Programação Orientada a Objetos

Unidade 3 – Classes de Tipos Genéricos e ArrayList

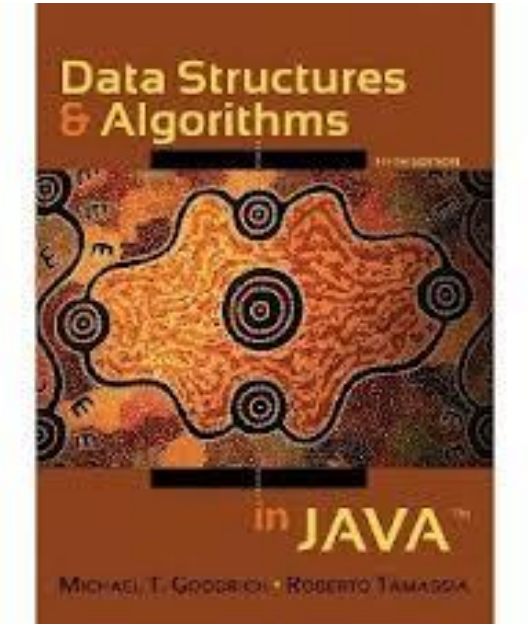
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

Bibliografia

- Data Structures and Algorithms in Java – Fourth Edition – Roberto Tamassia – Michael T. Goodrich – John Wiley & Sons, Inc
- Core Java Fundamentals – Horstmann / Cornell – PTR- Volumes 1 - 7th Edition
- Beginning Java 2 – Ivor Horton – 2011 – 7 Edittion - WROX
- Head First Java, 2nd Edition by Kathy Sierra and Bert Bates
- Estrutura de Dados e Algoritmos – Bruno R. Preiss, Editora Campus, 2001
- Estrutura de Dados e Algoritmos em Java – Robert Lafore, Editora Ciência Moderna, 2005
- Algoritmos e Estrutura de Dados – Niklaus Wirth – Editora Prentice Hall do Brasil, 1989

Leitura Recomendada para a Unidade 3

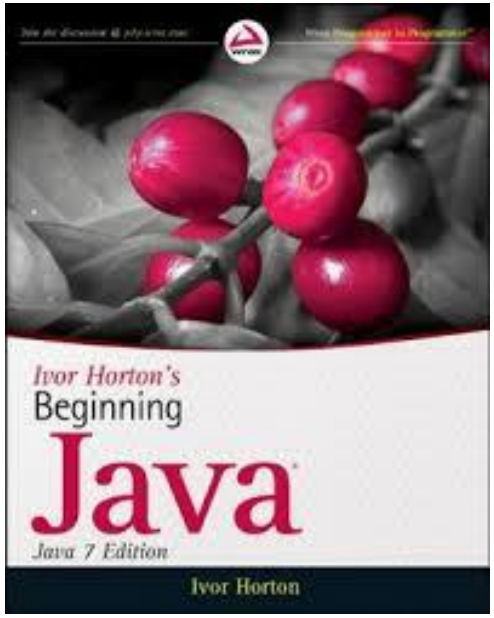
- Data Structures and Algorithms in Java (*), Tamassia, Seção 2.5.2 e Seção 6.1



(*) Em português, Estrutura de Dados e Algoritmos em Java

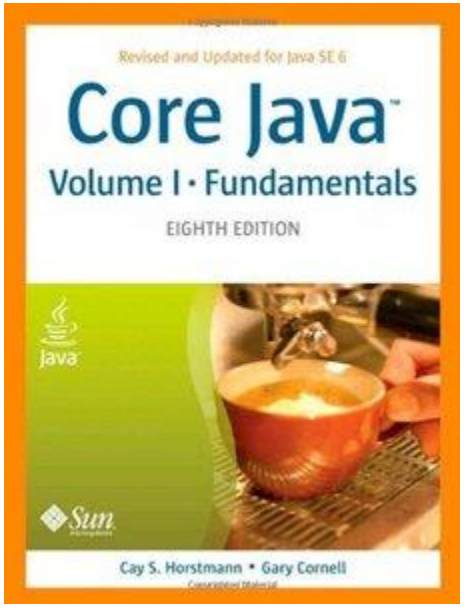
Leitura Recomendada para a Unidade 3

- Ivor Horton's Beginning Java – 7h Edition – WROX - 2011

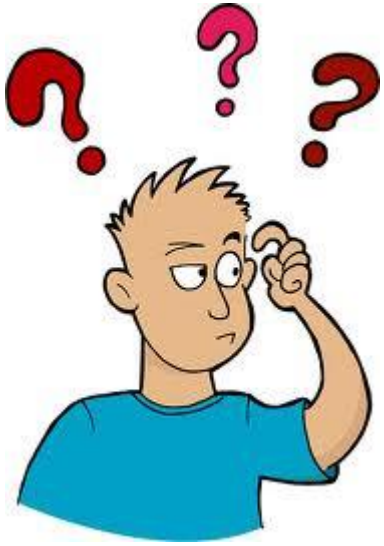


Leitura Recomendada para a Unidade 3

- Core Java – Volume 1 – Fundamentals – Eighth Edition – Cay. Horstmann – Gary Cornell



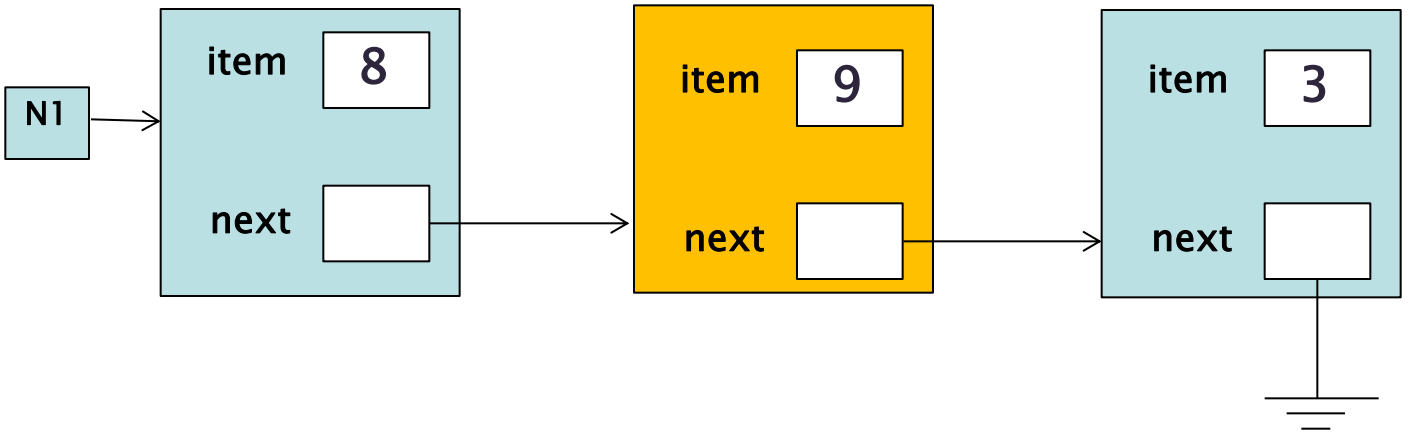
O que são tipos genéricos ?



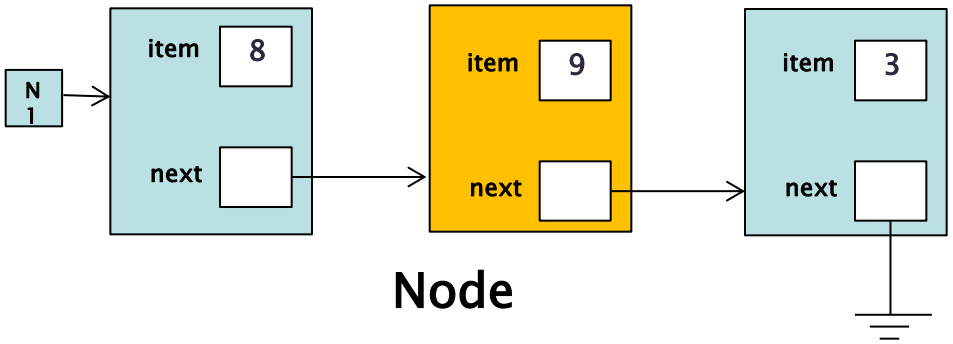
Tipo Genérico

- Também chamado de **tipo parametrizado**, é uma definição de classe que tem um ou mais tipos de parâmetros.
- Por exemplo, considere uma lista ligada de inteiros com um conjunto de operações definidas.
- Poderíamos necessitar da mesma lista ligada para implementar Strings, e assim por diante.

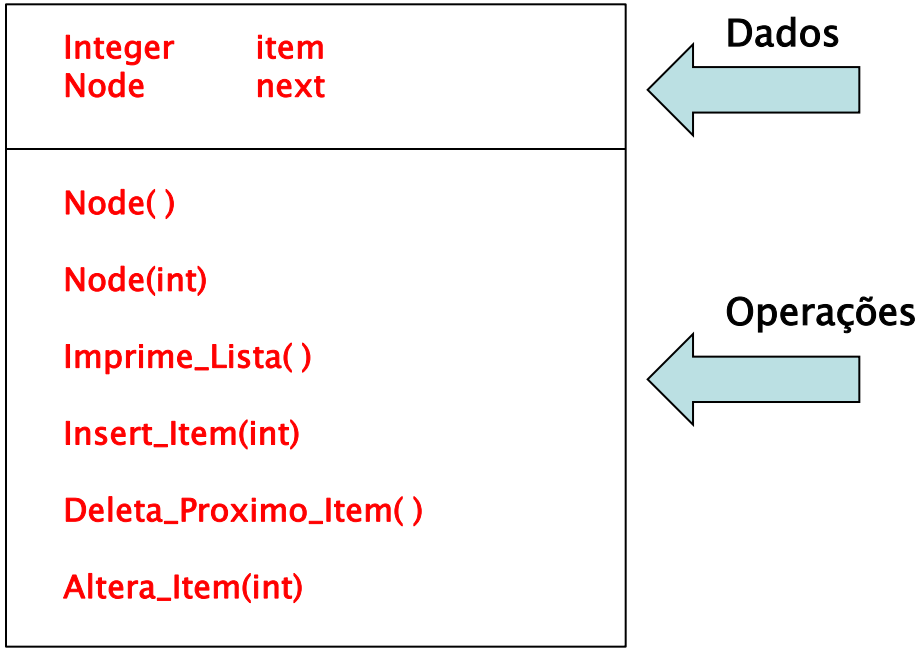
Lista ligada de Inteiros



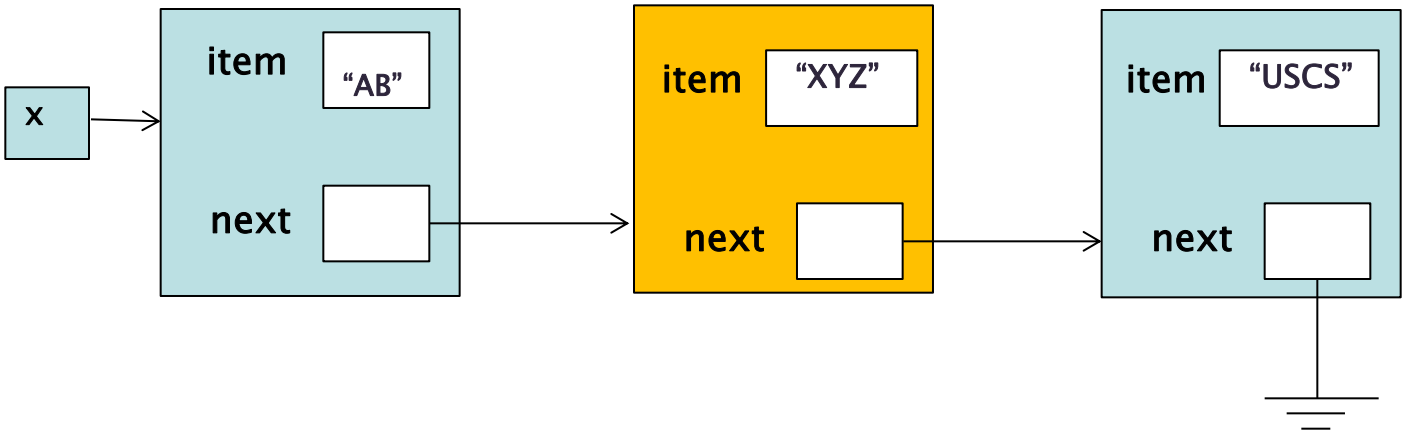
Lista ligada de Inteiros



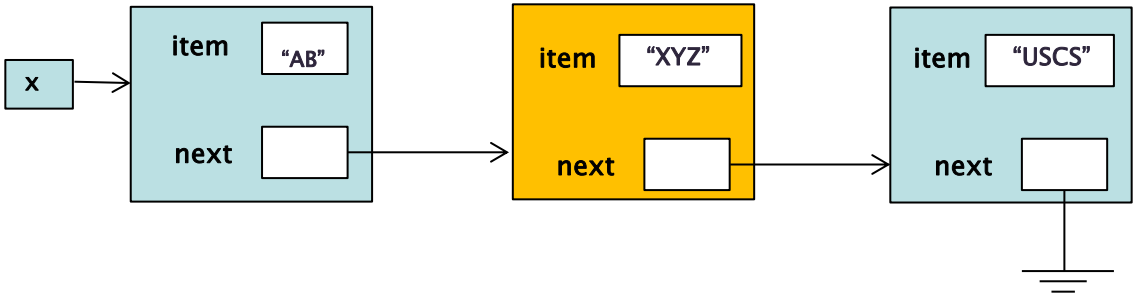
TAD



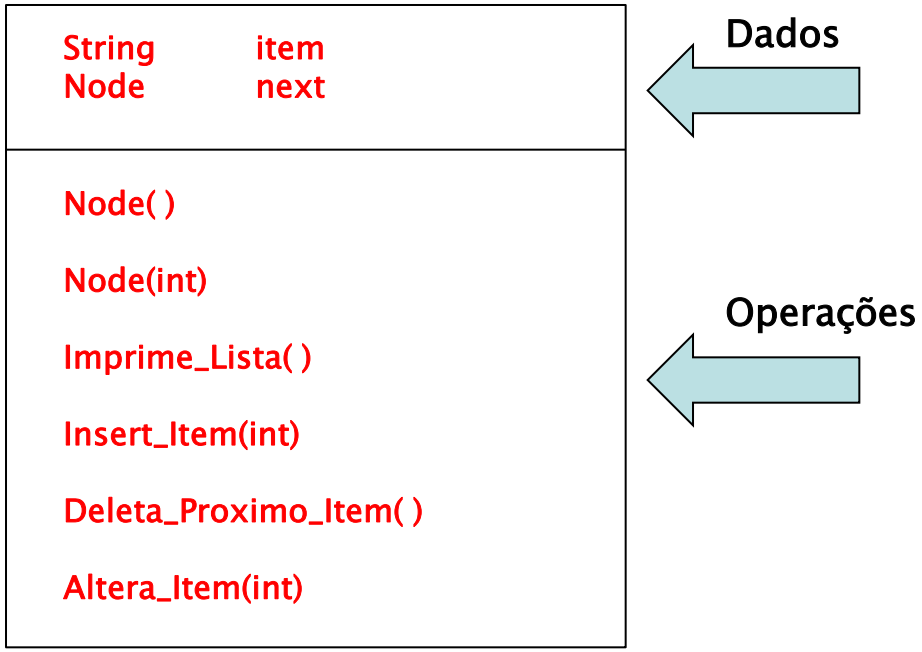
Lista ligada de Strings



Lista ligada de Strings



TAD



Coleção de classes

- A lista ligada é um exemplo de classe que pode ser definida como uma **classe de tipo genérico**.
- Uma classe de tipo genérico define coleção de classes (**collection**).
- Uma lista ligada com tipos genéricos apresenta a vantagem de ser aplicável à qualquer tipo de dados que quizermos implementar a lista.
- São semelhantes aos templates em C++ (**STL** = Standard Template Library).

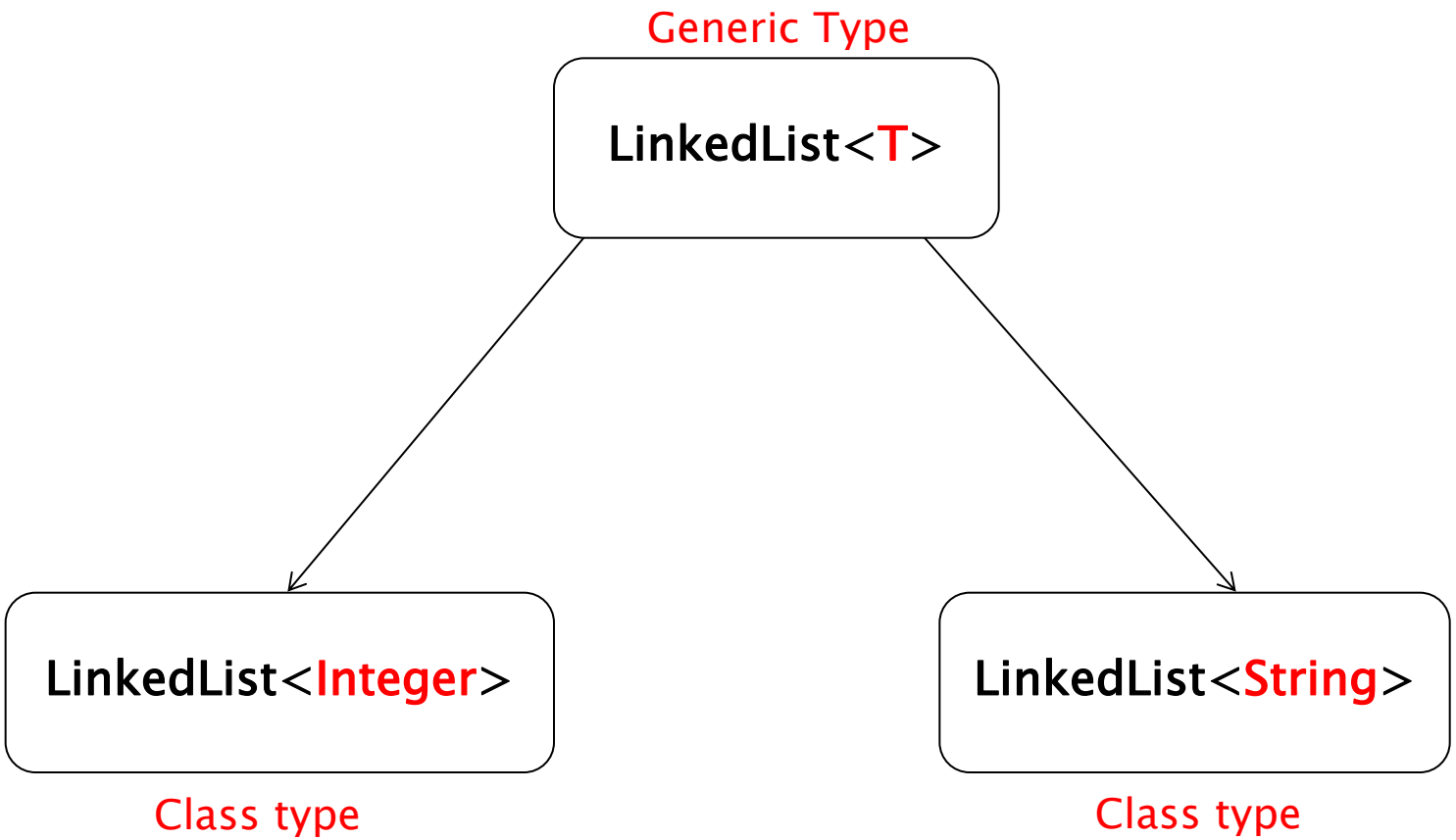
Como definir uma classe de tipo genérico ?

Classe Genérica – Definição

```
public    class    UserClass<T>  {

          //definicao de tipo genérico
}
```

- **T** entre **< >** é chamado tipo genérico (type parameter).
- Para se criar uma classe a partir da classe genérica, devemos simplesmente fornecer um apropriado argumento para o parâmetro entre **<** e **>**.
- Por exemplo: LinkedList<int> ou LinkedList<String> .

Classes Genéricas



Classes Genéricas – Observações

- O argumento para o tipo genérico deve ser uma **classe** ou **interface**.
- Ou seja, não é permitido o uso de tipos primitivos, por exemplo, int ou double. Deve-se usar wrapper classes, tais como Integer, Double, etc.
- Ao se criar um tipo particular a partir de um genérico, o argumento é substituído em toda a ocorrência de **T** na especificação genérica de tipo.

Implementação de Lista de Integers

Lista ligada de Integers

```
package uscs;

public class Node_Int {

        Integer item;
        Node_Int next;

public Node_Int(Integer item) {

        this.item = item;
        this.next = null;
}


public Node_Int() {

        this.item = 0;
        this.next = null;
}
```

Função para inserir Integers

```
public void insert_Item(Integer item) {

        Node_Int no_trab = new
        Node_Int(item);

        Node_Int p = this;

        while (p.next != null)

                p = p.next;

                p.next = no_trab;

}
```

Função para imprimir Lista

```
public void Imprime_Lista() {
        Node_Int no_trab = this;
        System.out.print("Lista: ");
        while (no_trab != null ) {
                System.out.print("   " + no_trab.item);
                no_trab = no_trab.next ;
        }
        System.out.println("");
 }
}
```

Classe para execução

```
package uscs;

public class Teste_Node_Int {

        public static void main(String[] args) {

                Node_Int x = new Node_Int(7);

                x.insert_Item(5);
                x.insert_Item(2);
                x.insert_Item(9);

                x.Imprime_Lista();

        }
}
```

Lista: 7 5 2 9

Implementação de Lista de Strings

Lista ligada de Strings

```
package uscs;

public class Node_String {

        String item;
        Node_String next;

public Node_String(String item) {

        this.item = item;
        this.next = null;
}


public Node_String() {

        this.item = "";
        this.next = null;
}
```

Função para inserir Strings

```
public void insert_Item(String item) {
        Node_String no_trab = new Node_String(item);
        Node_String  p = this;
        while (p.next != null)
                p = p.next;
                p.next = no_trab;
}
```

Função para imprimir Lista

```
public void Imprime_Lista() {
        Node_String  no_trab = this;
        System.out.print("Lista: ");
        while (no_trab != null ) {
                System.out.print("   " + no_trab.item);
                no_trab = no_trab.next ;
        }
        System.out.println("");
   }
}
```

Classe para execução

```
package uscs;

public class Teste_Node_String {

        public static void main(String[] args) {

                Node_String  x = new Node_String("Sao Paulo");

                x.insert_Item("Corinthians");
                x.insert_Item("Santos");
                x.insert_Item("Palmeiras");

                x.Imprime_Lista();

        }
}
```

Lista: Sao Paulo Corinthians Santos Palmeiras

Implementação de classe Genérica

Lista ligada Genérica

```
package uscs;

public class Node<T>     {

        T item;
        Node<T> next;

        public Node(T item) {

                this.item = item;
                this.next = null;
        }

        public Node() {

                this.item = null;
                this.next = null;
        }
```

Função Genérica de Inserção

```
public void insert_Item(T item) {
        Node<T> no_trab = new Node<T>(item);
        Node<T> p = this;
        while (p.next != null)
                p = p.next;
        p.next = no_trab;
}
```

Função Genérica de Impressão

```
  public void Imprime_Lista() {
        Node<T> no_trab = this;
        System.out.print("Lista: ");
        while (no_trab != null ) {
                System.out.print("   " + no_trab.item);
                no_trab = no_trab.next ;
        }
        System.out.println("");
   }


}
```

Classe para execução

```
package uscs;

public class Teste_Node {

   public static void main(String[] args ){

        Node<Integer> x = new Node<>(10);

        x.insert_Item(20);
        x.insert_Item(30);

        x.Imprime_Lista();

        Node<String> y = new Node<>("Corinthians");
        y.insert_Item("Sao Paulo");
        y.insert_Item("Santos");

        y.Imprime_Lista();

    }
}
```

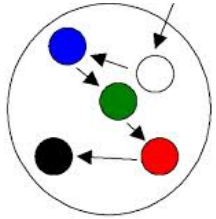
```
Lista:    10   20   30
Lista:    Corinthians   Sao Paulo   Santos
```

API Collections Framework

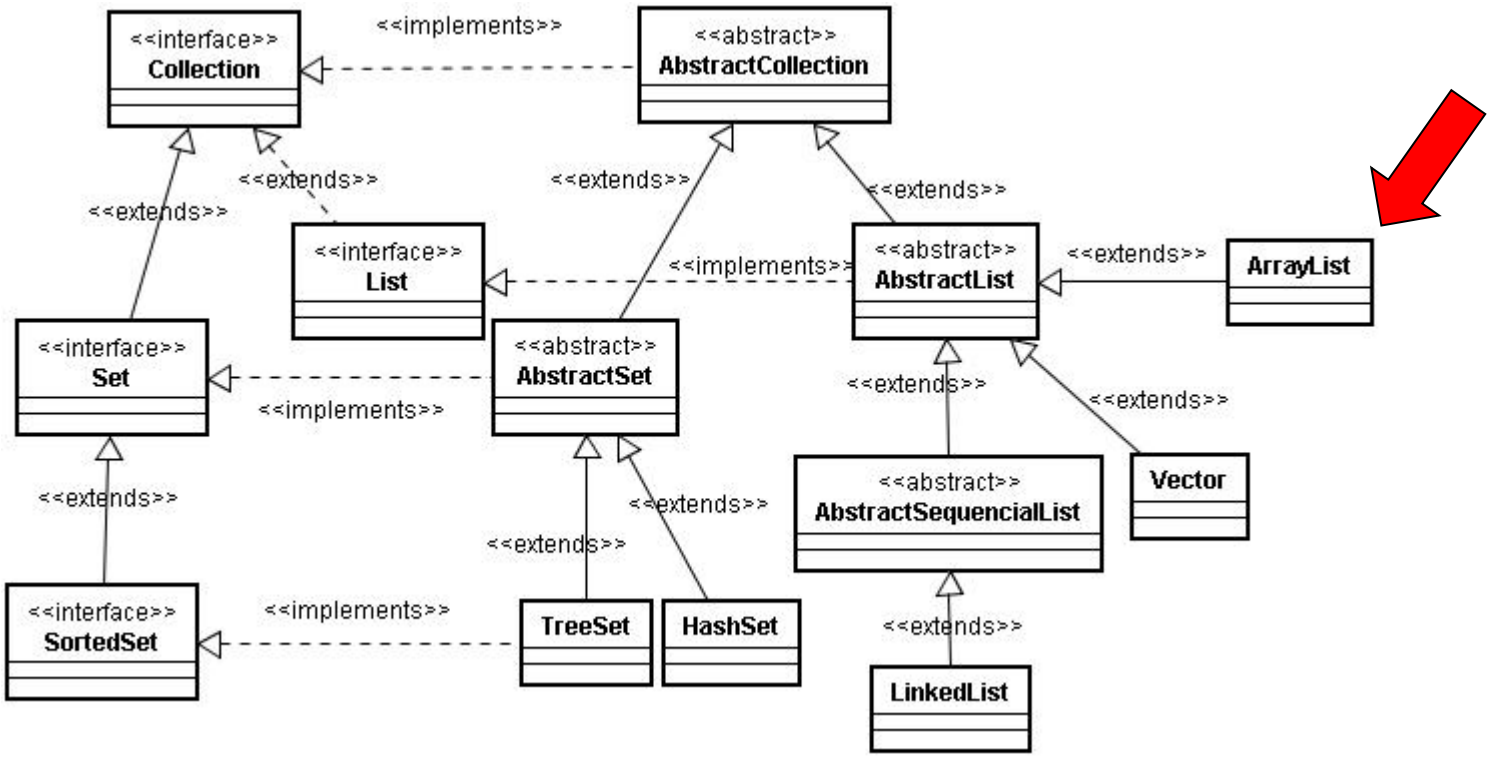
- Exemplos de tipos genéricos são encontrados na **Collections Framework Java SE 5.0**.
- A necessidade de tipos genéricos surgiu na implementação e uso de **collections**.
- Uma coleção sempre contém elementos de um determinado tipo, tais como uma lista de inteiros, ou uma lista de Strings, etc.
- **ArrayList** é uma classe genérica com um type parameter que pertence ao Framework.
- **ArrayList** implementa uma estrutura de dados que modifica dinamicamente o seu tamanho em tempo de execução.

Interfaces

- Encapsulam diferentes tipos de **collections**.
- Permitem que as **collections** sejam manipuladas independentemente dos detalhes de como foram implementadas.
- Representam a base da Java Collections Framework.



Classes e interfaces que estendem ou implementam a interface Collection



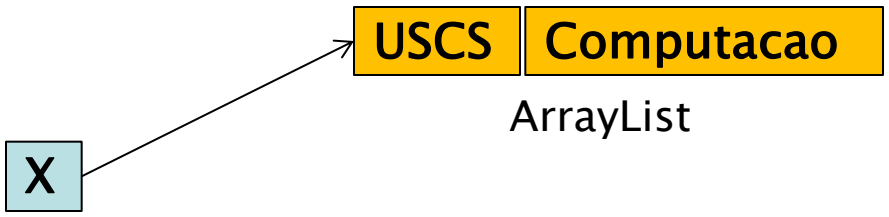
Classe ArrayList

- ArrayList é uma classe concreta da API Framework Collections.
- Diferentemente da estrutura Array que tem tamanho fixo, um arraylist é um objeto que pode modificar seu tamanho e, portanto, é adequado para situações onde se necessita de comportamento dinâmico.
- Assim, um ArrayList tem o mesmo propósito de um Array, mas seu tamanho pode se modificar em tempo de execução.

ArrayList – método add

boolean	add(E e)Appends the specified element to the end of this list.

```
package oop;

import java.util.ArrayList;

public class ArrayList_01 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                System.out.println(x.toString()) ;
        }
}
```

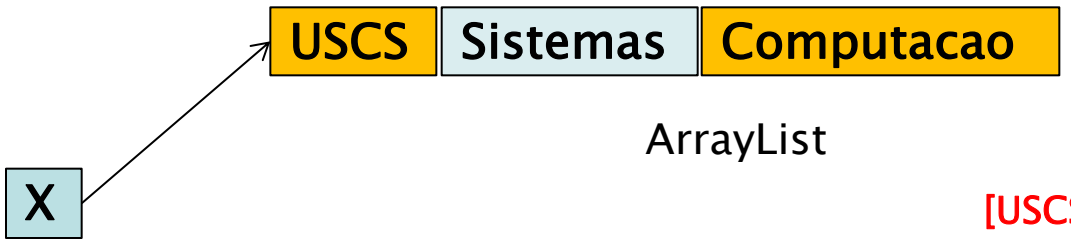


[USCS, Computacao]

ArrayList – método add

boolean	add(E e)Appends the specified element to the end of this list.

```
package oop;
import java.util.ArrayList;

public class ArrayList_02 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                x.add(1,"Sistemas");
                System.out.println(x.toString()) ;
        }
}
```

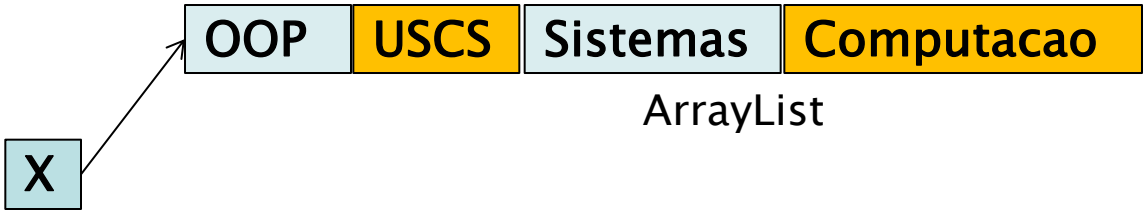


[USCS, Sistemas, Computacao]

ArrayList – método add

void	add(int index, E element)Inserts the specified element at the specified position in this list.

```
package oop;
import java.util.ArrayList;

public class ArrayList_03 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                x.add(1,"Sistemas");
                x.add(0,"OOP");
                System.out.println(x.toString()) ;
        }
}
```

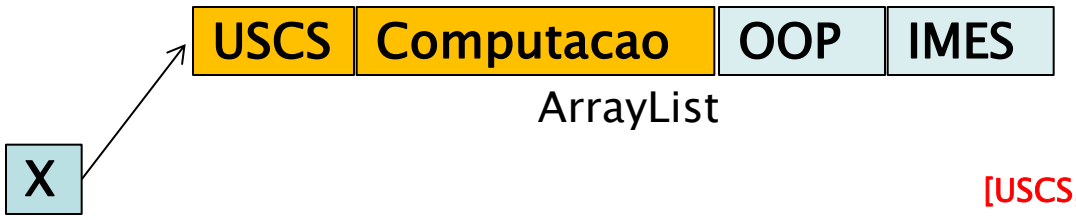


[OOP, USCS, Sistemas, Computacao]

ArrayList – método addAll

boolean	addAll(Collection<? extends E> c)Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.

```
package oop;
import java.util.ArrayList;

public class ArrayList_04 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                ArrayList<String> y = new ArrayList<String>();
                y.add("OOP");
                y.add("IMES");
                x.addAll(y);
                System.out.println(x.toString()) ;
        }
}
```

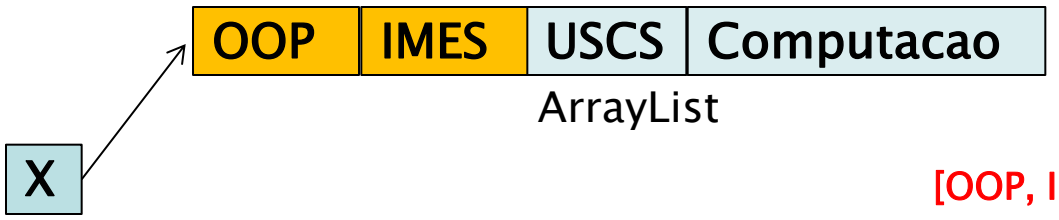


[USCS, Computacao, OOP, IMES]

ArrayList – método addAll

boolean	addAll(Collection<? extends E> c)Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.

```
package oop;
import java.util.ArrayList;

public class ArrayList_05 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                ArrayList<String> y = new ArrayList<String>();
                y.add("OOP");
                y.add("IMES");
                y.addAll(x);
                System.out.println(y.toString()) ;
        }
}
```

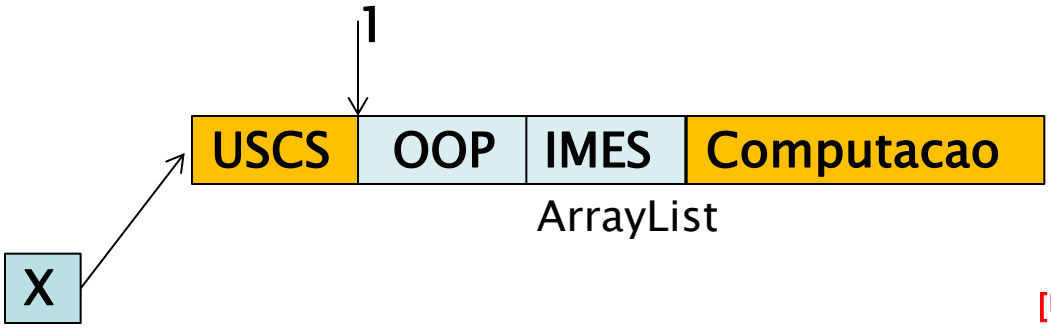


[OOP, IMES, USCS, Computacao]

ArrayList – método AddAll

boolean	addAll(int index, Collection<? extends E> c)Inserts all of the elements in the specified collection into this list, starting at the specified position.

```
package oop;
import java.util.ArrayList;

public class ArrayList_06 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                ArrayList<String> y = new ArrayList<String>();
                y.add("OOP");
                y.add("IMES");
                x.addAll(1,y);
                System.out.println(x.toString()) ;
        }
}
```

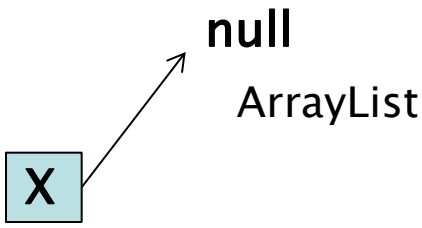


[USCS, OOP, IMES, Computacao]

ArrayList – clear

void	clear()Removes all of the elements from this list.

```
package oop;
import java.util.ArrayList;

public class ArrayList_07 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                x.clear();
                System.out.println(x.toString()) ;
        }
}
```



[]

ArrayList – método contains

boolean	contains(Object o)Returns true if this list contains the specified element.

```
package oop;
import java.util.ArrayList;

public class ArrayList_08 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("USCS");
                if (x.contains ("USCS") )
                        System.out.println(" x contem USCS ");
        }
}
```

x contem USCS

ArrayList – método get

E	get(int index)Returns the element at the specified position in this list.

```
package oop;
import java.util.ArrayList;

public class ArrayList_09 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("C");
                for (int i=0; i<3 ; i++)
                        System.out.print(x.get(i) + "  "  );
        }
}
```

A B C

ArrayList – método indexOf

int	indexOf(Object o)Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

```
package oop;
import java.util.ArrayList;

public class ArrayList_10 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("A");
                System.out.print(x.indexOf("A") + "  "  );
                System.out.print(x.indexOf("B") + "  "  );
        }

}
```

```
0  1
```

ArrayList – método isEmpty

boolean	isEmpty()Returns true if this list contains no elements.

```
package oop;
import java.util.ArrayList;

public class ArrayList_11 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                ArrayList<String> y = new ArrayList<String>();
                if ( ! x.isEmpty() )
                        System.out.println(" x tem dados ");
                if ( y.isEmpty() )
                        System.out.println(" y vazio ");

        }

}
```

x tem dados
y vazio

ArrayList – método lastIndexOf

int	lastIndexOf(Object o)Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

```
package oop;
import java.util.ArrayList;

public class ArrayList_12 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("A");
                System.out.print(x.lastIndexOf("A") + "  "  );
                System.out.print(x.lastIndexOf("B") + "  "  );
                System.out.print(x.lastIndexOf("C") + "  "  );

        }

}
```

2 1 -1

ArrayList – método remove

E	remove(int index)Removes the element at the specified position in this list.

```
package oop;
import java.util.ArrayList;

public class ArrayList_13 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();

                x.add("A");
                x.add("B");
                x.add("C");
                x.remove(0);
                x.remove(1);

                System.out.print(x.toString());
        }

}
```

[B]

ArrayList – método remove

boolean	remove(Object o)Removes the first occurrence of the specified element from this list, if it is present.

```
package oop;
import java.util.ArrayList;

public class ArrayList_14 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();

                x.add("A");
                x.add("B");
                x.add("A");
                x.remove("A");
                x.remove("B");

                System.out.print(x.toString());
        }

}
```

[A]

ArrayList – método removeAll

boolean	**removeAll(Collection<?> c)Removes from this list all of its elements that are contained in the specified collection.**

```
package oop;
import java.util.ArrayList;

public class ArrayList_15 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("C");
                ArrayList<String> y = new ArrayList<String>();
                y.add("A");
                y.add("B");
                x.removeAll(y);
                System.out.println(x.toString()) ;
        }
}
```

[C]

ArrayList – método retainAll

boolean	retainAll(Collection<?> c)Retains only the elements in this list that are contained in the specified collection.

```
package oop;
import java.util.ArrayList;

public class ArrayList_16 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("C");
                x.add("A");
                x.add("D");

                ArrayList<String> y = new ArrayList<String>();
                y.add("A");
                y.add("B");
                x.retainAll(y);
                System.out.println(x.toString()) ;
        }
}
```

[A, B, A]

ArrayList – método set

E	set(int index, E element)Replaces the element at the specified position in this list with the specified element.

```
package oop;
import java.util.ArrayList;

public class ArrayList_17 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();
                x.add("A");
                x.add("B");
                x.add("C");
                x.add("A");
                x.add("D");

                for (int i=0; i<5; i++)
                    if (i%2 == 0)
                         x.set(i , "X");
                System.out.println(x.toString()) ;
        }
}
```

[X, B, X, A, X]

ArrayList – método size()

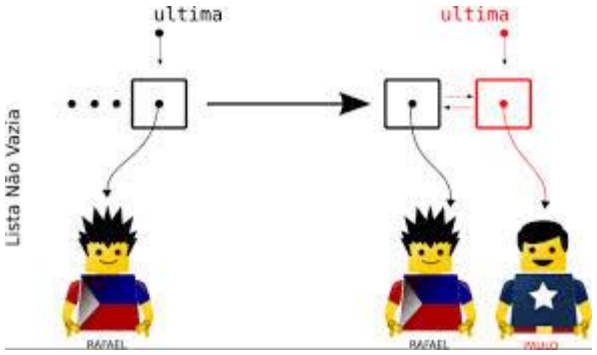
int	size()Returns the number of elements in this list.

```
package oop;
import java.util.ArrayList;
public class ArrayList_18 {

        public static void main(String[] args) {
                ArrayList<Integer> x = new ArrayList<Integer>();
                x.add(10);
                x.add(20);
                x.add(30);
                System.out.println(x.toString());
                System.out.println(x.size());
                ArrayList<String> y = new ArrayList<String>();
                y.add("São Paulo");
                y.add("Corinthians");
                System.out.println(y.toString());
                System.out.println(y.size());
        }
}
```

```
[10, 20, 30]
3
[São Paulo, Corinthians]
2
```

Percorrendo os elementos...

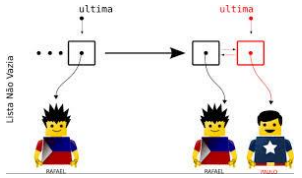
- Há duas formas de se percorrer (atravessar) os elementos de uma collection.
- Primeiro: com o construto **for-each**.
- Segundo: Por meio de **Iterators**.



for - each

```
package oop;

import java.util.ArrayList;
public class ArrayList_19 {

        public static void main(String[] args) {

                ArrayList<String> x = new ArrayList<String>();

                x.add("USCS");
                x.add("Computacao");

                for (String el: x)
                        System.out.println(el);
        }
}
```

USCS
Computacao



convertendo para array

Object[]	toArray()Returns an array containing all of the elements in this list in proper sequence (from first to last element).

```
package oop;

import java.util.ArrayList;

public class ArrayList_20 {

        public static void main(String[] args) {
                ArrayList<String> x = new ArrayList<String>();
                x.add("USCS");
                x.add("Computacao");
                String[] array_x = new String[x.size()];
                array_x = x.toArray(array_x);
                for (int i=0; i<array_x.length; i++)
                        System.out.println(array_x[i]);
        }
}
```

USCS
Computacao