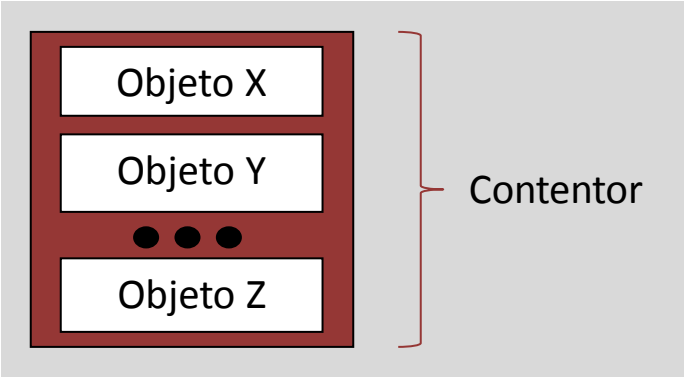
Linguagem JAVA

Contentores de Objetos

Sumário

Noção de Contentor

- **Estrutura de Dados**
  - Permite armazenar múltiplos objetos



Categorias de Contentores

Tipo	Características	Exemplo
Conjunto (Set)	Implementa um conjunto matemático finito: ▪ Não há noção de ordem (posição): 1º, 2º, n-ésimo ou último elemento ▪ Não são permitidos elementos repetidos	Conjunto de artigos científicos
Lista (List)	Implementa uma sequência: ▪ Há noção de ordem entre elementos ▪ São permitidos elementos repetidos	Pasta de correio electrónico (mensagens guardadas pela ordem de chegada)
Fila de Espera (Queue)	Destinada a guardar elementos à espera de serem processados A ordem de processamento é do tipo FIFO	Conjunto de pedidos de serviço recebidos por um servidor
Pilha (Stack)	Destinada a guardar elementos à espera de serem processados A ordem de processamento é do tipo LIFO	Conjunto de endereços navegados num browser Web
Mapeamento (Map)	Implementa correspondências unívocas (1 para 1) entre objetos do tipo chave-valor As chaves são o domínio das correspondências e são únicas	Lista Telefónica

Sumário

Interesse dos Arrays

- **Guardar uma lista de objetos**
  - Objetos todos do mesmo tipo (ou compatíveis) // ex: só objetos Automovel ou Livro ou Pessoa
  - Lista de dimensão fixa
- **Exemplos**

Tabela Unidimensional
(Dimensão N)

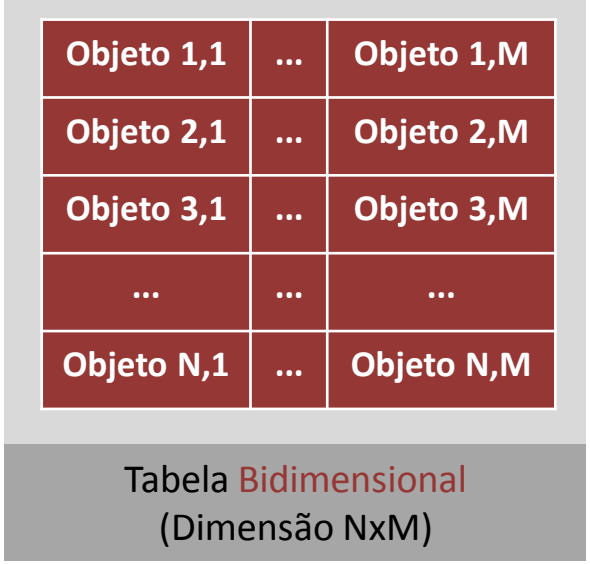


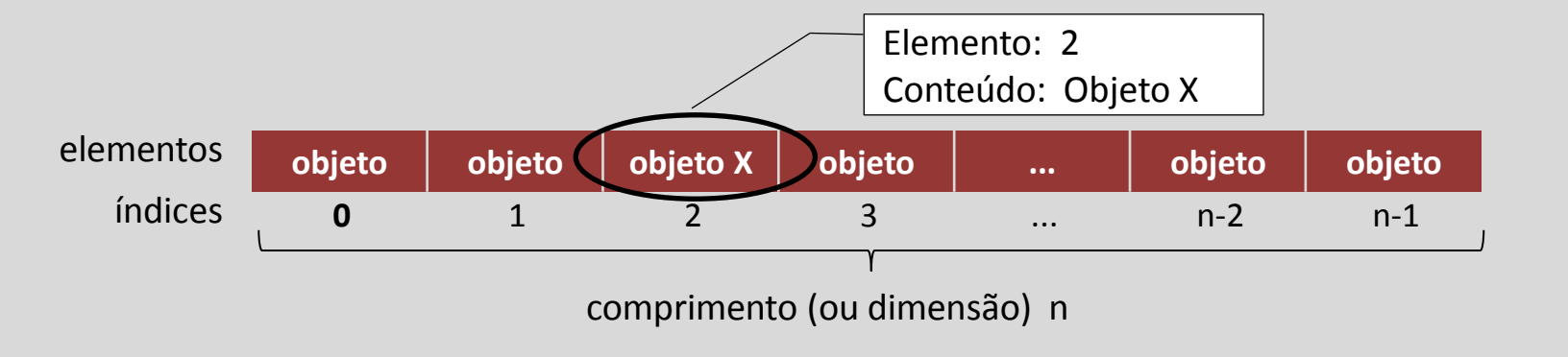
Tabela Bidimensional
(Dimensão NxM)

Categorias de Arrays

- **Unidimensionais**
- **Multidimensionais**
  - Bidimensionais
  - Tridimensionais
  - ...

Noção de Array Unidimensional

- **Array**
  - Constituído por elementos
  - Nº de elementos (comprimento) é fixo // Dimensão não modificável em tempo de execução
  - Elementos
    - Organizados de forma linear
    - Funcionam como variáveis simples
    - Podem armazenar objetos
      - Todos do mesmo tipo (ou tipos compatíveis)
    - Acesso através de índices
      - Índice
        - Indica posição de um elemento
        - Nº inteiro: [0, comprimento -1]



Uso

- **Preciso Saber**
  - Declarar um array
  - Java
    - Arrays são objetos
  - Manipular elementos do array

Declaração de Array Unidimensional

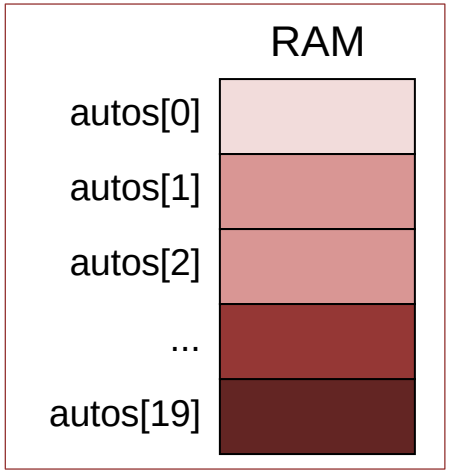
- **Java**

```
tipo  nomeArray[ ] = new  tipo[ dimensão ];
tipo[ ]  nomeArray = new tipo[ dimensão ];

tipo nomeArray[ ];
nomeArray = new tipo[ dimensão ];
```

- Array é **objeto**
- **Nome** do array é **referência** de objeto

- **Exemplos**

```
Automovel  autos[ ] =  new  Automovel [20];
Automovel[ ]  autos =  new  Automovel [20];

Automovel autos [ ];
autos[ ] = new Automovel[20];
```



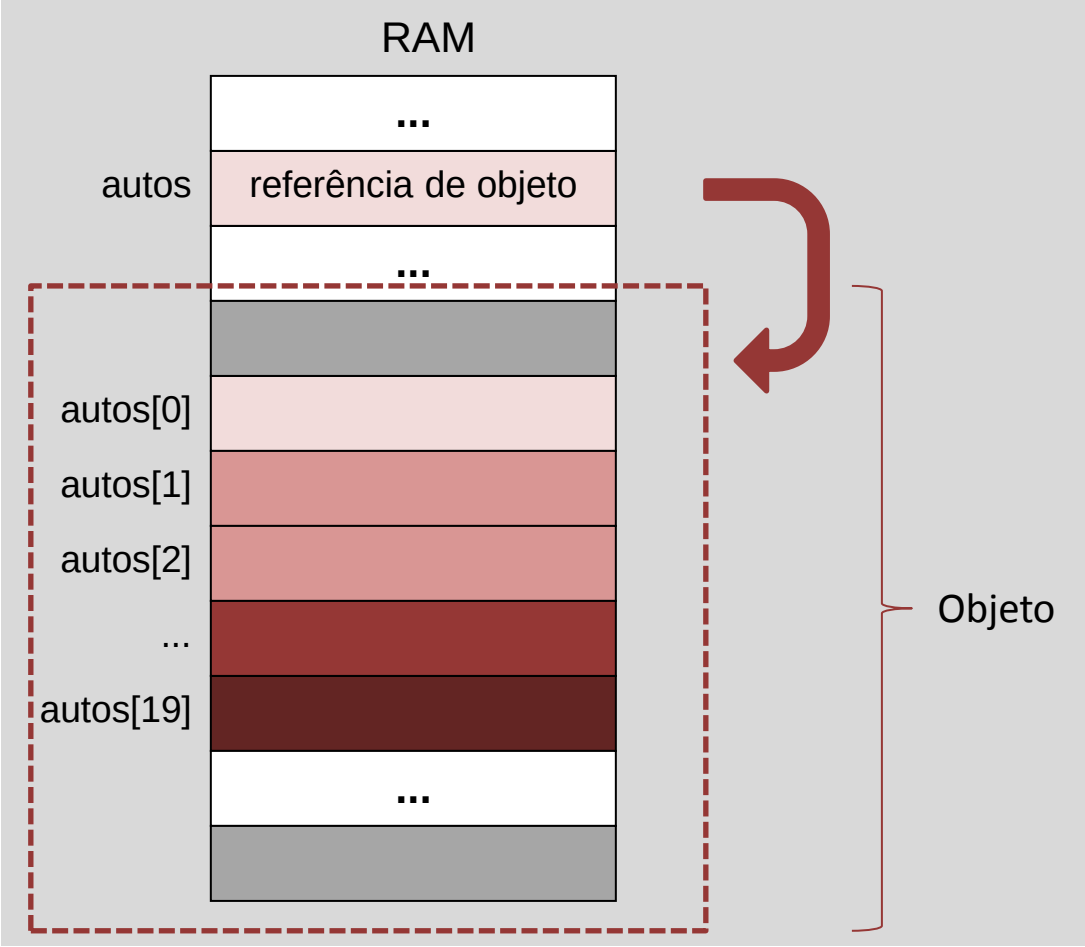
19 = 20-1

- **Inicialização dos Elementos**

Tipo referência: null (Ex: Automovel)

Java: Array é um Objeto

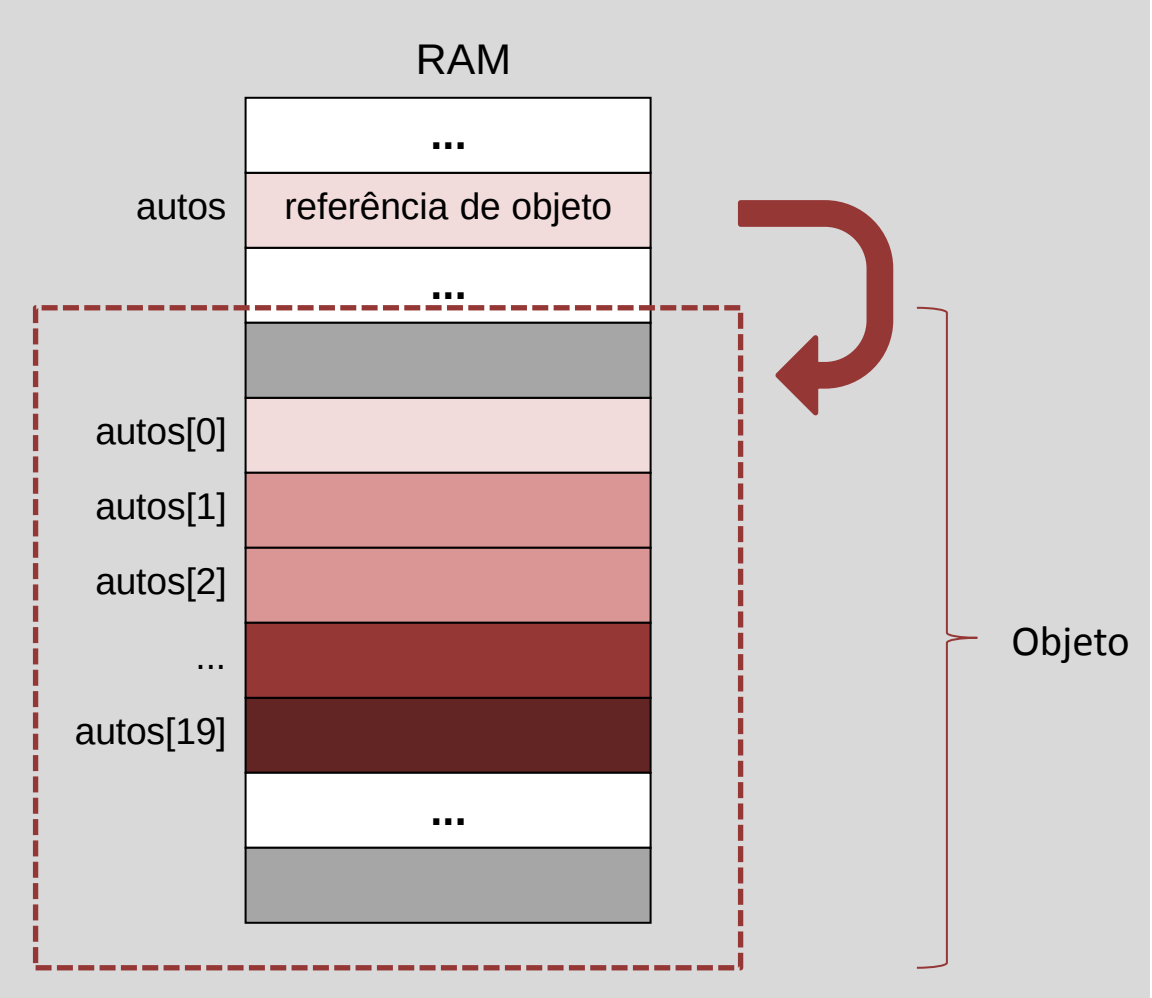
- **Nome de Array**
  - É uma referência do objeto que contém os seus elementos // referência = endereço
- **Exemplo**
  - Automovel[] autos = new Automovel[20];



Manipulação de Elementos

- **Elemento**
  - Pode ser manipulado individualmente
  - Funciona como uma variável simples
  - Identificado pelo
    - Nome do vetor
    - Índice respetivo
- **Indicar Elemento**
  - nomeArray[índice]
  - Exemplo

    autos[2]
- **Manipulações Típicas**
  - Um elemento
  - Todos elementos



Manipulação de um Elemento

- **Atribuir objeto a um elemento**
  - Guardar ou atualizar um elemento
  - Sintaxe: nomeArray[índice] = objeto;

    Ex: autos[5] = new Automovel("Toyota");

- **Atribuir conteúdo de elemento a uma variável**
  - Sintaxe: variável = nomeArray[índice];

    Ex: Automovel a = autos[5];

Manipulação de Todos os Elementos

- **Indicar todos os n elementos** (n = comprimento do array)

```
for(int i=0;  i < nomeArray.length; i++ ){
    ... nomeArray[i] ...
}
```

nomeArray.**length** (atributo)

- **Ex: Guardar no array n instâncias Automovel**

```
for(int i=0;  i<autos.length; i++){
    autos[i] = new Automovel();
}
```

- **Ex: Mostrar as instâncias Automovel guardadas no array**

```
for(int i=0;  i<autos.length; i++){
  if ( autos[i] != null )
     System.out.println( autos[i] );
}
```

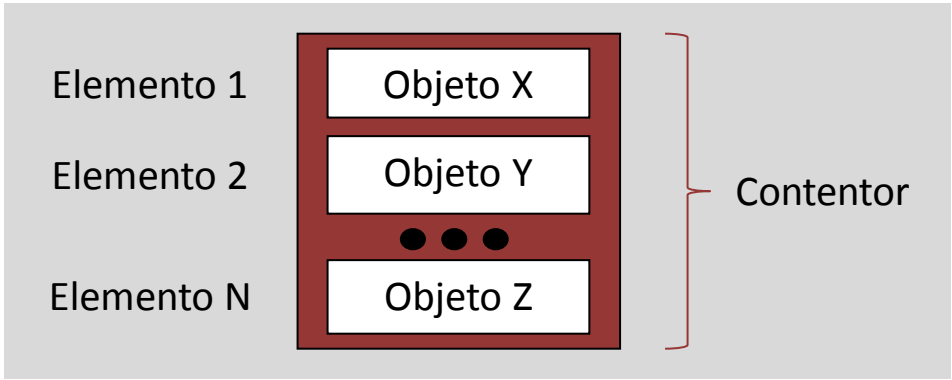
```
// Alternativa com foreach
for(Automovel a: autos)
  if ( a != null )
     System.out.println(a);
}
```

```
Sintaxe da repetição foreach:
for( Tipo_Elemento variável: nomeArray ){
    instruções    // sobre variável
}
```

Sumário

Noção de ArrayList

- **Classe**
  - Instanciável
  - Uma instância
    - Contentor de objetos
      - Tipo lista
        - Há ordem nos objetos (1º, 2º, ..., N)

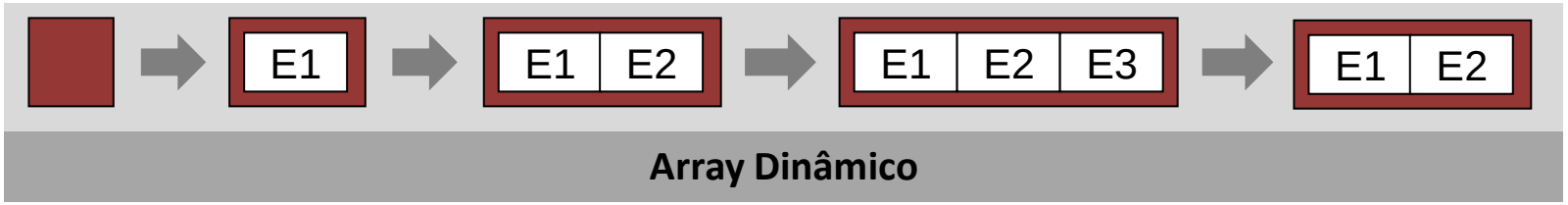


Elementos contêm Objetos

    - Fornece serviços para gerir objetos
      - Exemplos
        - Adicionar objetos
        - Remover objetos
  - Tipo coleção
    - Implementa interface *Collection*

Implementação de um ArrayList

- **ArrayList Implementa**
  - **Array dinâmico** // nº de elementos pode variar durante execução de programa
    - Cresce // adicionando novos objetos
    - Decresce // removendo objetos



  - **Baseado num array**

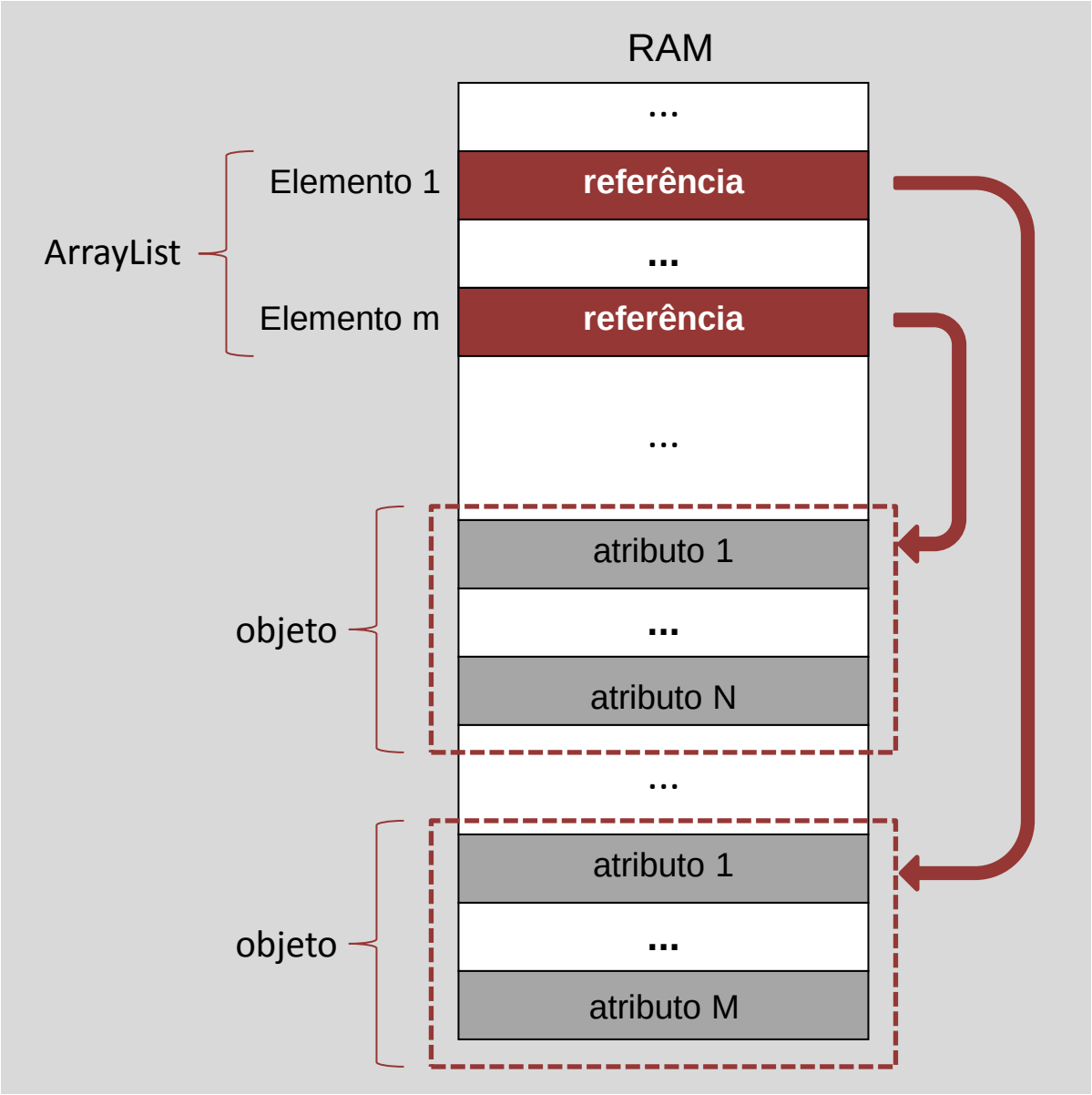
```
public class ArrayList<E> extends AbstractList<E>
        implements List<E>, RandomAccess, Cloneable, java.io.Serializable
{
    private static final long serialVersionUID = 8683452581122892189L;

    /**
     * The array buffer into which the elements of the ArrayList are stored.
     * The capacity of the ArrayList is the length of this array buffer.
     */
    private transient Object[] elementData;

    /**
     * The size of the ArrayList (the number of elements it contains).
     *
     * @serial
     */
    private int size;
```

Implementação de um ArrayList

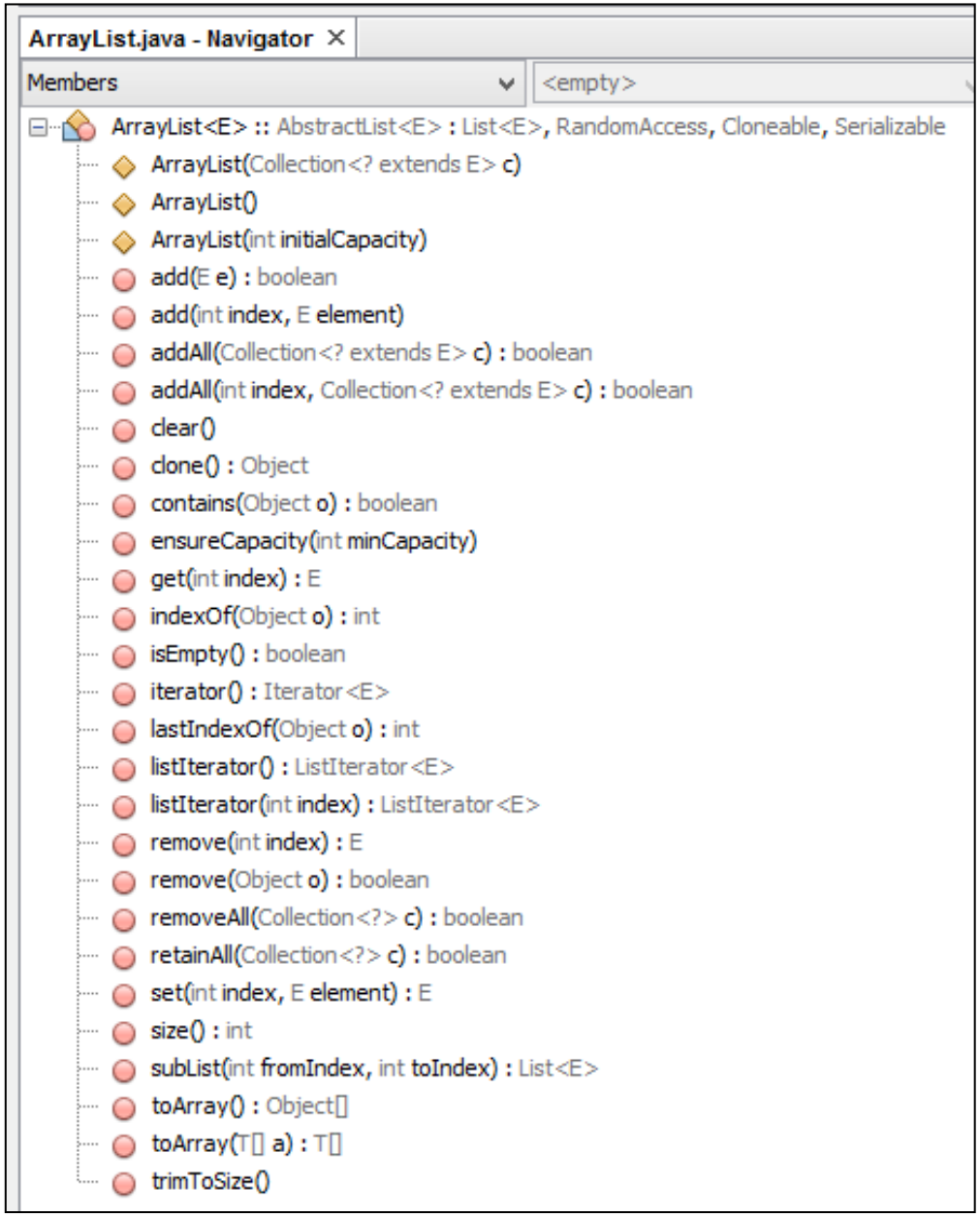
- **Elementos**
  - Tipo Object
    - Compatível com **todos** os tipos
  - Armazenam **qualquer** objeto

- **Estrutura de dados indexada**
  - Semelhante ao array
  - Índice
    - Indica posição dos elementos
    - Número inteiro desde zero

Modelo de Memória



API (Application Programming Interface)



Construtores

Métodos de Instância

- **Construtores**
  - public ArrayList() // tamanho inicial zero (nº de objetos adicionados)
    // capacidade inicial 10 (nº de elementos alocados)
    // primeiras 10 adições rápidas (s/ custos realocação)
  - public ArrayList(int capacidade_inicial); // tamanho inicial zero
  - public ArrayList(Collection<? extends E> c); // permite **copiar** *arrayList* recebido por parâmetro
    // há **partilha** de objetos

```
public class ExemploArrayList {

   public static void main(String[] args) {
        ...
        ArrayList plantel_1 = new ArrayList();        // declaração e instanciação
        ...
        ArrayList plantel_2 = new ArrayList(25);
        ...
        ArrayList plantel_3 = new ArrayList(plantel_2);
   }
}
```

- **Métodos para adicionar objetos**
  - boolean **add**(Object obj) — adiciona obj no final da lista e incrementa tamanho de 1 unidade; retorna *true* (sucesso) ou *false* (insucesso); pode ser adicionado *null*
  - void **add**(int índice, Object obj) — adiciona obj na posição índice; desloca uma posição à direita objetos, desde a posição índice
  - Object **set**(int índice, Object obj) — adiciona obj na posição índice; se estiver ocupada, objeto atual é substituído; se índice >= size() ou < 0, é gerado um erro de execução

```
public class ExemploArrayList {
   public static void main(String[] args) {
        ArrayList plantel = new ArrayList();
        plantel.add( "Nico" );
        plantel.add( "Bruno" );
        plantel.add( 1, "Artur" );
        plantel.set( 1, "Eduardo" );
   }
}
```

- **Métodos para remover objetos**
  - void **clear**() — remove todos os objetos da lista (tamanho=0, capacidade =)
  - Object **remove**(int índice) — remove objeto na posição índice; desloca objetos de índice superior para índice imediata/ inferior
  - boolean **remove**(Object obj) — remove a 1ª ocorrência de obj na lista, caso exista; desloca objetos de índice superior para índice imediata/ inferior

```
public class ExemploArrayList {
   public static void main(String[] args) {
        ArrayList plantel = new ArrayList();
        ...
        plantel.remove(2);
        plantel.remove("Bruno");

        plantel.clear();
   }
}
```

- **Métodos para pesquisar objetos**
  - boolean **isEmpty**() — retorna *true* se lista estiver vazia; caso contrário, retorna *false*
  - boolean **contains**(Object obj) — retorna *true* se obj estiver na lista; caso contrário, retorna *false* usa o método *equals* de obj
  - int **indexOf**(Object obj) — retorna o índice da 1ª ocorrência de obj na lista, caso exista caso não exista, retorna -1; usa o método equals de obj
  - int **lastIndexOf**(Object obj) — semelhante ao anterior, mas relativo à última ocorrência

```
public class ExemploArrayList {
   public static void main(String[] args) {
        ArrayList plantel = new ArrayList();
        ...
        System.out.println( plantel.isEmpty() ? "Não Há Jogo" : "Há Jogo" );
        System.out.println( plantel.contains("Artur") ? "Vence" : "Perde" );
        System.out.println( "Posição= " + plantel.indexOf("Artur") );
   }
}
```

- **Outros métodos**
  - int **size**() — retorna nº de objetos adicionados à lista
    diferente de capacidade
  - Object **get**(int índice) — retorna objeto guardado na posição índice
  - Object[] **toArray**() — retorna *array* contendo todos os objetos do *arrayList*
    mantém ordem dos objetos

```
ArrayList nomes = new ArrayList();
...
for( int i=0; i < nomes.size(); i++ ){        // tradicional ciclo for sobre arrays
   if( nomes.get(i) != null )
         System.out.println( nomes.get(i) )
}
...
Object[] nomes = nomes.toArray();
```

Iterações

- **Operações de iteração (varrimento)**
  - Para percorrer todos os elementos da lista
- **Formas**
  - Ciclo for sobre o índice da lista

```
ArrayList nomes = new ArrayList();
...
for( int i=0; i < nomes.size(); i++ ){     // tradicional ciclo for sobre arrays
   if( nomes.get(i)!= null )
         System.out.println( nomes.get(i) )
}
```

  - Repetição foreach sobre o ArrayList
    - Sintaxe

```
for( Tipo_Elemento variável: nomeArrayList ){
    instruções     // sobre variável
}
```

    - Exemplo

```
for( Object obj : nomes ){              // lê-se: para cada obj da lista nomes faz
    if( obj!=null )                     // arrayList pode ter elementos null
         System.out.println( obj );
}
```

Exemplo

```
public class ExemploArrayList {
   public static void main(String[] args) {
       ArrayList plantel = new ArrayList();
       plantel.add( "Nico" );
       plantel.add( "John" );
       plantel.add( "Cardoso" );
       for (Object obj : plantel) {
          if( obj!=null )
             System.out.println(obj);      // obj = obj.toString()
       }
       System.out.println( "Tamanho do arraylist: " + plantel.size() );
       System.out.println( "2° jogador:" + plantel.get( 1 ) );
       plantel.set( 1, "Salvio" );       // Substitui o 2° jogador
       plantel.remove( 0 );              // Remove 1° jogador
       plantel.remove( "Cardoso" );      // Remove jogador passado por parâmetro
       if( plantel.contains( "Eusébio" ) )
          System.out.println( "Eusébio faz parte do plantel" );
       else
          System.out.println( "Eusébio não faz parte do plantel" );
   }
}
```

Sumário

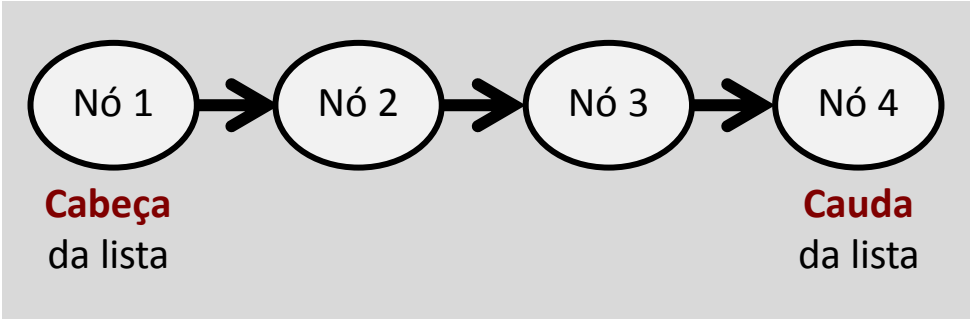
- Noção de Contentor
- Categorias de Contentores
  - Conjunto
  - Lista
  - Fila de Espera
  - Pilha
  - Mapeamento
- Exemplos
  - Array
  - ArrayList
  - Lista Ligada

Noção de Lista Ligada

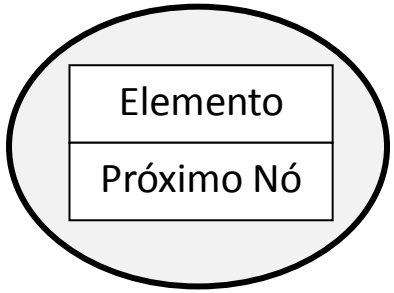
- **Contentor do tipo Lista**
  - Permite guardar uma sequência de objetos
  - Estabelece relação de ordem entre objetos
    - 1º objeto, último objeto, n-ésimo objeto, etc.
  - Permite objetos repetidos

Implementação de uma Lista Ligada

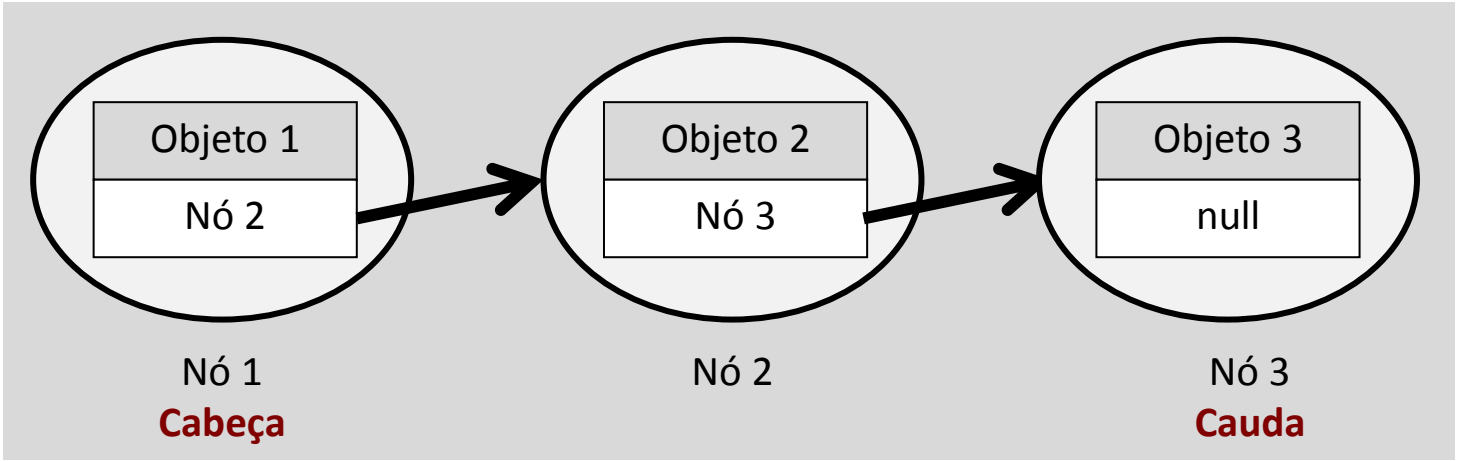
- **Constituída por sequência de nós**
  - Nós interligados



  - Cada nó guarda
    - Um elemento (objeto) da lista // Java: tipo Object
    - Referência do próximo nó

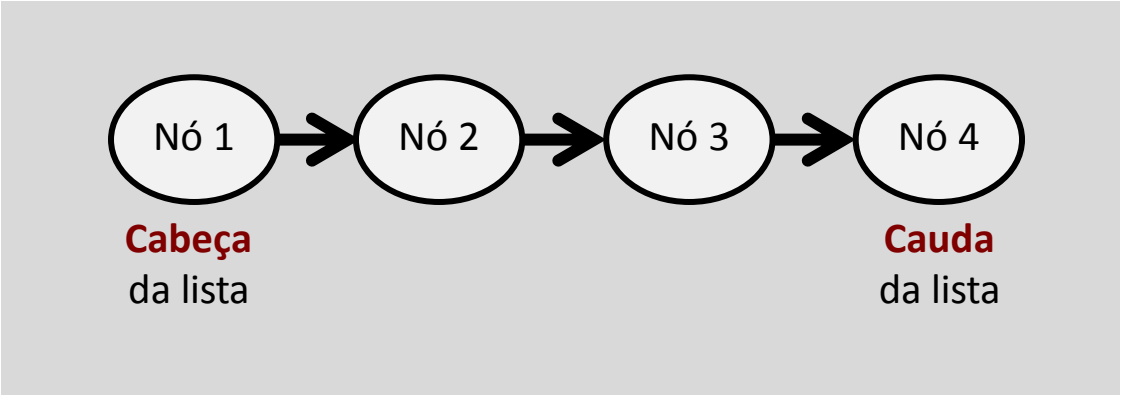


- **Exemplo**



Serviços Típicos de uma Lista Ligada

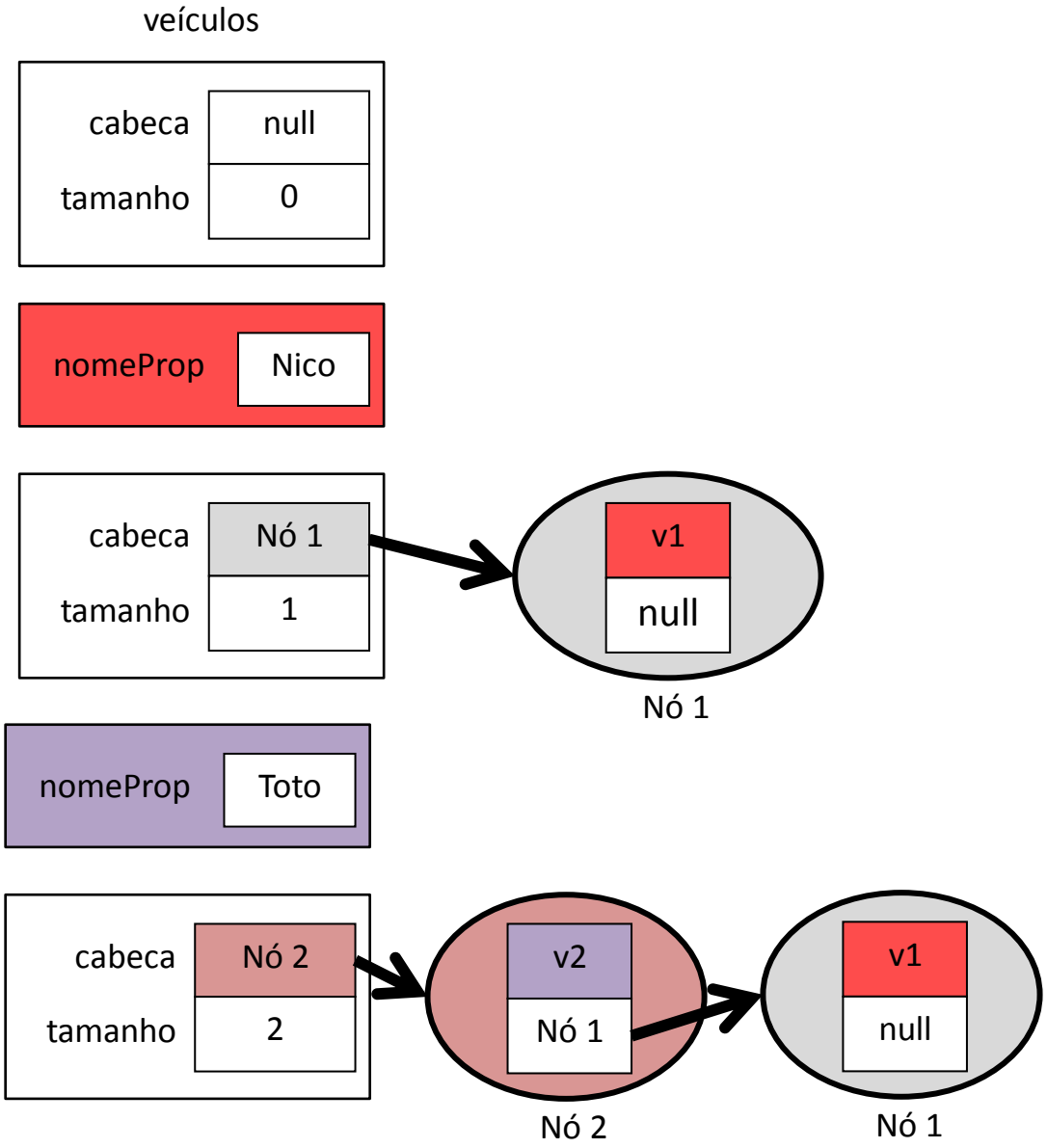
- **Inserir objeto**
  - Na cabeça da lista
  - Na cauda da lista
  - Na n-ésima posição
- **Remover objeto**
  - Na cabeça
  - Na cauda
  - Na n-ésima posição
- **Procurar objeto**



- **NOTA**
  - Implementação da classe Lista Ligada é transparente para o utilizador dela
    - Utilizador não precisa de saber que objetos são guardados em nós
    - Métodos públicos só passam, por parâmetro, os objetos a armazenar

Exercício da Aula Prática

```
public class TesteListaLigada {
   public static void main(String[] args) {
      ListaLigada veiculos = new ListaLigada();

      Veiculo v1 = new Veiculo("Nico");

      veiculos.inserirACabeca(v1);

      Veiculo v2 = new Veiculo("Toto");

      veiculos.inserirACabeca(v2);

      ....
```



Exercício da Aula Prática

```
public class TesteListaLigada {

   public static void main(String[] args) {
      ...
       veiculos.removerACabeca();




      veiculos.removerACabeca();


       ....
   }
}
```