
Residência em Software
Residência em Tecnologia da Informação e Comunicação
Coleções – Parte 1
Professor:
Alvaro Degas Coelho
Java
INSTITUIÇÃO EXECUTORA
cepedi
UESC
COORDENADORA
MCTI FUTURO
Softex
APOIO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

Framework Collections

- Estruturas de dados de Coleções
  - Interfaces e Algoritmos
- Exemplos
  - Rol de Disciplinas, Lista de Clientes, etc.
- Uma Coleção
  - Olhar para a estrutura sem se preocupar com sua implementação
- Já fazemos isso com ArrayList – um tipo de coleção

Visão Geral

- Coleção
  - Estrutura d dados que armazena referência a outros objetos
- Framework Coleções
  - Provê Interfaces para realizar operações sobre vários tipos de coleção
- Sem preocupação com a implementação!

Visão Geral

- Colecções estão inclusas no package java.util
- São “irmãs” de package:
  - Collection
  - Set
  - List
  - Map
- Dá uma pesquisada nas demais!

Classe Array

- Métodos ESTÁTICOS para manipular Arrays
- Provides “high-level” methods
- Method binarySearch for searching sorted arrays
- Method equals for comparing arrays
- Method fill for placing values into arrays (overloaded to fill all or part of the array)
- Method sort for sorting arrays (overloaded to sort all or part of the array)
- Method arraycopy to copy portion of one array into another (java.lang.System)
- Methods overloaded to work with primitive-type arrays and object arrays

Exemplo

```
1   /// Exempo de Array
2   // Java arrays
3   import java.util.*;
4
5   public class UsingArrays {
6     private int intValues[] = { 1, 2, 3, 4, 5, 6 };
7     private double doubleValues[] = { 8.4, 9.3, 0.2, 7.9, 3.4 };
8     private int filledInt[], intValuesCopy[];
9
10    // initialize arrays
11    public UsingArrays()
12    {
13      filledInt = new int[ 10 ];
14      intValuesCopy = new int[ intValues.length ];
15
16      Arrays.fill( filledInt, 7 );   // fill with 7s
17
18      Arrays.sort( doubleValues );   // sort doubleValues ascending
19
20      // copy array intValues into array intValuesCopy
21      System.arraycopy( intValues, 0, intValuesCopy,
22        0, intValues.length );
23    }
24
```

Exemplo

```
1   // Exempo de Array
2   // Java arrays
3   import java.util.*;
4
5   public class UsingArrays {
6     private int intValues[] = { 1, 2, 3, 4, 5, 6 };
7     private double doubleValues[] = { 8.4, 9.3, 0.2, 7.9, 3.4 };
8     private int filledInt[], intValuesCopy[];
9
10    // initialize arrays
11    public UsingArrays()
12    {
13      filledInt = new int[ 10 ];
14      intValuesCopy = new int[ intValues.length ];
15
16      Arrays.fill( filledInt, 7 );   // fill with 7s
17
18      Arrays.sort( doubleValues );   // sort doubleValues ascending
19
20      // copy array intValues into array intValuesCopy
21      System.arraycopy( intValues, 0, intValuesCopy,
22        0, intValues.length );
23    }
24
```

Preeencher (fill) com um valor

Ordenar os dados do Array (ordem crescente)

Copiar dados de um Array para outro (Observe que é da classe System!!!)

Mais do exemplo

```
25    // Mostrar os valores de cada array
26    public void printArrays()
27    {
28      System.out.print( "doubleValues: " );
29
30      for ( int count = 0; count < doubleValues.length; count++ )
31        System.out.print( doubleValues[ count ] + " " );
32
33      System.out.print( "\nintValues: " );
34
35      for ( int count = 0; count < intValues.length; count++ )
36        System.out.print( intValues[ count ] + " " );
37
38      System.out.print( "\nfilledInt: " );
39
40      for ( int count = 0; count < filledInt.length; count++ )
41        System.out.print( filledInt[ count ] + " " );
42
43      System.out.print( "\nintValuesCopy: " );
44
45      for ( int count = 0; count < intValuesCopy.length; count++ )
46        System.out.print( intValuesCopy[ count ] + " " );
47
48      System.out.println();
49
50    } // fim printArrays
51
```

Mais do exemplo

```
52    //localizar um item
53    public int searchForInt( int value )
54    {
55      return Arrays.binarySearch( intValues, value );
56    }
57
58    // comparar se são iguais
59    public void printEquality()
60    {
61      boolean b = Arrays.equals( intValues, intValuesCopy );
62
63      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
64        " intValuesCopy" );
65
66      b = Arrays.equals( intValues, filledInt );
67
68      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
69        " filledInt" );
70    }
71
```

Mais do exemplo

```
52    //localizar um item
53    public int searchForInt( int value )
54    {
55      return Arrays.binarySearch( intValues, value );
56    }
57
58    // comparar se são iguais
59    public void printEquality()
60    {
61      boolean b = Arrays.equals( intValues, intValuesCopy );
62
63      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
64        " intValuesCopy" );
65
66      b = Arrays.equals( intValues, filledInt );
67
68      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
69        " filledInt" );
70    }
71
```

Método estático `binarySearch` da clsse `Arrays` pra fazer uma busca binária (precisa estar ordenado)

Método estático `equals` da classe `Arrays` para determinar se os valores de dois arrays tem os mesmos itens

Mais do exemplo

```
52    //localizar um item
53    public int searchForInt( int value )
54    {
55      return Arrays.binarySearch( intValues, value );
56    }
57
58    // comparar se são iguais
59    public void printEquality()
60    {
61      boolean b = Arrays.equals( intValues, intValuesCopy );
62
63      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
64        " intValuesCopy" );
65
66      b = Arrays.equals( intValues, filledInt );
67
68      System.out.println( "intValues " + ( b ? "==" : "!=" ) +
69        " filledInt" );
70    }
71
```

Conhecem esta operação aqui?
(pesquise!)

Mais do exemplo

```
public static void main( String args[] )
73     {
74        UsingArrays usingArrays = new UsingArrays();
75
76        usingArrays.printArrays();
77        usingArrays.printEquality();
78
79        int location = usingArrays.searchForInt( 5 );
80        System.out.println( ( location >= 0 ? "Found 5 at element " +
81           location : "5 not found" ) + " in intValues" );
82
83        location = usingArrays.searchForInt( 8763 );
84        System.out.println( ( location >= 0 ? "Found 8763 at element " +
85           location : "8763 not found" ) + " in intValues" );
86     }
87
88  } // end class UsingArrays
```

Residência em Software

Arrays e Listas

- Listas são estruturas ordenadas com ligações (ponteiros) entre cada elemento
- Arrays são estruturas ordenadas com alocação em posições adjacentes de memória
- Listas são dinâmicas
- Arrays são estáticos
- Arrays e Listas são (meio) comutáveis

Arrays e Listas

- Ver um Array como um List: duas maneiras
  - `private static final String suits[] = { "Hearts", "Diamonds", "Clubs", "Spades" };`
  - `List list = new ArrayList( Arrays.asList( suits ) );`
- list é independente do array suits: mudar um não muda o outro!
  - Os objetos foram copiados!
- Outra opção
  - `List list = Arrays.asList( suits );`
- list é uma estrutura em que cada elemento do array aparece ligado ao próximo
  - Mudanças em objetos de um são refletidas no outro!

Arrays e Listas

```
public static void main( String args[] )
73     {
74        UsingArrays usingArrays = new UsingArrays();
75
76        usingArrays.printArrays();
77        usingArrays.printEquality();
78
79        int location = usingArrays.searchForInt( 5 );
80        System.out.println( ( location >= 0 ? "Found 5 at element " +
81           location : "5 not found" ) + " in intValues" );
82
83        location = usingArrays.searchForInt( 8763 );
84        System.out.println( ( location >= 0 ? "Found 8763 at element " +
85           location : "8763 not found" ) + " in intValues" );
86     }
87
88  } // UsingArrays
```

Arrays e Listas

```
1   // Exemplo
2   // metodo asList.
3   import java.util.*;
4
5   public class UsingAsList {
6     private static final String values[] = { "red", "white", "blue" };
7     private List list;
8
9     // inicializa List e muda o valor da posição 1
10    public UsingAsList()
11    {
12      list = Arrays.asList( values );   // gera a lista
13      list.set( 1, "green" );           // muda o valor
14    }
15
16    // mostrando a lista e o array
17    public void printElements()
18    {
19      System.out.print( "List elements : " );
20
21      for ( int count = 0; count < list.size(); count++ )
22        System.out.print( list.get( count ) + " " );
23
24      System.out.print( "\nArray elements: " );
25
26      for ( int count = 0; count < values.length; count++ )
27        System.out.print( values[ count ] + " " );
28
29      System.out.println();
30    }
31
32    public static void main( String args[] )
33    {
34      new UsingAsList().printElements();
35    }
36
37  } // UsingAsList
```

Exercício

- Suponha um método que preenche um array de nomes de pessoas.
  - Escreva um método que verifica se um nome específico está presente no array.
  - Elabore uma versão deste método usando o array e outra convertenddo o array para uma lista
  - Escreva um método que MODIFICA um nome específico no array, se ele existir (gere uma exception caso contrário)
  - Elabore uma versão deste método usando o array e outra convertenddo o array para uma lista