comando } | Comando executado caso as condições retornarem falsas |

#### 1.3.2 – If - let

If-let é usado em uma declaração if para verificar se um optional contém um valor ou é nil (vazia).

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| var optionalName: String? = "John Appleseed"  if let name = optionalName {    "\(name)"  } | Se a var **optionalName** não é vazia, atribuir e desembrulhar o valor desta, à constante **name** e converter esta, para **String** |

#### 1.3.3 – Where

Where é usado para aumentar o escopo da instrução condicional de if. Ou seja, if é executado somente se a condição de where for satisfeita também.

#### 1.3.4 – Switch

Educador, este é um elemento muito importante na linguagem em questão pos suporta uma grande variedade de comparação e qualquer tipo de dados.

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| let vegetable = "red pepper"  switch vegetable { | Função **switch** que comparará o valor da constante **vegetable** |
| case "celery":    let vegetableComment = "Add some raisins and make ants on a log." | **Case** indica se caso a variável atribuída anteriormente na declaração **switch** tiver valor igual ao valor descrito (no caso, **celery**), realizar a ação em seu escopo |
| default:    let vegetableComment = "Everything tastes good in soup."  } | O caso **defaut** deve ser instanciado para atribuir uma ação para caso nenhum dos **cases** sejam satisfeitos |

#### 1.3.5 – Loops

No operador de repetição for em Swift, vemos algumas diferenças com Java:

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| for i in 0..<4 { } | Realiza uma série de repetições sem incluir o índice superior. No caso, o operador irá repetir do índice 0 até o 3, num total de 4 repetições |
| for \_ in 0...4 { } | Realiza uma série de repetições incluindo o índice superior, no caso 5 repetições |

### 1.4 – Funções e Métodos

Educador, a princípio relembre ao aluno algumas definições.

#### 1.4.1. – Funções

Uma função é uma peça reutilizável, com um nome que pode ser referido a partir de muitos lugares em um programa.

Em Swift é representada por func seguido de uma lista de argumentos e não estão diretamente relacionadas a classes ou objetos, que faz com que sejam acessadas de qualquer objeto. Depois de chamar uma função, deve-se passar cada valor subsequente com seu nome.

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| func greet(name: String, day: String) -> String {    return "Hello \(name), today is \(day)."  } | Função greet com parâmetros name e day, do tipo String que retorna um objeto do tipo String. |
| greet(name: "Bob", day: "Friday") | Atribuições de valores aos parâmetros da função greet |

#### 1.4.2. – Métodos

São funções que são definidas dentro de uma classe específica e são explicitamente vinculados a ela.

|  |  |
| --- | --- |
| **Nomenclatura** | **Descrição** |
| let exampleString = "hello"  if exampleString**.**hasSuffix("lo") {    print("ends in lo")  } | O ponto ( . ) é a nomenclatura utilizada para chamar um método |
| let exampleString = "hello"  if exampleString.hasSuffix("lo") {    print("ends in lo")  } | Representação do método ‘temComoSufixo(“sufixo”)’, em tradução livre |

Há diversas variedades para se instanciar um método, de acordo com sua especificidade, neste item é mostrada algumas delas mas no decorrer do curso serão melhor abordadas estas e outras variedades de implementação de métodos.

### 1.5 – Resumo

Nesta aula o aluno aprendeu algumas formas básicas da linguagem Swift e já familiares proveniente da bagagem que o aluno obteve nos cursos anteriores. O intuito foi mostrar as diferenças e novidades vistos nos **tipos básicos** das variáveis e constantes, variáveis **opcionais** e vazias, revendo arrays e sua forma de implementação na linguagem Swift, assim como **condicionais** e **Loops**, partindo de suas particularidades na linguagem Swift. O aluno conheceu também o condicional complementar **Where**, que aumenta o escopo da instrução condicional do **if**, revisou conceitos e aprendeu algumas novidades sobre **switch**, **funções** e **métodos**.

#### 

## Aula 2

### 2. Swift – Parte 2

Agora entraremos um pouco mais a fundo na linguagem Swift, mostrando um pouco sobre os i**nitializers** e as **classes**, mostrando algumas novas propriedades. Também o aluno irá aprender um pouco mais sobre a **conversão de tipos** e **enumeração**, que define um grupo de valores relacionados, as **estruturas** e suas diferenças com as classes. Aprenderá também sobre **protocolos**, que pode descrever como uma implementação deve se assemelhar. Aprofundaremos um pouco mais em **Funções** e **Closurers**.

### CARGA HORÁRIA

Conforme o plano de aula, esta aula terá duração de 1h30 e deverá ser conduzida de acordo com as orientações pedagógicas.

### OBJETIVO DA AULA

Ao final da aula, você deverá garantir que o aluno tenha subsídios para

* Definir uma classe e suas propriedades;
* Utilizar o método Initializer, failable initializer e seus modificadores;
* Realizar casting e downcasting para tratamento de superclasses ou subclasses;
* Criar enumerações e associá-las entre si;
* Utilização de estruturas para dados que não precisam ter capacidades de herança e casting;
* Utilizar protocolos para descrever como uma implementação deve se assemelhar.

### ORIENTAÇÕES PEDAGÓGICAS

Para atender os objetivos de aprendizagem, você deverá conduzir o processo de ensino considerando a organização didática apresentada a seguir:

* 40 minutos de aula expositiva;
* 10 minutos para tirar as dúvidas dos alunos;
* 40 minutos para desenvolver as atividades propostas para a turma e apresentação da TDP.

### TÓPICOS DE ESTUDO

Todos os tópicos a seguir, conforme livro do aluno, devem ser trabalhados de forma dinâmica, criativa, com embasamento teórico e prático voltado ao mercado de trabalho.

* Classes e Initializers;
* Casting;
* Enumerações e estruturas;
* Protocolos.

### PONTOS IMPORTANTES

Esta aula, também faz parte da base da linguagem Swift, e é importante que o aluno tire todas as suas dúvidas para que ele não somente saiba programar, mas entenda toda a sintaxe do que se está programando.

### 2.1. Classes e Initializers

Neste início, o aluno deverá associar seus conceitos de classes aprendidos em POO com o que será apresentado aqui. Revise tais conceitos, inclusive sobre objetos e métodos e de continuidade ao conteúdo.

#### 2.1.1. Classes

Educador, aqui é importante que o aluno aprenda como são instanciadas as classes e suas propriedades na linguagem Swift. Escrevendo class nomeClasse { } se declara uma classe, e suas propriedades são declaradas como visto anterior mente com as constantes e variáveis. Seus métodos seguem a mesma forma de declaração de funções vista na aula passada. Para instanciar uma classe basta adicionar () após o nome da mesma: nomeClasse(). Suas propriedades e métodos estarão restrito apenas à classe.

#### 2.1.2. Initializers

Já os **initializers** se assemelham aos métodos construtores em Java. Ele é declarado com a palavra init.

* O Initializer prepara uma instancia de uma classe para uso, envolvendo a definição de valores iniciais de cada propriedade e outras configurações iniciais do objeto.

Podemos chamar os inicializadores colocando parênteses após o nome da classe e incluir todos os argumentos e nomes, juntamente com seus valores.

Educador, aqui podemos revisar o conceito de hierarquia de classes e em seguida como declaramos uma subclasse. Em Swift, escrevemos uma subclasse incluindo o nome da superclasse depois de seu nome separado por “ : ” .

Uma classe pode herdar apenas uma superclasse, e os métodos dessa sub classe pode sobrepor a implementação da superclasse, para isso é necessário usar a palavra override antes do método.

Em casos que solicitamos valores dentro de um intervalo especifico e os valores fornecidos estão fora deste intervalo, não se deve inicializar certo objeto, ou seja, a inicialização precisa falhar. Para isso precisamos criar um falso inicializador, e para isso utilizamos init? para declará-lo.

Também podemos encontrar diversos modificadores, estes mudam o comportamento inicial de acordo com cada tipo. Educador, explique sobre o padrão designado, o modificador convenience e o required.

### 2.2. Casting

Educador, explique ao aluno que o casting, ou conversão de tipo, verifica o tipo de uma instancia e a trata como uma sub ou superclasse de sua própria hierarquia de classes. Também explique o que é downcast, referindo-se a uma subclasse e os operadores de conversões opcionais e forçadas, as? e as!, este utimo faz com que o downcasting do objeto seja forçado. Explique outras possibilidades de downcast e utilize os exemplos do livro. Lembrando que este é um conceito introdutório que será posto em prática mais a frente, mas não deixe com que reste dúvidas aos alunos.

### 2.3. Enumerações e estruturas

Classes não são as únicas maneiras de definir tipos de dados em Swift. Enumerações e estruturas têm capacidades semelhantes às classes, mas podem ser uteis em diferentes contextos.

#### 2.3.1. Enumerações

Educador, defina enumerações como uma forma de definir um tipo comum de um grupo de valores relacionados que nos permite trabalhar com esses valores de maneira mais segura. Mostre a forma de implementação de uma enumeração usando enum e a necessidade de explicitar um tipo para ela. Por fim, explique como as enumerações são usadas de maneira mais segura, como no exemplo, em referências abreviadas.

#### 2.3.2. Structs

Educador, explique o que são estruturas e a forma de instanciá-las, mas principalmente a diferença que esta possui das classes, pois as classes são referenciadas, já as estruturas são copiadas quando definimos tipos de dados que não precisam ter capacidades de herança e casting.

### 2.4. Protocolos

Protocolos descrevem como uma implementação deverá se assemelhar. Ele é adotado para fornecer uma implementação efetiva, fazendo com que qualquer que satisfaça seus requisitos estejam em conformidade.

Protocolos são análogos às interfaces em Java e podem ser declarados inserindo a palavra protocol antes do objeto.

Educador, dê exemplos e fale da exigência que os protocolos tem sobre os tipos. Mostre também como as classes, estruturas e enumerações podem adotar um protocolo.

### 2.5. Funções (avançado) e Closures

Educador, vendo um pouco mais de Funções, explique como escrever um rótulo personalizado. Mostre exemplos ao aluno de outras utilidades para as funções como usar uma tupla para definir um valor composto, que possibilita retornar vários valores de uma vez só numa mesma função.

Algumas características das funções é que elas podem ser aninhadas, que permite acesso a variáveis externas e podem também ter um número variável de argumentos e ainda podem retornar outra função como o seu valor e pode levar outra função como um dos seus argumentos.

Por fim, explique o que são closures e o envolvimento das funções com estes.

### 2.6. Resumo

Nesta aula, o aluno reviu e reforçou conceitos de classes, foi apresentado aos Initializers e Casting, conceitos fundamentais e recorrentes na linguagem Swift 3. Também aprendeu sobre enumerações, estruturas e protocolos, e por fim se aprofundou um pouco mais nos conceitos das funções. Sendo assim, o aluno tem agora uma base para leitura e compreensão da linguagem Swift e está preparado para que nas próximas aulas já possa utilizar o Xcode para programar.