

Tabelas

- ❑ Uma tabela é uma entidade que contém uma lista ordenada de elementos
- ❑ Os elementos podem ser de qualquer tipo, mas têm que ser **todos do mesmo tipo**
- ❑ Os elementos são indexados por um índice que pode variar entre 0 e n-1, sendo n a dimensão da tabela

| | | | | | | | | | | | | | | | | | |
|-----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|
| 43 | 22 | 77 | 12 | 89 | 23 | 66 | 25 | 99 | 46 | 77 | 39 | 24 | 27 | 11 | 87 | ... | 26 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | ... | n-1 |
| length <input type="text"/> | | | | | | | | | | | | | | | | | |

Tabelas

- ❑ Para criar uma tabela:
 - Declarar a referência para a tabela
`float [] notas; // notas refere uma tabela de floats`
 - Criar a tabela
`notas = new float [50]; // tabela com espaço para 50 floats`
 - Inicializar e aceder aos elementos (usar `nome_da_tabela [i]` para aceder ao elemento índice i da tabela)
`notas [0] = 14.7;`
`notas [1] = 10.2;`
`float soma = notas [0] + notas [1];`
`System.out.println (notas [0]);`
 - O índice pode ser qualquer expressão inteira, ou seja uma expressão que tenha como resultado um inteiro

Tabelas

- Um elemento de uma tabela é um espaço onde se pode armazenar um valor do tipo declarado
 - `notas [0]` é um espaço onde pode ser armazenado um float
- Os elementos de uma tabela podem ser usados em qualquer ponto de um programa onde um elemento do mesmo tipo do dos seus elementos possa ser usado
 - Ex: `notas [1]` pode ser usado nas mesmas circunstâncias em que uma variável do tipo float pode ser usada (podemos atribuí-lhe um valor, imprimi-la, utilizá-la em expressões, ...)

Tabelas

- Quando se cria uma tabela com dimensão `n`, esse é o número máximo de elementos que ela pode armazenar
 - Uma vez criada não é possível alterar o seu tamanho
 - O campo `length` guarda o número máximo de elementos que a tabela pode armazenar. Este campo pode ser acedido (mas não modificado) pelo programa
`int numMax = notas.length;`

Tabelas

- Isto não implica que a tabela deva estar sempre cheia
 - Em cada momento poderá conter entre 0 e n elementos
 - Um cuidado importante quando se usam tabelas é respeitar os limites para os índices (0 a n-1)
 - A utilização de índices fora dos limites leva o intérprete de Java a gerar uma exceção (que deve ser tratada pelo programa como veremos mais tarde)

Tabelas

- É possível fazer atribuições entre tabelas

```
float [ ] notas1;  
notas1 = notas;
```

 - No entanto, dado que notas e notas1 são referências, o que se passa é que notas1 fica com o mesmo valor que notas, ou seja ambas referenciam a mesma tabela
 - Para criar uma cópia de uma tabela é necessário criar uma segunda tabela do mesmo tipo e depois fazer a atribuição explícita de todos os seus elementos

Tabelas

- ❑ Uma tabela tanto pode conter elementos de um tipo simples, como referências para outros objectos
 - Por exemplo `String [] frases = new String [25];` reserva espaço para 25 referências para objectos `String`
 - No entanto, esta instrução não cria as `Strings`, mas apenas as referências respectivas
 - As `Strings` têm que ser criadas explicitamente:
`frases [0] = new String ("Bom dia");` ou
`frases [1] = "Boa noite";`
 - Dado que um objecto pode conter tabelas nos seus campos, verifica-se que apenas com tabelas e objectos é possível criar estruturas de dados muito complexas

Tabelas

- ❑ Uma tabela pode ser passada como parâmetro a um método

```
float calcMedia (float [ ] tab, int num_als) {  
    ...  
}  
// chamada  
media = calcMedia (notas, conta);
```
- ❑ O que é realmente passado como parâmetro (por valor) é a referência da tabela
- ❑ Então o parâmetro formal e o real são duas referências para a mesma tabela

Tabelas

- ❑ Assim, alterar um elemento da tabela dentro do método, muda a tabela original
- ❑ Também é possível passar como parâmetro um elemento de uma tabela, desde que se sigam as regras referentes ao seu tipo
- ❑ Vamos ver alguns exemplos
- ❑ Consideremos as tabelas:

```
int[] lista = { 11, 22, 33, 44, 55};  
int[] lista2 = { 99, 99, 99, 99, 99};
```

Java - Tabelas

- ❑ E o método

```
public void passaElem (int num) {  
    num = 1234;  
}
```

 - Fazendo `passaElem (lista [1]);` como ficará a tabela?
 - Resposta: { 11, 22, 33, 44, 55};
- ❑ Agora o método

```
public void mudaElementos (int[] tab) {  
    tab [2] = 77;  
    tab [4] = 88;  
}
```

 - Fazendo `mudaElementos (lista);` como ficará a tabela?
 - Resposta: { 11, 22, 77, 44, 88};

Java - Tabelas

❑ Agora o método

```
public void mudaRef (int[] tab, int[] tab2) {  
    tab = tab2;  
}
```

■ Fazendo `mudaRef (lista, lista2)`; como ficará a tabela?

■ Resposta: { 11, 22, 77, 44, 88};

❑ Agora o método

```
public void copiaTab (int[] tab, int[] tab2) {  
    for (int index=0; index < tab2.length; index++)  
        tab[index] = tab2[index];  
}
```

■ Fazendo `copiaTab (lista, lista2)`; como ficará a tabela?

■ Resposta: { 99, 99, 99, 99, 99};

Java - Tabelas

❑ Agora o método

```
public int[] devolveRef (int[] tab2) {  
    tab2[1] = 9876;  
    return tab2;  
}
```

■ Fazendo `lista = devolveRef (lista2)`; como ficará a tabela?

■ Resposta: { 99, 9876, 99, 99, 99};

Tabelas

- ❑ Uma tabela uni-dimensional armazena uma lista de valores
- ❑ Uma tabela bi-dimensional representa uma matriz, com linhas e colunas
- ❑ Cada elemento de uma tabela bi-dimensional é referenciado usando dois índices (um para as linhas e outro para as colunas)
- ❑ Em Java uma tabela bi-dimensional é uma tabela de tabelas, ou seja é uma tabela uni-dimensional em que cada elemento é uma referência para um objecto tabela

Tabelas

- ❑ É possível criar e inicializar uma tabela bi-dimensional numa única instrução:

```
int [][] tabela = {{1,0,1}, {0,1,0}}
```
- Esta instrução cria uma tabela de inteiros com duas linhas e três colunas, inicializada com os valores dados

Tabelas

- ❑ Por exemplo, imaginemos que é preciso armazenar as classificações de cem alunos em quatro testes
- ❑ Poderíamos usar:
 - `float [][] notas = new float [100][4];`
 - Como seria de esperar, neste caso os índices podem variar entre 0 e 99 e entre 0 e 3, sendo errado tentar aceder a índices fora destas gamas
 - Para aceder a um dado elemento usam-se dois índices:
`notas [0][0] = 14.3;`
`notas [99][3] = 10.2;`
 - Pode saber-se a dimensão da tabela:
`int nLin = notas.length; int nCol = notas [0].length;`