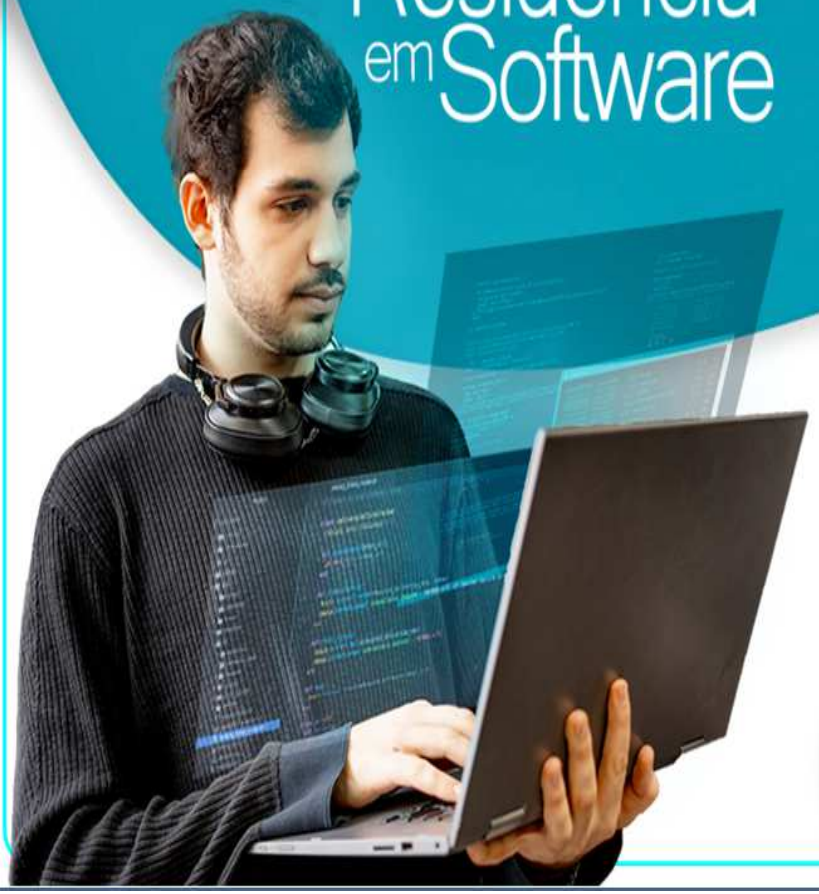
# Residência em Tecnologia da Informação e Comunicação

## Coleções – 2ª parte

Residência em Software

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

# Collections: Interface e Classe

- Há uma Interface Collections
- Determina padrões de métodos para
  - Adicionar, apagar, comparar…
- Duas outras interfaces: `Set` e `List` estendem a interface `Collection`
- Provê o uso de um `Iterator`
  - `Similar ao Iterator em C++`

# Collections: Interface e Classe

- Há uma Classe Collections
- Dispõe de métodos estáticos para manipular coleções em geral
  - Há coleções de múltiplos tipos de objetos
- As coleções podem ser manipuladas polimorficamente
  - É possível aplicar os métodos para coleções de quaisquer objetos

# List

- A Nossa primeira Collection
- Listas (Lists)
  - Coleção Ordenada que permite elementos duplicados
  - Implementada pela interface List (que disponibiliza vários métodos)
- Tipos de objetos que implementam List
  - ArrayList (resizable array) – Já é nosso conhecido!
  - LinkedList
  - Vector
- Iterator (para percorrer)
  - Método listIterator: um iterator bidirecional
  - Parâmetro listIterator: indica onde a iteração inicia

# Exemplo

```
1   // CollectionTest.java
2   // Collection interface.
3   import java.awt.Color;
4   import java.util.*;
5
6   public class CollectionTest {
7     private static final String colors[] = { "red", "white", "blue" };
8
9     // Cria um ArrayList, manipula objetos nele
10    public CollectionTest()
11    {
12      List list = new ArrayList();
13
14      // adiciona
15      list.add( Color.MAGENTA );    // add a color object
16
17      for ( int count = 0; count < colors.length; count++ )
18        list.add( colors[ count ] );
19
20      list.add( Color.CYAN );       // add a color object
21
22      // Mostra
23      System.out.println( "\nArrayList: " );
24
25      for ( int count = 0; count < list.size(); count++ )
26        System.out.print( list.get( count ) + " " );
27
```

# Exemplo

```
1   // CollectionTest.java
2   // Collection interface.
3   import java.awt.Color;
4   import java.util.*;
5
6   public class CollectionTest {
7     private static final String colors[] = { "red", "white", "blue" };
8
9     // Cria um ArrayList, manipula objetos nele
10    public CollectionTest()
11    {
12      List list = new ArrayList();
13
14      // adiciona
15      list.add( Color.MAGENTA );    // add a color object
16
17      for ( int count = 0; count < colors.length; count++ )
18        list.add( colors[ count ] );
19
20      list.add( Color.CYAN );       // add a color object
21
22      // Mostra
23      System.out.println( "\nArrayList: " );
24
25      for ( int count = 0; count < list.size(); count++ )
26        System.out.print( list.get( count ) + " " );
27
```

Add: adiciona elementos na lista

Get: retorna um elemento (numa posição) da lista

# Exemplo

```
28      // remove Strings
29      removeStrings( list );
30
31      // lista elementos
32      System.out.println( "\n\nArrayList after calling removeStrings: " );
33
34      for ( int count = 0; count < list.size(); count++ )
35        System.out.print( list.get( count ) + " " );
36
37    } // CollectionTest
38
39    // método que remove Strings
40    private void removeStrings( Collection collection )
41    {
42      Iterator iterator = collection.iterator(); // cria iterator
43
44      // percorre
45      while ( iterator.hasNext() )
46
47        if ( iterator.next() instanceof String )
48          iterator.remove();  // remove objectos String
49    }
50
```

# Exemplo

```
28    // remove Strings
29    removeStrings( list );
30
31    // lista elementos
32    System.out.println( "\n\nArrayList after calling removeStrings: " );
33
34    for ( int count = 0; count < list.size(); count++ )
35       System.out.print( list.get( count ) + " " );
36
37 } // CollectionTest
38
39 // método que remove Strings
40 private void removeStrings( Collection collection )
41 {
42    Iterator iterator = collection.iterator(); // cria
43
44    // percorre
45    while ( iterator.hasNext() )
46
47       if ( iterator.next() instanceof String )
48          iterator.remove();  // remove objectos String
49 }
50
```

Este método remove objetos do tipo String

Cria o inerator

Método hasNext verifica se ainda há elementos

Next avança o iterator

Método remove exclui um objeto da lista


# Final do exemplo (main)

```
51    public static void main( String args[] )
52    {
53       new CollectionTest();
54    }
55
56 } CollectionTest
```

# LinkedList

```
1   // ListTest.java
2   // LinkLists.
3   import java.util.*;
4
5   public class ListTest {
6     private static final String colors[] = { "black", "yellow",
7       "green", "blue", "violet", "silver" };
8     private static final String colors2[] = { "gold", "white",
9       "brown", "blue", "gray", "silver" };
10
11    // Manipular objetos LinkedList
12    public ListTest()
13    {
14      List link = new LinkedList();
15      List link2 = new LinkedList();
16
17      // adicionar
18      for ( int count = 0; count < colors.length; count++ ) {
19        link.add( colors[ count ] );
20        link2.add( colors2[ count ] );
21      }
22
23      link.addAll( link2 );         // concatenar listas
24      link2 = null;               // reiniciar a lista (vazia)
25
```

# LinkedList

```
1  // ListTest.java
2  // LinkLists.
3  import java.util.*;
4
5  public class ListTest {
6    private static final String colors[] = { "black", "yellow",
7      "green", "blue", "violet", "silver" };
8    private static final String colors2[] = { "gold", "white",
9      "brown", "blue", "gray", "silver" };
10
11   // Manipular objetos LinkedList
12   public ListTest()
13   {
14     List link = new LinkedList();
15     List link2 = new LinkedList();
16
17     // adicionar
18     for ( int count = 0; count < colors.length; count++ ) {
19       link.add( colors[ count ] );
20       link2.add( colors2[ count ] );
21     }
22
23     link.addAll( link2 );        // concatenar listas
24     link2 = null;                // reiniciar a lista (vazia)
25
```

Cria dois objetos LinkedList

Método `addAll` acrescentar os elementos de `link2` em `link`

Zera (null) `link2` (vai para o Garbage Collection)

# LinkedList

```
26        printList( link );
27
28        uppercaseStrings( link );
29
30        printList( link );
31
32        System.out.print( "\nDeleting elements 4 to 6..." );
33        removeItems( link, 4, 7 );
34
35        printList( link );
36
37        printReversedList( link );
38
39     } // end constructor ListTest
40
41     // output List contents
42     public void printList( List list )
43     {
44        System.out.println( "\nlist: " );
45
46        for ( int count = 0; count < list.size(); count++ )
47           System.out.print( list.get( count ) + " " );
48
49        System.out.println();
50     }
```

# LinkedList

```
26        printList( link );
27
28        uppercaseStrings( link );
29
30        printList( link );
31
32        System.out.print( "\nDeleting elements 4 to 6..." );
33        removeItems( link, 4, 7 );
34
35        printList( link );
36
37        printReversedList( link );
38
39     } // end constructor ListTest
40
41     // output List contents
42     public void printList( List list )
43     {
44        System.out.println( "\nlist: " );
45
46        for ( int count = 0; count < list.size(); count++ )
47           System.out.print( list.get( count ) + " " );
48
49        System.out.println();
50     }
```

Método printList para listar os valores (observe o for)

# LinkedList

```
51
52    // locate String objects and convert to uppercase
53    private void uppercaseStrings( List list )
54    {
55       ListIterator iterator = list.listIterator();
56
57       while ( iterator.hasNext() ) {
58          Object object = iterator.next();  // get item
59
60          if ( object instanceof String )   // check for String
61             iterator.set( ( ( String ) object ).toUpperCase() );
62       }
63    }
64
65    // obtain sublist and use clear method to delete sublist items
66    private void removeItems( List list, int start, int end )
67    {
68       list.subList( start, end ).clear();  // remove items
69    }
70
71    // print reversed list
72    private void printReversedList( List list )
73    {
74       ListIterator iterator = list.listIterator( list.size() );
75
```

# LinkedList

```
51
52  // locate String objects and convert to uppe
53  private void uppercaseStrings( List list )
54  {
55    ListIterator iterator = list.listIterator();
56
57    while ( iterator.hasNext() ) {
58      Object object = iterator.next();  // get item
59
60      if ( object instanceof String )  // check for String
61        iterator.set( ( ( String ) object ).toUpperCase() );
62    }
63  }
64
65  // obtain sublist and use clear method t
66  private void removeItems( List list, int start, int end )
67  {
68    list.subList( start, end ).clear();  // remove items
69  }
70
71  // print reversed list
72  private void printReversedList( List list )
73  {
74    ListIterator iterator = list.listIterator( list.size() );
75
```

O Iterator para percorrer. Se for String, tornar as letrasmaiúsculas

O método sublist seleciona intervalos da lista para serem removidos com o método clear

# LinkedList

```
76        System.out.println( "\nReversed List:" );
77
78        // iterator em ordem reversa
79        while( iterator.hasPrevious() )
80           System.out.print( iterator.previous() + " " );
81     }
82
83     public static void main( String args[] )
84     {
85        new ListTest();
86     }
87
88  } // ListTest
```

# LinkedList

```
76        System.out.println( "\nReversed List:" );
77
78        // iterator em ordem reversa
79        while( iterator.hasPrevious() )
80           System.out.print( iterator.previous() + " " );
81     }
82
83     public static void main( String args[] )
84     {
85        new ListTest();
86     }
87
88  } // ListTest
```

Método hasPrevious indica se ainda há elementos anteriores

O método previous retorna o itertor para a posição anterior

# ArrayLists e Vectors

- Estes já são nossos conhecidos
  - Pesquise!
- Exercícios
  - Crie um método para Inserir elementos em uma LinkedList e, em seguida, exibir os elementos na ordem inversa.
    - Use iterator
  - Crie um método para adicionar números inteiros a um ArrayList, calcular a soma e exibir a média.
  - Crie métodos para adicionar e remover elementos de um Vector de strings.