SWIFT 101

BASICO

- Linguagem Fortemente tipada
- Inferência de tipos
- Case sensitive
- Sem necessidade do ;
- “Não há” valores nil (variáveis sem valor)

TIPOS DE DADOS BÁSICOS

- String - “Ola mundo @ 2018”
- Int - 123456789
- Double 1.1234 (Precisão de 15 a 16 casas decimais)
- Float 1.2345 (Precisão de 7 casas decimais)
- Boolean - true ou false

VARIÁVEIS E CONSTANTES

- "caixa de sapatos" onde guardamos valores
  - Identificador de uma posição em memória onde os dados são armazenados
- Variáveis - o seu conteúdo pode mudar
- Constantes - depois de inicializadas o seu valor não pode mudar

DECLARAÇÃO – VARIÁVEIS E CONSTANTES

- Inferencia de tipo

```
var/let nome = valor

var numero = 10

let numero = true
```

- Tipo Explicito

```
var nome:tipo = valor

var numero:Int = 10

let numero:Int = 10
```

CASTING – VARIÁVEIS E CONSTANTES

- Conversão de tipos

```
NovoTipo(nomeVar)

var x = 10
var z = String(x)
```

OPERAÇÕES COM VARIÁVEIS E CONSTANTES

- Apenas são possíveis operações com variáveis do mesmo tipo
- Operações em Swift
  - +
  - -
  - /
  - *
  - %

OPERAÇÕES COM VARIÁVEIS E CONSTANTES

- Incremento
- ++ /- - não disponível
- += 1
- -= 1
- *= 1

TUPLOS

- Variáveis que podem conter mais que um elemento
- Inferencia de tipo

```
var v1 = (1, “Dois”, true)
```

```
let v1 = (id:1, nome:“Dois”, inscrito:true)
```

- tipo Explicito

```
let v1:(Int, String, Bool) = (1, “Dois”, true)

var v1 (id:Int, nome:String, inscrito:Bool) = (id:1, nome:“Dois”,
inscrito:true)
```

STRINGS

ESTRUTURAS DE CONTROLE

- if
- switch / case

ESTRUTURAS DE CONTROLE – IF

- if

```
if condition {
    code
}
```

- switch / case

```
switch Value {
case pattern:
    code
default:
    code
}
```

ESTRUTURAS DE REPETIÇÃO

- for
- while
- do/while

ESTRUTURAS DE REPETIÇÃO

- for

```
for a in collection{
  //do something
}
```

- while

```
while true{
  //do something
}
```

OPCIONAIS

- Variáveis que podem ou não ter valor
- Forma de atribuir valor *nil*
- Para serem utilizados tem de ser *unwrapped*

OPCIONAIS

- Criar opcionais

```
var a:Int?  // nil

var a:Int? = 10 // Optional(10)
```

```
var a:Int?
a = 10 // Optional(10)
```

OPCIONAIS

- Operações com opcionais

```
var a:Int?
a = 10 // Optional(10)
a = nil // nil
```

```
var a:Int
a = 10 // 10
a = nil // Erro
```

OPCIONAIS

- *unwrap* opcionais

```
var a:Int? = 10 // Optional(10)

a! // 10

a // Optional(10)
```

OPCIONAIS

- ! vs ?

OPCIONAIS

- *unwrap* opcionais

```
var a:Int? = 10 // Optional(10)
```

```
if let x = a {
  print(x)  // 10
}
```

OPCIONAIS

- *unwrap* opcionais

```
var a:Int? = 10 // Optional(10)

var x = a ?? 0  //10
```

```
var a:Int? // nil

var x = a ?? 0  // 0
```

OPCIONAIS

- *unwrap* opcionais

```
var a:Int? = 10 // Optional(10)

guard let t = a else { return }
```

COLLECTIONS

- Array
- Set
- Dicionários

COLLECTIONS - ARRAYS

- Conjunto ordenado
- Os valores são acedidos através do seu index
- Criação de arrays

```
let arr:[Int]
```

- Criação de arrays

```
arr = []
arr = [1,2,3,4,5,6]
```

COLLECTIONS – ARRAYS

- Adicionar elementos

```
arr.append(10)
```

- Remover elementos

```
var x = arr.popLast()

var x = arr.remove(at: 0)
```

- Aceder a um elemento

```
x[0]
```

COLLECTIONS - ARRAYS

- Listar os valores do array

```
for a in arr{
    print(a)
}

arr.forEach{ a in
    print(a)
}
```

COLLECTIONS – DICIONÁRIOS

- Conjunto Chave - Valor
- Não ordenado
- Os valores são acedidos através da seu chave
- Criação de Dicionários

```
let dict:[String:String]
```

- Instanciação de Dicionários

```
let dict = [:]
let dict = [“nome”:”Gonçalo”, “lastName”:”Feliciano”]
```

COLLECTIONS – DICIONÁRIOS

- Ler elementos

```
dict[key]
```

- Adicionar elementos

```
dict[New key] = new value
```

- Remover elementos

```
dict[key] = nil
```

COLLECTIONS – DICIONÁRIOS

- Listar os valores do array

```
for a in dic{
    print(a)
  }

dic.forEach{ a in
    print(a)
  }
```

COLLECTIONS - SETS

- Conjuntos de Valores
- Não ordenado
- Não contem valores repetidos
- Criação de Sets

```
let set:Set<String>
```

- Instanciação de Sets

```
let set = Set<String>()
let arr:Set = [”ovos”, “açúcar”, “farinha”]
```

COLLECTIONS - SETS

- Iterar pelo set

```
for a in set{
    print(a)
}
```

COLLECTIONS – SETS

- Adicionar elementos

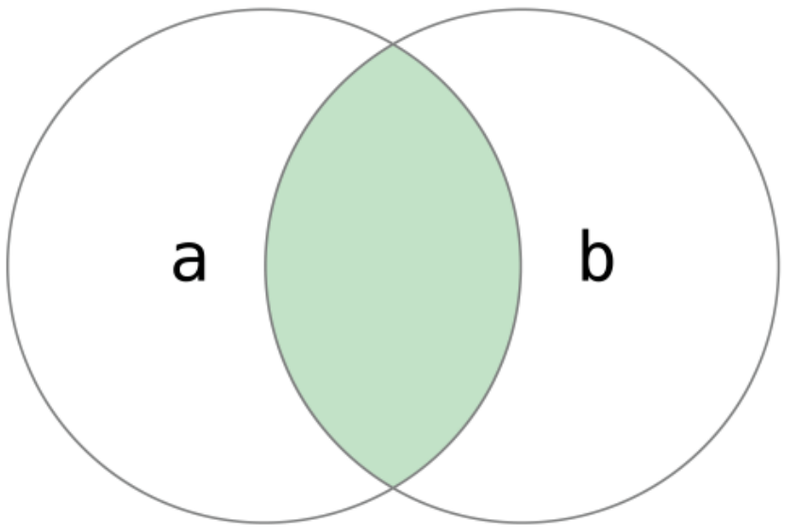
```
set.insert(“mel")
```

- Remover elementos

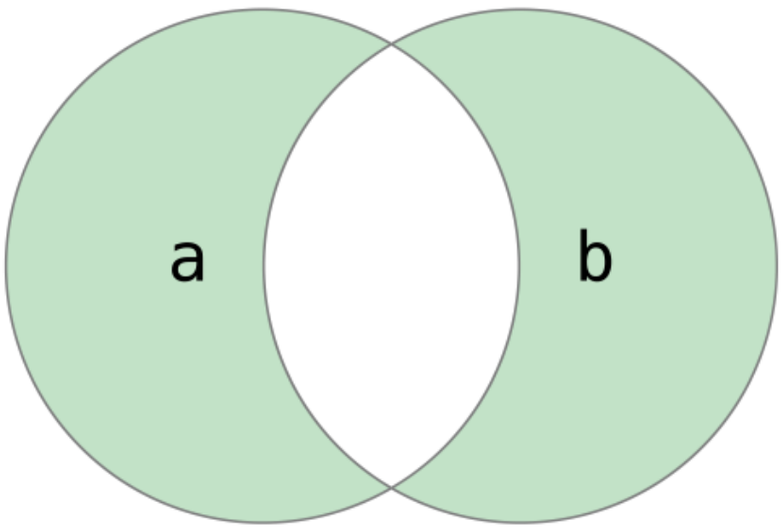
```
var x = set.remove(“ovos”)
```

COLLECTIONS – SETS

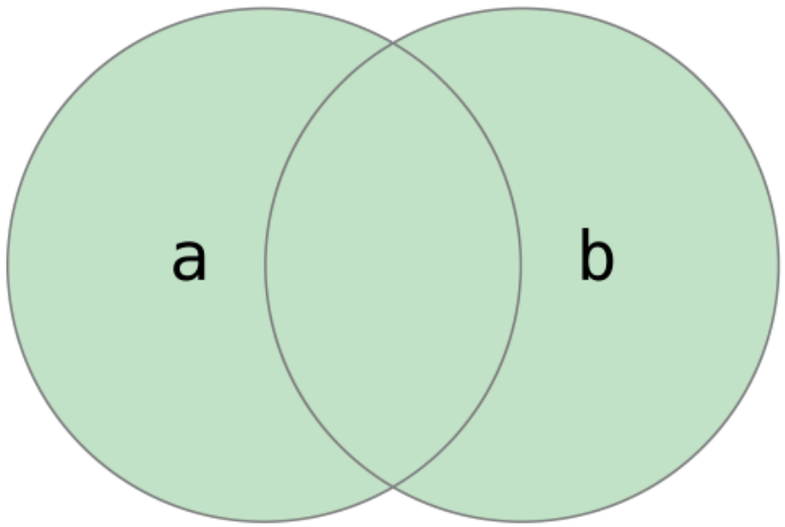
a.intersection(b)



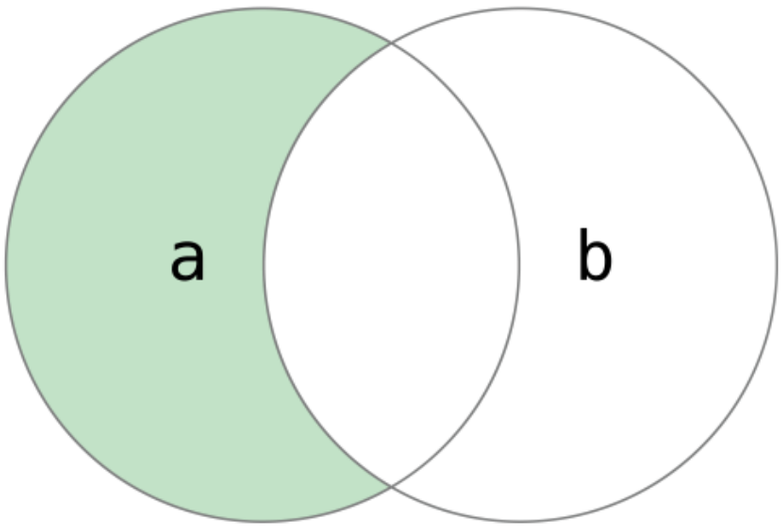
a.symmetricDifference(b)



a.union(b)



a.subtracting(b)



O QUE DEVO SABER NESTE PONTO

- Os tipos de dados básicos disponíveis em Swift
- Criar e manipular variáveis e constantes
- Tuplos
- Estruturas de control
- Loops
- Opcionais
- Collections

FUNÇÕES

- Bloco de código auto contido
  - Escopro proprio
  - Pode ser chamado varias vezes
- E possível recursividade

FUNÇÕES

- Cria funções
  - Sem retorno nem parâmetros

```
func nome() {
  // code goes here
}
```

- Sem retorno com parâmetros

```
func nome(parm:Type) {
  // code goes here
}

func nome(parm:Type, parm2:Type,) {
  // code goes here
}
```

FUNÇÕES

- Cria funções
  - Com retorno e com parâmetros

```
func nome(parm:Type) -> Type {
  // code goes here
}
```