

Linguagem de Programação II

Arrays

Universidade do Estado do Rio de Janeiro-UERJ
Instituto de Matemática e Estatística-IME
Ciência da Computação
Professor: Alexandre Sztajnberg

Array

- ☐ Array é um tipo composto do Java
 - ☐ Tem tratamento diferenciado
- ☐ São estruturas que armazenam uma coleção de dados, homogênea, de um mesmo tipo
 - Isso fica um pouco “diferente” com herança e polimorfismo
- ☐ Arrays são criados com um tamanho fixo definido em sua declaração
- ☐ Os elementos são acessados por índice, começando a contagem pelo índice zero
- ☐ Arrays têm um campo chamado `length`, do tipo *int*, inicializado com o valor correspondente ao tamanho do array instanciado
 - ☐ Para percorrer todo os elementos do array (0 à `length-1`)
- ☐ Tem um método `toString()` que exibe os elementos do array???
 - ☐ A turma deve confirmar isso ... Não exibe os elementos ...

Declaração

```
<tipo/classe>[] nomeReferência;  
nomeReferência = new <tipo/classe>[literal/constante/expressão-inteira];
```

- ❑ O nome de uma referência a um array deve seguir o padrão Camel Case
- ❑ A referência aponta para o primeiro elemento do array (elemento zero)

Referência

```
double[] salesAmt;  
salesAmt = new double[6];
```



Variações

Lista de inicialização

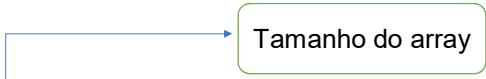
- ❑ `int[] matricAluno = {201101, 201102, 201532, 201902, 201602}`
 - Tamanho 5 (`matricAluno.length == 5`), 5 posições (ou *slots*)
 - `matricAluno[0]` a `matricAluno[4]`
- ❑ `double[] notaAluno = {10.0, 6.7, 7.5, 8.0, 4.3}`
 - Inicialização com lista
 - Define o tamanho (`notaAluno.length == 5`) e o conteúdo inicial

String

- ❑ `String[] nomeAluno = new String[10]`
 - Tamanho 10, 10 posições (ou *slots*)
 - De 0 a 9
- ❑ `String[] nomeAluno = {"Caio", "Ciro", "Alex", "Marcia", "Mateus", "Ana"};`
 - Inicialização com lista
 - Define o tamanho (7) e o conteúdo inicial

Exemplo

```
public class Vetores1{
    public static void main(String[] args){
        int[] notas = new int[8];
        String[] nomes = {"Caique", "Caio", "Carlos", "Alexandre", "Matheus", "Mateus", "Alan"};
        notas[0]=9;
        notas[1]=8;
        notas[2]=7;
        notas[0]=6;
        notas[3]=9;
        notas[4]=10;
        notas[5]=10;
        notas[6]=8;
        notas[7]=9;
        for(int i =0; i< notas.length; i++){
            System.out.println(nomes[i]);
            System.out.println(notas[i]);
        }
    }
}
```



Tamanho do array

For-each

❑ *for* especial

- ❑ percorre um Array ou um objeto cuja classe implemente a interface *Iterable*
 - ❑ List e outras classes relacionadas (e herdeiros)
- ❑ Otimizado para esse tipo de função
- ❑ Ao percorrer iterativamente o array
 - ❑ Retorna o valor ou referência de cada item em sequência a cada passada
 - ❑ Atribuindo o mesmo à uma variável criada dentro da declaração do for

```
for(double nota: notaAluno) {  
    // atribui o valor ou referência nota a cada loop  
}
```

- **notaAluno**: array percorrido ... `notaAluno[0] ... notaAluno[notaAluno.length -1]`
- **nota**: deve ser do mesmo tipo do array (ou dá erro de compilação ... **Salvo nos casos de polimorfismo** ...)

Exemplo for-each

```
public class ForEach {  
    public static void main(String[] args) {  
        int[] numbers = {3, 9, 5, -5};  
        for (int number: numbers) {  
            System.out.println(number);  
        }  
    }  
}
```

Array como parâmetro de entrada de método

- ❑ Quem “vai” é a referência!

```
public static double average(int[] grades)
// Calculates and returns the average grade in an
// array of grades.
// Assumption: All array slots have valid data.
{
    int total = 0;
    for (int i = 0; i < grades.length; i++)
        total = total + grades[i];
    return (double) total / (double) grades.length;
}
```

Array como retorno do método

```
public static double[] mediaPer(double[] P1, double P2)
// Recebe dois arrays, com as notas da P1 e P2 dos alunos
// da turma de LP II. Devolve a media do período
// Assumption: All array slots have valid data. Por que??
{ // Vamos completar juntos!!!
    double
    for (int i = 0; i < ; i++)

    return ;
}
```

Exemplo de programa

❑ Contar o número de vezes que cada letra aparece num arquivo de texto

- Um truque para armazenar a contagem
- Outro para identificar cada letra

letter	ASCII
'A'	65
'B'	66
'C'	67
'D'	68
⋮	⋮
⋮	⋮
⋮	⋮
'Z'	90

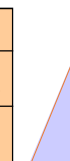
datafile.dat

This is my text file.
It contains many
things!
5 + 8 is not 14.
Is it?

Exemplo de programa

❑ Um array para guardar a contagem ...

- `int[] letterCount = new int[26];`

letterCount [0]	2		
letterCount [1]	0		counts 'A' and 'a'
			counts 'B' and 'b'
.	.		.
.	.		.
.	.		.
letterCount [24]	1	counts 'Y' and 'y'	
letterCount [25]	0	counts 'Z' and 'z'	

Pseudo código

```

Prepare dataFile
Read one line from dataFile
While not EOF on dataFile
    For each letter in the line
        If letter is an alphabetic character
            Convert uppercase of letter to index
            Increment letterCount[index] by 1
    Read next line from dataFile
Print characters and frequencies to outFile
  
```

Contagem da frequência

```

String line; line = dataFile.readLine(); // Le uma linha por vez
int location; char letter;

while (line != null) // Enquanto houver linhas a serem lidas
{
    for (location = 0; location < line.length(); location++)
    {
        letter = line.charAt(location);
        if ((letter >= 'A' && letter <= 'Z') || (letter >= 'a' && letter <= 'z'))
        {
            // 2o truque
            index = (int)Character.toUpperCase(letter) - (int) 'A';
            letterCount[index]++;
        }
    }
    line = dataFile.readLine(); // Lê a próxima linha
}
  
```

... e imprime

```
// print each alphabet letter and its frequency count
for (index = 0; index < letterCount.length; index++)
{
    System.out.println("The total number of "
        + (char) (index + (int) 'A')
        + "'s is ")
        + letterCount[index]);
}
```

Array de objetos

- **Class Point**
- [java.lang.Object](#)
 - [java.awt.geom.Point2D](#)
 - java.awt.Point

Constructor and Description

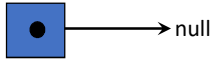
[Point](#)() Constructs and initializes a point at the origin (0, 0) of the coordinate space.

[Point](#)(int x, int y) Constructs and initializes a point at the specified (x,y) location in the coordinate space.

[Point](#)([Point](#) p) Constructs and initializes a point with the same location as the specified Point object.

Array de objetos

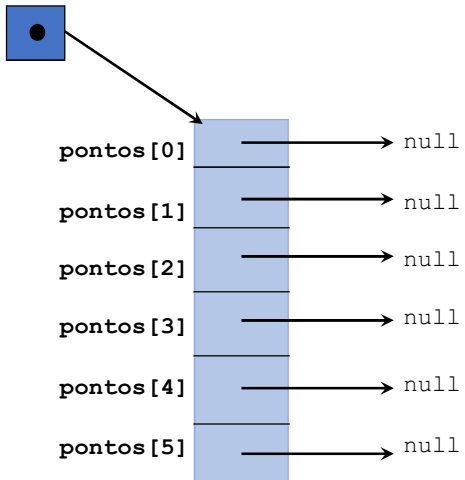
pontos



```
public class ArrayObjetos {  
    public static void main(String[] args){  
        //criando o array  
        Point[] pontos;
```

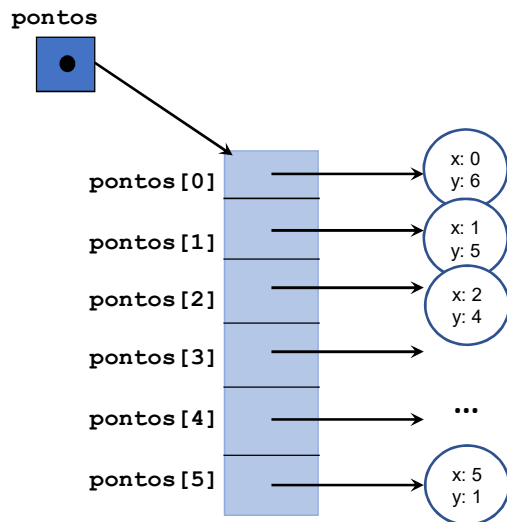
Array de objetos

pontos



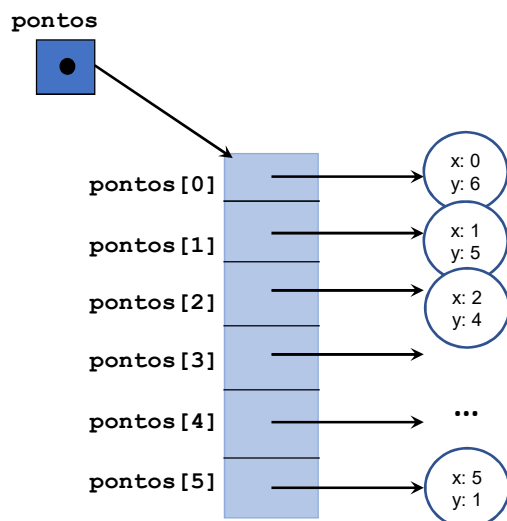
```
public class ArrayObjetos {  
    public static void main(String[] args){  
        //criando o array  
        Point[] pontos;  
        pontos = new Point [6];
```

Array de objetos



```
public class ArrayObjetos {
    public static void main(String[] args){
        //criando o array
        Point[] pontos;
        pontos = new Point [6];
        // instanciando os pontos
        for (int x = 0, y = 6;
            x < pontos.length;
            x++, y--){
            pontos[x] = new Ponto (x,y);
        }
    }
}
```

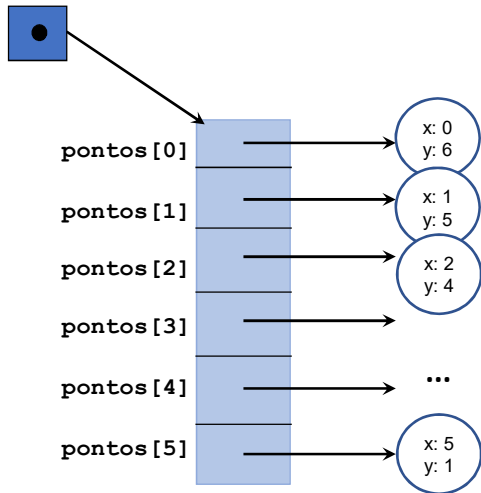
Array de objetos



```
public class ArrayObjetos {
    public static void main(String[] args){
        //criando o array
        Point[] pontos;
        pontos = new Point [6];
        // instanciando os pontos
        for (int x = 0, y = 6;
            x < pontos.length;
            x++, y--){
            pontos[x] = new Ponto (x,y);
        }
        // imprimindo (opção 1)
        for (Ponto p: Point) System.out.print(p);
    }
}
```

Array de objetos

pontos

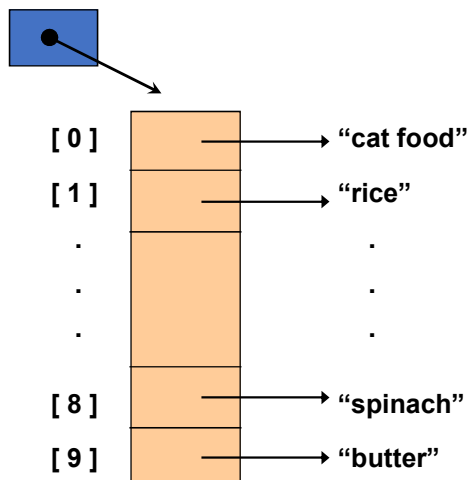


```
public class ArrayObjetos {
    public static void main(String[] args){
        //criando o array
        Point[] pontos;
        pontos = new Point [6];
        // instanciando os pontos
        for (int x = 0, y = 6;
            x < pontos.length;
            x++, y--){
            pontos[x] = new Ponto (x,y);
        }
        // imprimindo (opção 2)
        for (int i = 0; i < pontos.length; i++)
            System.out.print(pontos[i]);
    }
}
```

Array de String

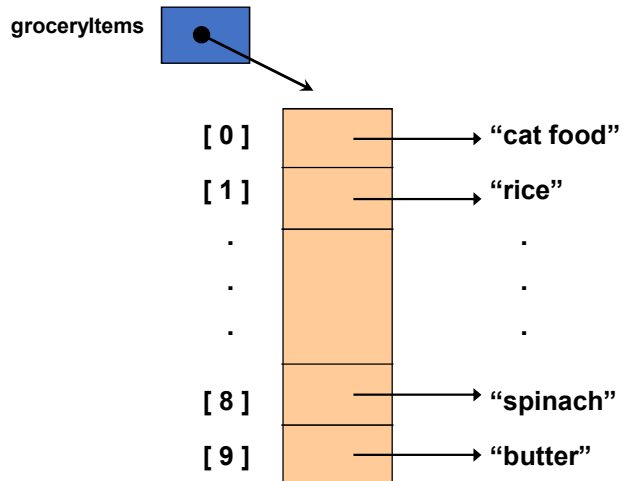
```
String[] groceryItems = new String[10];
```

groceryItems



Array de String

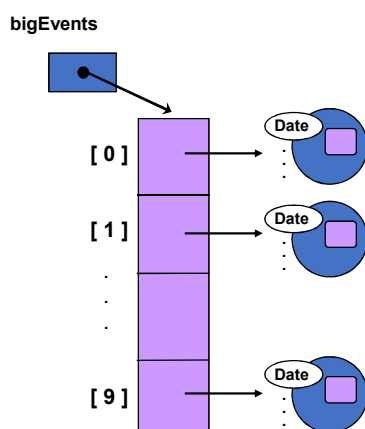
```
String[] groceryItems = new String[10];
```



Expression	Class/Type
<code>groceryItems</code>	Array
<code>groceryItems[0]</code>	String
<code>groceryItems[0].charAt(0)</code>	char

Array de Data

```
Date[] bigEvents = new Date[10];
```



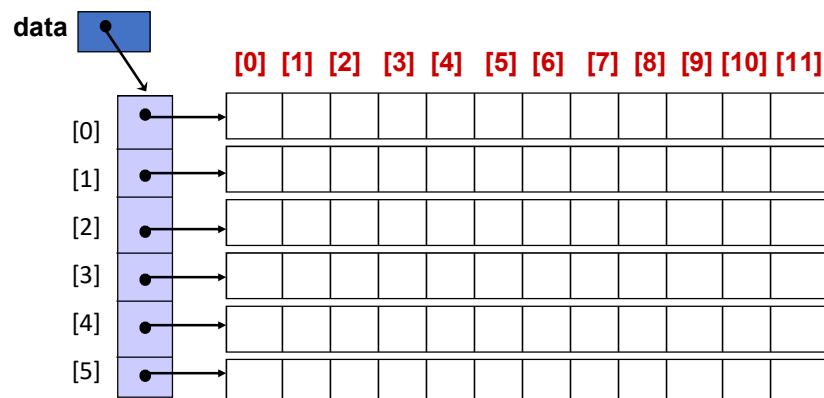
Expression	Class/Type
<code>bigEvents</code>	Array
<code>bigEvents[0]</code>	Date
<code>bigEvents[0].month</code>	String
<code>bigEvents[0].day</code>	int
<code>bigEvents[0].year</code>	int
<code>bigEvents[0].month.charAt(0)</code>	char

Array de duas ou mais dimensões

- ❑ Array de duas ou mais dimensões, ou matrizes, são “array de arrays”
- ❑ Cada dimensão é acessada por um índice
- ❑ Declaração
 - ❑ 2 dimensões
 - ❑ `int[<primeira dim>][<segunda dim>]`
 - ❑ 3 dimensões
 - ❑ `int[<primeira dim>][<segunda dim>][<terceira dim>]`

Array de duas dimensões

- ❑ `int[][] data = new int[6][12];`
- ❑ 1a dimensão é um array de referências para a 2a dimensão

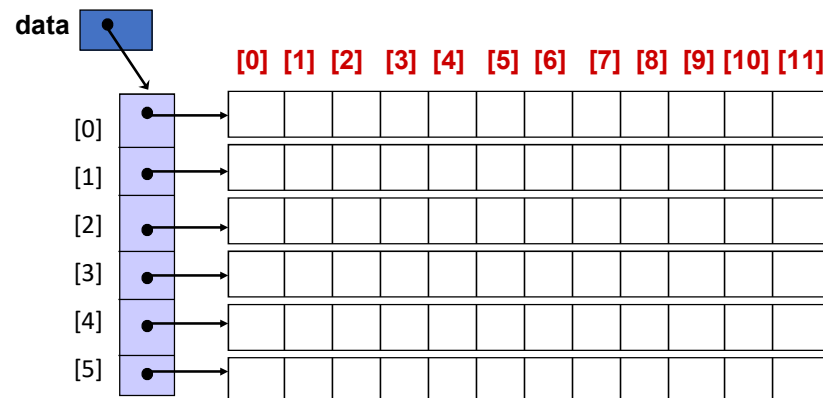


Array de duas dimensões

- Cada array em cada dimensão tem o seu campo **length**

- `data.length`
- `data[1].length`

... vantagem? Se, em princípio, são todos iguais?



Forma “tradicional” para declarar array de duas dimensões

- `new`

```
public class Matriz1 {
    public static void main(String[] args) {
        int[][] data = new int[6][12];
        data[0][0] = 4;
        data[0][1] = 3;
        data[1][0] = 2;
        data[1][1] = 1;
        for(int i=0; i < data.length; i++){ // iterando pela 1ª dimensão
            for(int j=0; j < data[i].length; j++){ // iterando pela 2ª dimensão
                System.out.print("data[" + i + "][" + j + "]: " + data[i][j] + " ");
            }
            System.out.print("\n");
        }
    }
}
```

Declarado array de duas dimensões com lista

```
public class Matriz2{  
    public static void main(String[] args) {  
        int data[][] = { { 1, 2, 3, 4},  
                           { 5, 6, 7, 8},  
                           {32, 2, 7, 3},  
                           { 4, 10, 27, 28}  
        };  
  
        // mesmo código  
    }  
}
```

... e se for assim?

```
public class Matriz3{  
    public static void main(String[] args) {  
        int data[][] = { { 1, 2, 3, 4, 5, 6},  
                           { 5, 6, 7, 8, 25, 13},  
                           {32, 2, 7},  
                           { 4, 10, 27, 28},  
                           {72}  
    };  
}
```

... e se for assim? ... Pode?

```
public class Matriz3{
    public static void main(String[] args) {
        int data[][] = { { 1, 2, 3, 4, 5, 6},
                        { 5, 6, 7, 8, 25, 13},
                        {32, 2, 7},
                        { 4, 10, 27, 28},
                        {72}
                      };

        // mesmo código!

    }
```

... e se for assim? ... Pode!

```
public class Matriz3{
    public static void main(String[] args) {
        int data[][] = { { 1, 2, 3, 4, 5, 6},
                        { 5, 6, 7, 8, 25, 13},
                        {32, 2, 7},
                        { 4, 10, 27, 28},
                        {72}
                      };

        // mesmo código!

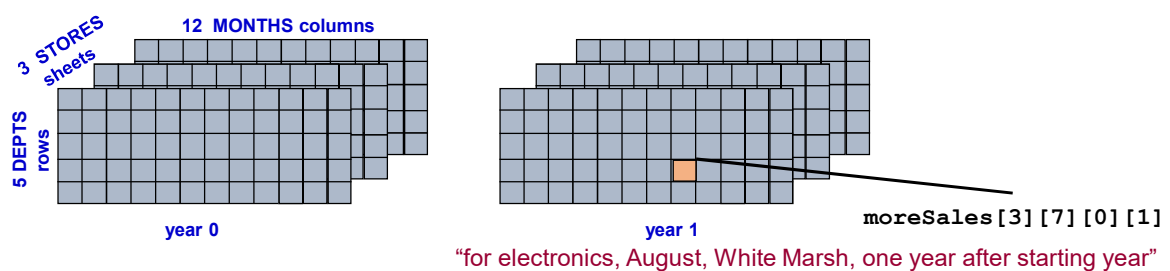
        data.length == 5 // 1ª dimensão
        data[0].length == 6 // arrays da 2ª dimensão ...
        data[1].length == 6
        data[2].length == 3
        data[3].length == 4
        data[4].length == 1

    }
```


Três ou mais dimensões

- ☐ Vale o mesmo raciocínio ...
- ☐ Faça o desenho tri/poli dimensional na sua cabeça ...
- ☐ Exemplo

```
final int NUM_DEPTS = 5;
final int NUM_STORES = 3;
final int NUM_YEARS = 2;
int[][][] moreSales;
moreSales = new int[NUM_DEPTS][12][NUM_STORES][NUM_YEARS];
```

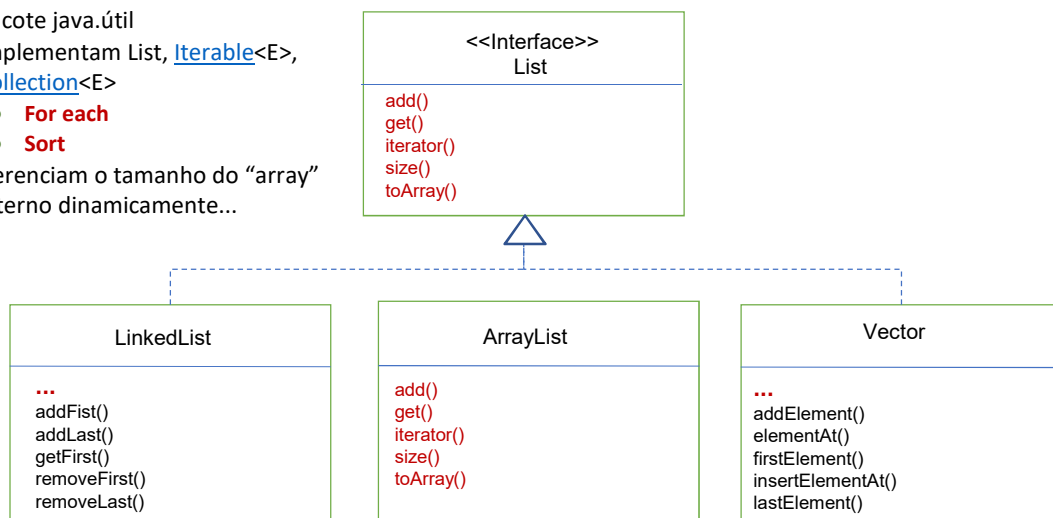


Mais sobre índices

- ☐ Array indexes can be any integral expression of type `char`, `short`, `byte`, or `int`
- ☐ It is the **programmer's responsibility** to make sure that an array index does not go out of bounds. The index must be within the range 0 through the array's length minus 1
- ☐ Using an index value outside this range throws an `ArrayIndexOutOfBoundsException`; prevent this error by using public instance variable `length`
- ☐ Isso não quer dizer que você não precisa TAMBÉM controlar se determinado índice contém um element válido ou não!
 - São coisas diferentes ... índice válido versus elemento existente

Classes para gerenciar coleções e listas

- ❑ Pacote java.util
- ❑ Implementam List, [Iterable](#)<E>, [Collection](#)<E>
 - **For each**
 - **Sort**
- ❑ Gerenciam o tamanho do “array” interno dinamicamente...



ArrayList

- ❑ Para usar arrays ou listas simples de forma mais flexível em Java podemos usar a classe ArrayList
- ❑ Ela é uma classe dinâmica que aumenta seu tamanho após o preenchimento de todo seu espaço, ou seja, se temos uma lista com um número de posições iniciais
 - Pode ser configurado
 - Amplia o tamanho dinamicamente
- ❑ O ideal é inicializar o número de elementos necessários
 - Aumento dinâmico é custoso em CPU e memória
- ❑ Para inserir, recuperar, alterar e remover elementos no ArrayList usamos métodos
 - `add (conteudo)`
 - `remove (indice)`
 - `get (indice)`
 - `set (indice)`
- ❑ Existem mais métodos ... Consulte a documentação

ArrayList

```
import java.util.ArrayList;
public class Lista {
    public static void main(String[] args) {
        ArrayList<Integer> notas = new ArrayList();
        notas.add(1);
        notas.add(9);
        notas.add(10);
        notas.remove(2);
        for(int i =0; i< notas.size();i++){
            System.out.println(notas.get(i));
        }
    }
}
```

Generics*. Usa-se o nome da classe do tipo primitivo

Tamanho da Lista ... Um método, não um campo !!!
Cuidado!

***Não usem agora!!! Vamos ver isso depois ...**

LinkedList

- ❑ O LinkedList possui as mesmas funções que o ArrayList e mais algumas (especialização)
- ❑ A principal diferença está em alguns métodos especializados, relacionados à sua função e propósito de ser uma lista encadeada
 - addFirst()
 - addLast()
 - getFirst()
 - removeFirst()
 - removeLast()

Exemplo LinkedList

- ❑ Tem a mesma interface do ArrayList
- ❑ Mas tem métodos especializados ...
- ❑ No exemplo, vamos também explorar
 - For each
 - E o toString
 - Poupa muito trabalho ...

```
import java.util.LinkedList;
public class Lista {
    public static void main(String[] args){
        LinkedList notas = new LinkedList();
        //Appends the specified element to the end of this list
        notas.add(new Integer(1)); notas.add(new Integer(9));
        notas.add(new Integer(10));
        //Removes the element at the specified position in this list
        notas.remove(1);
        //Inserts the specified element at the beginning of this list
        notas.addFirst (new Integer(0));

        Integer myInt = (Integer) notas.get(3);

        for(Integer i: notas) {System.out.println(i);}

        System.out.println(notas); // teste para ver se dá certo ...
    }
}
```

Exemplo LinkedList

```
import java.util.LinkedList;
public class Lista {
    public static void main(String[] args){
        LinkedList notas = new LinkedList();
        //Appends the specified element to the end of this list
        notas.add(new Integer(1));
        notas.add(new Integer(10));
        //Removes the element at the specified position in this list
        notas.remove(1);
        //Inserts the specified element at the beginning of this list
        notas.addFirst (new Integer(0));

        Integer myInt = (Integer) notas.get(3);

        for(Integer i: notas) {System.out.println(i);}

        System.out.println(notas); // teste para ver se dá certo ...
    }
}
```

C:\Users\alexszt\Documents>javac Lista.java

Lista.java:15: error: incompatible types: Object cannot be converted to Integer

```
for(Integer i: notas) {System.out.println(i);}
```

^

Note: Lista.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

1 error

Exemplo LinkedList

```
import java.util.LinkedList;
public class Lista {
    public static void main(String[] args){
        LinkedList notas = new LinkedList();
        //Appends the specified element to the end of t
        notas.add(new Integer(1)); notas.add(new Integer(9));
        notas.add(new Integer(10));
        //Removes the element at the specified position
        notas.remove(1);
        //Inserts the specified element at the beginning of this list
        notas.addFirst (new Integer(0));

        Integer myInt = (Integer) nota.get(3);

        for(Object i: notas) {System.out.println((Integer)i);}

        System.out.println(notas); // teste para ver se dá certo ...
    }
}
```

```
C:\Users\alexszt\Documents>java Lista
Exception in thread "main" java.lang.IndexOutOfBoundsException:
Index: 3, Size: 3
    at java.util.LinkedList.checkElementIndex(Unknown Source)
    at java.util.LinkedList.get(Unknown Source)
    at Lista.main(Lista.java:13)
```

Exemplo LinkedList

```
import java.util.LinkedList;
public class Lista {
    public static void main(String[] args){
        LinkedList notas = new LinkedList();
        //Appends the specified element to the end of this list
        notas.add(new Integer(1)); notas.add(new Integer(9));
        notas.add(new Integer(10)); notas.add(new Integer(16));
        //Removes the element at the specified position in this list
        notas.remove(new Integer(1));
        //Inserts the specified element at the beginning of this list
        notas.addFirst (new Integer(0));

        Integer myInt = (Integer) nota.get(1);

        for(Object i: notas) {System.out.println((Integer)i);}

        System.out.println(notas); // teste para ver se dá certo ...
    }
}
```

```
C:\Users\alexszt\Documents>java Lista
0
9
10
16
[0, 9, 10, 16]
```

Exemplo LinkedList

C:\Users\alexszt\Documents>javac -Xlint:unchecked Lista.java
 Lista.java:6: warning: [unchecked] unchecked call to add(E) as a member of the raw type LinkedList

```
    notas.add(new Integer(1)); notas.add(new Integer(9));
           ^
```

where E is a type-variable:

E extends Object declared in class LinkedList

Lista.java:6: warning: [unchecked] unchecked call to add(E) as a member of the raw type LinkedList

```
    notas.add(new Integer(1)); notas.add(new Integer(9));
           ^
```

where E is a type-variable:

E extends Object declared in class LinkedList

Lista.java:7: warning: [unchecked] unchecked call to add(E) as a member of the raw type LinkedList

```
    notas.add(new Integer(10)); notas.add(new Integer(16));
           ^
```

where E is a type-variable:

E extends Object declared in class LinkedList

Lista.java:7: warning: [unchecked] unchecked call to add(E) as a member of the raw type LinkedList

```
    notas.add(new Integer(10)); notas.add(new Integer(16));
           ^
```

where E is a type-variable:

E extends Object declared in class LinkedList

Lista.java:11: warning: [unchecked] unchecked call to addFirst(E) as a member of the raw type LinkedList

```
    notas.addFirst (new Integer(0));
           ^
```

where E is a type-variable:

E extends Object declared in class LinkedList

5 warnings

Polimorfismo versus Generics

Vector

- ❑ A classe Vector é obsoleta (*depricated*)
 - ❑ Na prática foi substituída pela classe ArrayList
 - ❑ Também pode aumentar o tamanho do array interno
 - ❑ ... mas seu tamanho dobra cada vez que o vetor se esgota.
 - ❑ Assim um array com tamanho 10.000, ao esgotar sua capacidade dobra de tamanho, 20.000
 - ❑ automaticamente com 10.000 espaços ociosos e 10.000 completos
- ❑ Esta classe é a razão de tratarmos os **vetores** em Java sempre como o termo “**array**”.

Exercícios

- ☐ Substitua os objetos da classe *Integer*, por objetos da classe Gato, na lista encadeada