FIAP

Arrays

NSArray, NSMutableArray, NSDictionary, NSMutableDictionary

OJB-C
Prof. Agesandro Scarpioni
agesandro@fiap.com.br

Arrays

FIAP

- Existem dois tipos de arrays imutável e mutável
  - Existe no Objective C um objeto NSArray capaz de receber uma coleção de objetos. Arrays imutáveis são tratados nessa classe.
  - Já os arrays mutáveis são tratados na classe NSMutableArray que é uma subclasse de NSArray.
  - No array da linguagem C, ele é incapaz de crescer ou encolher de tamanho, pois quando você cria um array ele tem um tamanho fixo e não se pode remover ou adicionar elementos sem recompilar o programa.

Arrays Imutáveis

FIAP

- Criando um array com NSArray e o método arrayWithObjects.

```
NSArray *array = [NSArray arrayWithObjects:@"Leão",@"Elefante",@"Girafa",@"Rinoceronte", nil];
```

  - O método arrayWithObjects espera um valor nil que indica o fim da lista, é errado chamar este método sem o nil ao final.
  - É possível converter um array em C utilizando o método arrayWithObjects.

```
NSString *cArray[] = {@"Pato",@"Marreco",@"Ganso",@"Cisne"};
NSArray *array2 = [NSArray arrayWithObjects:cArray count:3];
```

  - Também com NSArray podemos preencher um array com conteúdo da internet, desde que o arquivo seja um XML e esteja em conformidade com o esquema de uma Property List (plist).

```
NSArray *array3 = [NSArray arrayWithContentsOfURL:[NSURL URLWithString:@"http://www.teste.com.br/lista.plist"]];
```

Obs: Veja um exemplo de arquivo plist com um array no próximo slide, o arquivo plist nada mais é que um arquivo XML com algumas tags específicas. Sua utilidade é como a de um dicionário de dados (chave e valor), com a diferença que podemos especificar o tipo de dados.

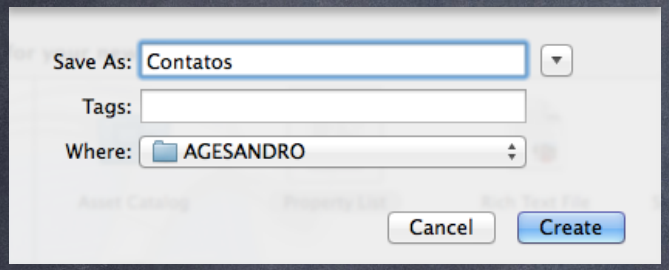
Arrays exemplo de plist xml

FIAP

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Contatos</key>
        <array>
                <dict>
                        <key>nome</key>
                        <string>Agesandro</string>
                        <key>telefone</key>
                        <string>7777-8899</string>
                </dict>
                <dict>
                        <key>nome</key>
                        <string>Maria</string>
                        <key>telefone</key>
                        <string>2222-4455</string>
                </dict>
                <dict>
                        <key>nome</key>
                        <string>Mariana</string>
                        <key>telefone</key>
                        <string>1111-2233</string>
                </dict>
                <dict>
                        <key>nome</key>
                        <string>Lorena</string>
                        <key>telefone</key>
                        <string>3333-4455</string>
                </dict>
                <dict>
                        <key>nome</key>
                        <string>Bianca</string>
                        <key>telefone</key>
                        <string>1234-5678</string>
                </dict>
        </array>
</dict>
</plist>
```

Criando um arquivo plist.

- Crie o arquivo contatos.plist em: File -> New -> File -> Resource -> Property List.



Contatos.plist

Contatos.plist

Contatos.plist 〉 No Selection

Key	Type	Value
▼ Root	Dictionary	(1 item)
▼ Contatos	Array	(5 items)
▼ Item 0	Dictionary	(2 items)
nome	String	Agesandro
telefone	String	7777-8899
▼ Item 1	Dictionary	(2 items)
nome	String	Maria
telefone	String	2222-4455
▼ Item 2	Dictionary	(2 items)
nome	String	Mariana
telefone	String	1111-2233
▼ Item 3	Dictionary	(2 items)
nome	String	Lorena
telefone	String	3333-4455
▼ Item 4	Dictionary	(2 items)
nome	String	Bianca
telefone	String	1234-5678

Arrays Mutáveis

FIAP

- Criando um array com NSMutableArray e o método arrayWithObjects.

```
NSMutableArray *array4 = [NSMutableArray arrayWithObjects:@"Onça",@"Leopardo",@"Tigre", nil];
```

Obs: Note que a única diferença de imutáveis para mutáveis é o uso da classe NSMutableArray

Arrays Acesso objectAtIndex

FIAP

- Acessando um elemento do array.

```
NSString *item = [array objectAtIndex:2];
NSLog(@"O ítem 2 do array é %@", item);
```

- Descobrindo quantos elementos possui um array.

```
NSUInteger qtd = [array count];
```

- Loop no array.

```
int x;
for (x=0; x < [ array count]; x++){
    NSString *animaisZoo = [array objectAtIndex:x];
    NSLog(@"Animal : %@", animaisZoo);
}
```

Arrays Acesso objectAtIndex

FIAP

- Acessar o último elemento forma 1.

```
NSString *item2 = [array objectAtIndex:[array count]-1];
NSLog(@"O último 2 do array é %@", item2);
```

- Acessar o último elemento por meio de um método.

```
item = [array lastObject];
NSLog(@"O último 2 do array é %@", item);
```

Arrays Busca
containsObject e indexOfObject

- Buscando um elemento específico em um array

```
BOOL achei = [array containsObject:@"Girafa"];
if (achei) {
    NSLog(@"Encontrei 'Girafa' no array");
} else {
    NSLog(@"Não encontrei 'Girafa' no array");
}
```

- Buscando uma posição específica em um array

```
int indice = [array indexOfObject:@"Girafa"];
if (indice != NSNotFound) {
    NSLog(@"Encontrei a 'Girafa' no índice %d",indice);
}
```

Obs: Lembre-se que se fosse em C teríamos que percorrer o array com estrutura de repetição, comparar cada ítem, sair do loop caso encontre e exibir o resultado.

Arrays Mutáveis

Inclusão

- Primeira etapa é criar um array que inicialmente pode ser até vazio, desde que seja do tipo NSMutableArray.

```
NSMutableArray *arrayZoo = [NSMutableArray  array];
```

- À medida que novos elementos são adicionados mais memória vai sendo alocada, porém, não é a forma mais eficiente de utilizarmos a memória, caso você já saiba pelo menos o valor mínimo de elementos você pode fazer uma pré alocação.

```
NSMutableArray *arrayZoo = [NSMutableArray arrayWithCapacity:50];
```

Obs: Não é porque aumentamos a capacidade para pelo menos 50 elementos, que não podemos adicionar mais itens, isso serve apenas para evitar a alocação excessiva, inclusive o número 50 não tem relação com o count, se o utilizarmos aparecerá 0 pois o array não possui elementos.

Arrays Mutáveis
Inclusão

FIAP

- A segunda etapa é utilizar o método addObject

```
NSMutableArray *arrayZoo = [NSMutableArray arrayWithCapacity:50];
[arrayZoo addObject:@"Urso"];
[arrayZoo addObject:@"Zebra"];
[arrayZoo addObject:@"Hipopótamo"];
```

- Também é possível adicionar um elemento no meio de um array, indicando o índice no qual o novo elemento deve ser inserido.

```
[arrayZoo insertObject:@"Camelo" atIndex:1];
```

```
2013-08-21 16:59:34.435 Testando_Array[696:303] Animal : Urso
2013-08-21 16:59:34.436 Testando_Array[696:303] Animal : Camelo
2013-08-21 16:59:34.436 Testando_Array[696:303] Animal : Zebra
2013-08-21 16:59:34.436 Testando_Array[696:303] Animal : Hipopótamo
```

Arrays Mutáveis
Alteração

- Para alterarmos o conteúdo de um array podemos utilizar o exemplo abaixo:

```
[arrayZoo replaceObjectAtIndex:3 withObject:@"Dromedário"];
```

```
NSMutableArray *arrayZoo = [NSMutableArray arrayWithCapacity:50];
[arrayZoo addObject:@"Urso"];
[arrayZoo addObject:@"Zebra"];
[arrayZoo addObject:@"Hipopótamo"];
[arrayZoo insertObject:@"Camelo" atIndex:1];
[arrayZoo replaceObjectAtIndex:3 withObject:@"Dromedário"];
```

Arrays Mutáveis

Exclusão

- Para alterarmos o conteúdo de um array podemos utilizar o removeObjectAtIndex informando o número do índice, como no exemplo abaixo:

```
[arrayZoo removeObjectAtIndex:1];
```

```
2013-08-21 18:03:25.755 Testando_Array[367:303] Animal : Urso
2013-08-21 18:03:25.756 Testando_Array[367:303] Animal : Zebra
2013-08-21 18:03:25.756 Testando_Array[367:303] Animal : Dromedário
```

FIAP

NSDictionary

- Muito semelhante a NSArray e NSMutableArray, porem, NSDictionary trabalha com o conceito de chave/valor. Segue abaixo um exemplo para criarmos um dicionário vazio, veja que como criamos com NSDictionary ficará para sempre vazio, o correto é criar com NSMutableDictionary.

```
NSDictionary *dicionario = [NSDictionary dictionary];
```

- No exemplo abaixo criamos um outro dicionário imutável com um valor e uma chave.

```
NSDictionary *dicio = [NSDictionary dictionaryWithObject:@"Mariana Medeiros" forKey:@"Nome"];
```

- Para exibirmos o conteúdo do NSDictionary, utilizamos o método objectForKey.

```
NSDictionary *dicio = [NSDictionary dictionaryWithObject:@"Mariana Medeiros" forKey:@"Nome"];
NSLog(@"O segundo dicionário possui o conteúdo %@", [dicio objectForKey:@"Nome"]);
```

Obs: Para contar quantos ítens temos em um Dictionary existe o método count como no Array.
Para URL no Dictionary temos o dictionaryWithContentsOfURL.

NSDictionary

- Uma outra forma de gerar o NSDictinary é utilizando um array com as chaves e outro array com os valores.

```
112
113     //Uma outra forma de montar um dicionário com dois arrays de chave e valor
114     NSArray *chave = [NSArray arrayWithObjects:@"Nome", @"Rua", @"Cidade", nil];
115     NSArray *valor = [NSArray arrayWithObjects:@"Maria", @"Rua da paz 9890", @"São Paulo", nil];
116     NSDictionary *dados=[NSDictionary dictionaryWithObjects:valor forKeys:chave];
117     NSLog(@"Este terceiro dicionário para o campo Nome possui o conteúdo %@", [dados objectForKey:@"Nome"]);
118     NSLog(@"Este terceiro dicionário para o campo Rua possui o conteúdo %@", [dados objectForKey:@"Rua"]);
119     NSLog(@"Este terceiro dicionário para o campo Cidade possui o conteúdo %@", [dados objectForKey:@"Cidade"]);
```

- É mais prático criar dicionários utilizando o método dictionaryWithObjectsAndKeys, sem a necessidade de criar arrays temporários.

```
NSDictionary *dados2 = [NSDictionary dictionaryWithObjectsAndKeys:@"Bianca",@"Nome",@"Av Paulista 15323",@"Rua",@"São Paulo",@"Cidade",nil];
NSLog(@"Este quarto dicionário para o campo Nome possui o conteúdo %@", [dados2 objectForKey:@"Nome"]);
NSLog(@"Este quarto dicionário para o campo Rua possui o conteúdo %@", [dados2 objectForKey:@"Rua"]);
NSLog(@"Este quarto dicionário para o campo Cidade possui o conteúdo %@", [dados2 objectForKey:@"Cidade"]);
```

FIAP

NSDictionary

- Outra forma de exibir o conteúdo de um dicionário:

```
NSArray *chave2 = [NSArray arrayWithObjects:@"Nome", @"RUA", @"Cidade", nil];
NSArray *valor2 = [dados objectsForKeys:chave2  notFoundMarker:@"Não Encontrado"];
NSLog(@"Os valores do dicionário são %@, %@, %@", [valor2 objectAtIndex:0], [valor2 objectAtIndex:1],[valor2 objectAtIndex:2]);
```

- A chave não encontrada tem como retorno a definição do notFoundMarker, que no nosso exemplo foi "Não Encontrado", pois escrevemos a chave Rua como RUA.

```
2013-08-26 00:04:23.590 Testando_Array[1444:303] Os valores do dicionário são Maria, Não Encontrado, São Paulo
```

FIAP

NSMutableDictionary

- Criando e inserindo uma chave e um valor utilizando a mensagem setObject: forKey:

```
131         NSMutableDictionary *dicionarioMutavel = [NSMutableDictionary dictionary];
132         [dicionarioMutavel setObject:@"João" forKey:@"Nome"];
133         [dicionarioMutavel setObject:@"Santos" forKey:@"Cidade"];
134         // se a chave já existir , o valor anterior será descartado e o novo valor assumirá seu lugar.
```

- Apagando uma chave/valor de um dicionário.

```
[dicionarioMutavel removeObjectForKey:@"Nome"];
NSLog(@"O dicionário mutável para o campo Nome possui o conteúdo %@", [dicionarioMutavel objectForKey:@"Nome"]);
```

- Também é possível criar um array de chaves e apagá-los.

```
NSArray *listaDeChaves = [NSArray arrayWithObjects:@"Nome", @"Cidade", "@Cep",nil];
[dicionarioMutavel removeObjectForKey:listaDeChaves];
```

- Para apagar todas as chaves e valores de um dicionário mutável usamos:

```
[dicionarioMutavel removeAllObjects];
```

NSMutableDictionary

- Alterando, ou seja, atualizando um valor de uma chave, podemos fazer de duas formas.

```
[dicionarioMutavel setObject:@"Maria" forKey:@"Nome"];
[dicionarioMutavel setValue:@"Florianópolis" forKey:@"Cidade"];
```

- Usar setObject:forKey: como o parâmetro nil fará com que uma excessão trave o aplicativo, pois nil não é um valor válido que podemos armazenar em um dicionário, para isso temos a classe NSNull semelhante a NSNumber que faz o boxing e o unboxing de nil, mais detalhes de boxing e unboxing no próximo slide.

```
170         //boxing com NSNull, nil (obj-c) Null(c)
171         NSNull *valorNulo = [NSNull null];
172         [dicionarioMutavel setValue:valorNulo forKey:@"Email"];
173          NSLog(@"O dicionário para o campo Email possui o conteúdo %@", [dicionarioMutavel objectForKey:@"Email"]);
174
175         //Unboxing
176          id meuValor = [dicionarioMutavel objectForKey:@"Email"];
177          NSLog(@"O valor armazenado é %@", meuValor);
178
```

Boxing

FIAP

- Classes como NSArray e NSDictionary esperam que suas chaves e seus valores sejam objetos do mesmo tipo. Um tipo boleano BOOL como YES ou NO, um float ou um int são primitivos e não são objetos, para isso temos que fazer o boxing, ou seja, um valor primitivo é encapsulado em um objeto contêiner (ou seja um box), e este objeto é enviado ao invés do primitivo, quando precisamos ler esse array ocorre o processo inverso, o unboxing, ou seja, o compilador detecta um valor encapsulado e retorna o conteúdo primitivo.

- O obj-C não permite o boxing e unboxing automático de primitivos como na linguagem Java 5 ou superior, VB ou C#, para isso, temos a classe NSNumber que realiza esta tarefa. No exemplo do próximo slide podemos converter qualquer tipo de primitivo para NSNumber e depois fazendo o unboxing devolve-lo. Assim conseguimos armazenar dados primitivos encapsulados em nosso array ou dicionário.

Classe NSNumber

```
159         //boxing
160         NSNumber *oNumero = [NSNumber numberWithInt:28];
161         [dicionarioMutavel setValue: oNumero forKey:@"Idade"];
162         NSLog(@"O dicionário para o campo Idade possui o conteúdo %@", [dicionarioMutavel objectForKey:@"Idade"]);
163
164        //unboxing para o primitivo
165         oNumero = [dicionarioMutavel objectForKey:@"Idade"];
166         int meuInteiro = [oNumero intValue];
167         NSLog(@"O primitivo que estava armazenado no array é %d", meuInteiro);
```

Obs: Da mesma forma que usamos para fazer o boxing o método numberWithInt, temos o numberWithFloat, numberWithChar, NumberWithBool. Para fazer o unboxing além de termos o intValue temos o floatValue, boolValue, etc.

Classe NSValue

- No exemplo anterior utilizamos o NSNumber que é uma subclasse de NSValue, da mesma forma que NSNumber é utilizado para boxing e unboxing de primitivos, o NSValue é utilizado para o boxing e unboxing de qualquer valor em C, claro que com uma programação mais complexa. Como não será nosso foco, fica a dica para quem necessitar pesquisar sobre NSValue para boxing de valores em C.

Prática

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

- Criar um array mutável vazio, inserir nesse array nomes de cursos da Fiap, fazer um loop para mostrar o conteúdo desse array.
- Incluir nesse array os nomes dos cursos da pós, alterar para um texto todo em maiúsculo apenas o nome do seu curso, refazer o loop e mostrar o novo conteúdo.
- Criar um dicionário mutável vazio, incluir dados para matrícula, nome, sobrenome, idade, altura, peso.
- Listar estes dados do dicionário.
- Apagar o dicionário com todas as suas chaves e seus respectivos valores.
- Listar novamente e comprovar que os dados foram apagados.