

Programação Orientada a Objetos e a Linguagem Java

Rodrigo da Cruz Fujioka
fujiokabr@gmail.com



Objetivos

www.rodriogofujioka.com

- Mais Sobre Arrays.
- Introdução a Coleções em Java - Collections.

Rodrigo Fujioka - Ling de Prog 2



Arrays

www.rodrihofujioka.com

- Em alguns programas sabemos a quantidade de objetos necessários (1 carro - 4 pneus)
- Em outros programas não sabemos quantos objetos serão necessários (1 conta – Várias transações)
 - Quantos objetos declarar então? Qual os nomes deles?
 - Uma solução é organizar objetos em estruturas que funcionem como “coleções”

Rodrigo Fujioka - Ling de Prog 2



Arrays

www.rodrigofujioka.com

- Array: arranjo de elementos
- Características
 - rápidos e eficientes
 - tamanho fixo
 - armazenam tipo específico (tipo base)
- Quando escolher arrays
 - problema é simples o suficiente
 - o número de elementos é fixo
 - performance é importante

Rodrigo Fujioka - Ling de Prog 2



Arrays

www.rodrigofujioka.com

- Array: arranjo de elementos
- Características
 - rápidos e eficientes
 - tamanho fixo
 - armazenam tipo específico (tipo base)
- Quando escolher arrays
 - problema é simples o suficiente
 - o número de elementos é fixo
 - performance é importante

Rodrigo Fujioka - Ling de Prog 2

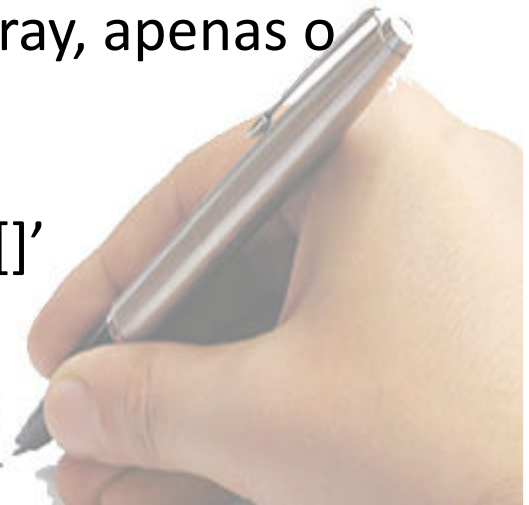


Arrays São Objetos

www.rodrigofujioka.com

- Um array é um objeto que armazena referências para outros objetos, podendo ser criados:
 - implicitamente: sintaxe de inicialização de arrays
 - `int[] tabela = {1,2,4,8,10};`
 - explicitamente: operador new
 - `int[] tabela = new int[10];`
- Único campo deste objetos: **length**
 - não sabemos quantos elementos estão no array, apenas o seu tamanho total
- Única forma de acesso ao conteúdo: operador '['

Rodrigo Fujioka - Ling de Prog 2

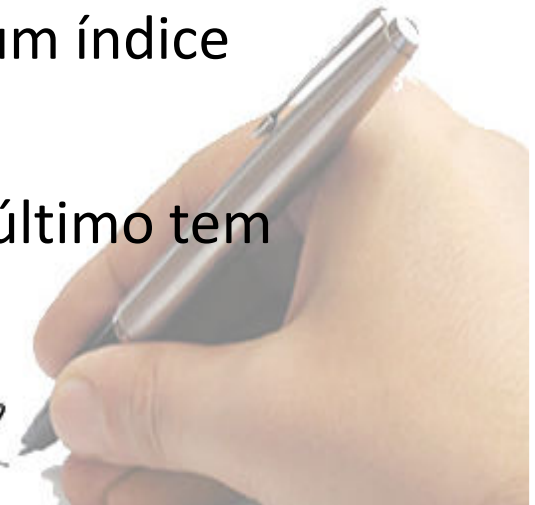


Arrays

www.rodrigofujioka.com

- Um array é declarado usando a sintaxe a seguir:
 - **Tipo[] nomeDoArray** ou
 - **Tipo nomeDoArray[]**
- Onde **nomeDoArray** armazena uma referência para um objeto array
- Cada elemento de um array é identificado por um índice
- O primeiro elemento do array tem índice 0 e o último tem índice tamanho – 1

Rodrigo Fujioka - Ling de Prog 2

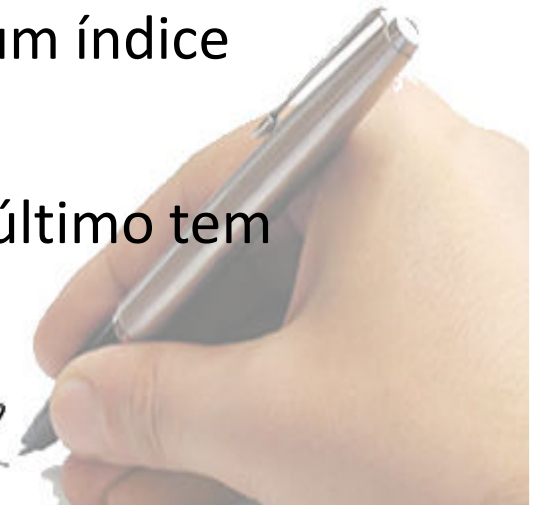


Declaração de Arrays

www.rodriofujioaka.com

- Um array é declarado usando a sintaxe a seguir:
 - **Tipo[] nomeDoArray** ou
 - **Tipo nomeDoArray[]**
- Onde **nomeDoArray** armazena uma referência para um objeto array
- Cada elemento de um array é identificado por um índice
- O primeiro elemento do array tem índice 0 e o último tem índice **tamanho – 1**

Rodrigo Fujioaka - Ling de Prog 2



Arrays

www.rodrihofujioka.com

```
// Initialization & re-assignment of arrays.
class Weeble {} // A small mythical creature
public class ArraySize {
    public static void main(String[] args) {
        // Arrays of objects:
        Weeble[] a; // Null reference
        Weeble[] b = new Weeble[5]; // Null references
        Weeble[] c = new Weeble[4];
        for(int i = 0; i < c.length; i++)
            c[i] = new Weeble();
        // Aggregate initialization:
        Weeble[] d = {new Weeble(), new Weeble(), new Weeble() };
        // Dynamic aggregate initialization:
        a = new Weeble[] { new Weeble(), new Weeble() };
        System.out.println("a.length=" + a.length);
        System.out.println("b.length = " + b.length);
        //References inside the array were initialized to null:
        for(int i = 0; i < b.length; i++)
            System.out.println("b[" + i + "]=" + b[i]);
    }
}
```

Rodrigo Fujioka - Ling de Prog 2



Arrays

www.rodrigofujioka.com

```
System.out.println("c.length = " + c.length);
System.out.println("d.length = " + d.length);
a = d;
System.out.println("a.length = " + a.length);
// Arrays of primitives:
int[] e; // Null reference
int[] f = new int[5];
int[] g = new int[4];
for(int i = 0; i < g.length; i++)
    g[i] = i*i;
int[] h = { 11, 47, 93 };
// Compile error: variable e not initialized:
//!System.out.println("e.length=" + e.length);
System.out.println("f.length = " + f.length);
//Primitives inside the array were initialized to zero:
for(int i = 0; i < f.length; i++)
    System.out.println("f[" + i + "]= " + f[i]);
```

Rodrigo Fujioka - Ling de Prog 2



Arrays

www.rodrigofujioka.com

```
System.out.println("g.length = " + g.length);  
System.out.println("h.length = " + h.length);  
e = h;  
System.out.println("e.length = " + e.length);  
e = new int[] { 1, 2 };  
System.out.println("e.length = " + e.length);  
}  
}
```

```
a.length=2  
b.length = 5  
b[0]=null  
b[1]=null  
b[2]=null  
b[3]=null  
b[4]=null  
c.length = 4  
d.length = 3  
a.length = 3
```

```
f.length = 5  
f[0]=0  
f[1]=0  
f[2]=0  
f[3]=0  
f[4]=0  
g.length = 4  
h.length = 3  
e.length = 3  
e.length = 2
```

Manipulando Arrays

www.rodrigofujioka.com

- Passando um array como parâmetro

```
Weeble[] d = {new Weeble(), new Weeble(), new Weeble() };  
ordene(d);
```

```
ordene(new Weeble[] {new Weeble(), new Weeble() } );
```

- Retornando um array

```
class Paises  
{  
    private static final String[] paises =  
        {"ALGERIA", "ANGOLA", "BENIN", "BOTSWANA"  
        /*...*/ }  
  
    public String[] retornaPaísesMundo()  
    { return paises; }  
}
```

Rodrigo Fujioka - Ling de Prog 2 4



Manipulando Arrays

www.rodrihofujioka.com

```
Conta [ ] contas = new Conta[10];  
...  
contas [0] = new Conta ("Conta Corrente BB");  
...  
if (contas[0].getSaldo() > 20.0)  
    // faz alguma coisa  
else  
    ...
```

Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodriofujioka.com

- A classe **java.util.Arrays**
 - conjunto de métodos (estáticos) utilitários
 - **equals()**: comparar igualdade de arrays
 - **fill()**: preencher um array com um valor
 - **sort()**: ordenar um array
 - **binarySearch()**: procurar elemento em array ordenado

Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodrigofujioka.com

Preenchendo: **Arrays.fill()**

- primitivo: copia mesmo valor em todas as posições
- referência: copia referência do objeto em todo array

Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodrigofujioka.com

```
// Using Arrays.fill()
import java.util.*;

public class FillingArrays {
    public static void main(String[] args) {
        int size = 6;
        // Or get the size from the command line:
        if(args.length != 0)
            size = Integer.parseInt(args[0]);
        boolean[] a1 = new boolean[size];
        int[] a2 = new int[size];
        String[] a3 = new String[size];
        Arrays.fill(a1, true);
        // a1 = {true, true, true, true, true, true}
        Arrays.fill(a2, 19);
        // a2 = {19, 19, 19, 19, 19, 19}
        Arrays.fill(a3, "Hello");
        // a3 = (Hello, Hello, Hello, Hello, Hello, Hello)
        Arrays.fill(a3, 3, 5, "World");
        // a3 = (Hello, Hello, Hello, World, World, Hello)
    }
}
```

Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodrihofujioka.com

- Comparando: **Arrays.equals()**
 - comparar igualdade de arrays
 - mesmo tamanho
 - cada elemento deve ser “**equals()**” ao seu correspondente


Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodrigofujioka.com

```
// Using Arrays.equals()  
import java.util.*;  
public class ComparingArrays {  
    public static void main(String[] args) {  
        int[] a1 = new int[10];  
        int[] a2 = new int[10];  
        Arrays.fill(a1, 47);  
        Arrays.fill(a2, 47);  
        System.out.println(Arrays.equals(a1, a2));  
        a2[3] = 11;  
        System.out.println(Arrays.equals(a1, a2));  
        String[] s1 = new String[5];  
        Arrays.fill(s1, "Hi");  
        String[] s2 = {"Hi", "Hi", "Hi", "Hi", "Hi"};  
        System.out.println(Arrays.equals(s1, s2));  
    }  
}
```



true
false
true

Rodrigo Fujioka - Ling de Prog 2

Classe Arrays

www.rodrihofujioka.com

- Realizando uma busca binária:

Arrays.binarySearch()

- antes da busca devemos ordenar o array
- caso contrário: resultado imprevisto
- produz um valor ≥ 0 se o item foi encontrado
- produz valor negativo caso contrário
- elementos duplicados
 - sem garantia de qual será retornado

Rodrigo Fujioka - Ling de Prog 2



Classe Arrays

www.rodrihofujioka.com

```
// Using Arrays.binarySearch()
import java.util.*;
public class ArraySearching {
    public static void main(String[] args) {
        int[] a = {3, 6, 8, 2, 1};
        Arrays.sort(a); // a = {1, 2, 3, 6, 8}
        int x = 2;
        int location = Arrays.binarySearch(a, x);
        if( location >= 0)
            System.out.println("Location of " + x + " is " +
                               location + ", a[" + location + "] = " + a[location]);
    }
}
```

Location of 2 is 1, a[1] = 2

Rodrigo Fujioka - Ling de Prog 2



Coleções(*Containers*)

www.rodrihofujioka.com

- Container
 - Objeto que agrupa vários outros objetos
- Biblioteca de classes containeres de Java
 - várias estruturas de dados
 - flexibilidade e reuso de implementação

Rodrigo Fujioka - Ling de Prog 2



Coleções(*Containers*)

www.rodrigofujioka.com

- Objetos manipulados por dois conceitos
 - **Collection: um agrupamento de objetos**
 - Set: armazena elementos únicos
 - List: armazena em uma seqüência particular
 - **Map: um agrupamento de pares chave-valor**

também chamadas de “arrays associativas”

Rodrigo Fujioka - Ling de Prog 2



Coleções(*Containers*)

www.rodrihofujioka.com

As coleções crescem de forma dinâmica não sendo necessário informar o tamanho das mesmas.

Rodrigo Fujioka - Ling de Prog 2



Coleções(*Containers*)

www.rodrigofujioka.com

```
import java.util.*;

public class PrintingContainers {
    static Collection fill(Collection c) {
        c.add("dog");
        c.add("dog");
        c.add("cat");
        return c;
    }
    static Map fill(Map m) {
        m.put("dog", "Bosco");
        m.put("dog", "Spot");
        m.put("cat", "Rags");
        return m;
    }
    public static void main(String[] args) {
        System.out.println(fill(new ArrayList()));
        System.out.println(fill(new HashSet()));
        System.out.println(fill(new HashMap()));
    }
}
```

```
C:\...>java PrintingContainers
[dog, dog, cat]
[cat, dog]
{cat=Rags, dog=Spot}
```

Rodrigo Fujioka - Ling de Prog 2

Coleções(*Containers*)

www.rodrigofujioka.com

- Comentários
- **ArrayList é um tipo de List, que é uma Collection**
 - uso do método **add()** para coleções
 - valores armazenados exatamente como fornecidos
 - nenhuma ordenação
- **HashSet é um tipo de Set, que é uma Collection**
 - um único valor “dog”
 - usa ordenação interna
 - nos preocupamos apenas com a pertinência de objetos
- **HashMap é um tipo de Map**
 - uso do método **put()**
 - um único valor chave
 - usa ordenação interna, não leva em conta a ordem de inclusão

Rodrigo Fujioka - Ling de Prog 2



Exercício

www.rodrigofujioka.com

- 1) Crie um método que receba 10 nomes em ordem aleatória e ao final informe a lista de nomes de forma ordenada.
- 2) Preencha um array de inteiro com 30 elementos com valor 8.

Nota: O programa deverá chamar uma outra classe que faça todos estes cálculos.

Rodrigo Fujioka - Ling de Prog 2



Dúvidas?

www.rodrihofujioka.com

? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ?

Rodrigo Fujioka - Ling de Prog 2



Bibliografia

www.rodrigofujioka.com

- **PEREIRA**, Frederico C. G.; Slides do Curso de Java. Disponível por WWW em <http://asterix.coinfo.cefetpb.edu.br/~fred>
- **ROCHA**, Helder da; Curso de Java. Disponível por WWW em <http://www.argonavis.com.br>.
- **The Java Tutorial**. Disponível por WWW em <http://java.sun.com/docs/books/tutorial/>
- **DEITEL**, Harvey M.; Paul.J. Java How to Program. 3rd. ed. - Prentice Hall.

