

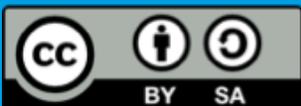


UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Centro Tecnológico
Departamento de Informática

Prof. Vítor E. Silva Souza
<http://www.inf.ufes.br/~vitorsouza>

[Desenvolvimento OO com Java] Um pouco de vetores



Esta obra está licenciada com uma licença Creative Commons Atribuição-
Compartilha Igual 4.0 Internacional: <http://creativecommons.org/licenses/by-sa/4.0/>.

Conteúdo do curso

- O que é Java;
- Variáveis primitivas e controle de fluxo;
- Orientação a objetos básica;
- ➔ Um pouco de vetores;
- Modificadores de acesso e atributos de classe;
- Herança, reescrita e polimorfismo;
- Classes abstratas e interfaces;
- Exceções e controle de erros;
- Organizando suas classes;
- Utilitários da API Java.

Estes slides foram baseados na [apostila do curso FJ-11: Java e Orientação a Objetos da Caelum](#) e na apostila Programação Orientada a Objetos em Java do [prof. Flávio Miguel Varejão](#).

Motivação

- Representar **diversas** variáveis associadas:

```
// Números para um bilhete da Mega Sena (R$ 3,50).
int numLoteria1;
int numLoteria2;
int numLoteria3;
int numLoteria4;
int numLoteria5;
int numLoteria6;

// Mas por R$ 17.517,50 eu posso jogar 15 números...
// E agora?
```

Vetores (arrays) em Java

- **Sintaxe** herdada de C:

```
int[] jogo1;    // Sintaxe preferida.
int jogo2[];   // Sintaxe C...
```

- Vetores são **objetos**, o que significa que no código acima temos apenas **referências**;
- Para **criar** objetos vetores se usa uma sintaxe **especial**:

```
jogo1 = new int[6];    // Jogo de R$ 3,50.
jogo2 = new int[15];   // Jogo de R$ 17.517,50 (!!)
```

Vetores (arrays) em Java

- Acessar o vetor também é igual a C:

```
jogo1[0] = 4; // Como em C, índices começam em 0
jogo1[1] = 8; // .
jogo1[2] = 15; // .
jogo1[3] = 16; // .
jogo1[4] = 23; // .
jogo1[5] = 42; // e vão até (tamanho - 1)
```

[] = operador de indexação

- Diferente de C, Java não permite acessos fora do vetor:

```
jogo1[6] = 43; // Só mais um número...
```

```
// Exception in thread "main"
```

```
// java.lang.ArrayIndexOutOfBoundsException: 6
```

Generalizando o tamanho do vetor

- Vetores são criados **dinamicamente**:

```
// Quantos você quer?
Scanner scanner = new Scanner(System.in);
int tamanho = scanner.nextInt();
int[] vetor = new int[tamanho];
```

- E possuem o **atributo** `length` pra facilitar seu uso:

```
// Preenchendo o vetor...
for (int i = 0; i < vetor.length; i++)
    vetor[i] = i;
```

Depois de criados, vetores não podem mudar de tamanho!

Vetores "de objetos"

- Vetores não estão limitados a tipos primitivos:

```
Conta[] minhasContas = new Conta[7];
Coordenadas[] pontos = new Coordenadas[5];
```

- Responda rápido: quantas contas e coordenadas foram criadas no código acima?

Zero!

```
// O código acima criou apenas referências!
// Então o que acontece se eu fizer isso aqui:
System.out.println(minhasContas[5].numero);
```

NullPointerException

Vetores são "zerados" na criação

- Mesma regra para atributos (0, false, null):

```
int[] jogo1 = new int[6]; // {0, 0, 0, 0, 0, 0}
Conta[] contas = new Conta[3]; // {null, null, null}
```

- Para usarmos objetos no vetor precisamos criá-los:

```
// Preenchendo o vetor... com objetos!
for (int i = 0; i < contas.length; i++)
    vetor[i] = new Conta();
```

```
// Pode usar atribuição também.
Conta conta = new Conta();
minhasContas[0] = conta;
```

Percorrendo um vetor

- Podemos usar um for e o atributo `length` como fizemos para preencher o vetor:

```
// Percorrendo o vetor...
for (int i = 0; i < vetor.length; i++)
    System.out.println(vetor[i]);
```

- A partir do Java 5, criou-se uma sintaxe mais interessante (*enhanced for* ou *for-each*):

```
// Idem acima...
for (int elem : vetor)
    System.out.println(elem);
```

```
// Agora com objetos...
for (Conta c : contas) System.out.println(c.saldo);
```

Atribuição composta

- Posso atribuir os valores do vetor em sua definição:

```
// Sintaxe completa:
int[] fibonacci6 = new int[] {1, 1, 2, 3, 5, 8};

// Sintaxe simplificada:
int[] copas = {1958, 1962, 1970, 1994, 2002};
```

Vetores multidimensionais

- Podemos criar vetores de múltiplas **dimensões**:

```
float[][] matriz = new float[5][6];
long[][] m2 = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

for (int i = 0; i < m2.length; i++) {
    for (int j = 0; j < m2[i].length; j++) {
        System.out.println(m2[i][j]);
    }
}
```

Matrizes são vetores de vetores, então é possível criar uma matriz cujas linhas tenham tamanhos diferentes! Consegue fazer?

Exercitar é fundamental

- Apostila FJ-11 da Caelum:
 - *Seção 5.5, página 62 (Empresa e Funcionario);*
 - *Seção 5.6, página 64 (Matrizes);*
 - *Seção 5.7, página 66 (Fibonacci);*
 - *Seção 5.8, página 66 (Casa e Porta).*