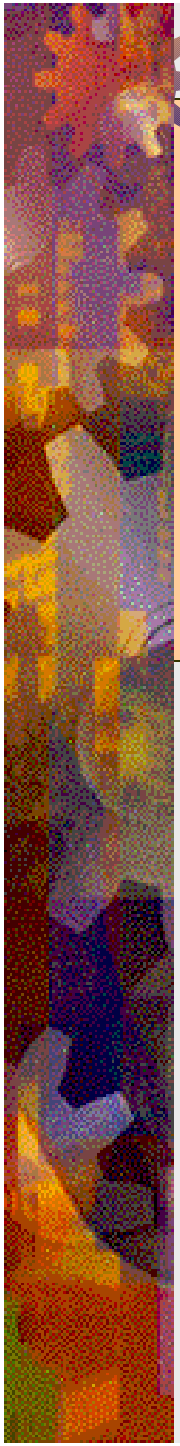


Paradigmas de Programação

Prof. Ricardo Ribeiro dos Santos

ricrs@ec.ucdb.br

Curso de Engenharia de Computação – UCDB – 1º Semestre/2003

Tópicos

- Manipulação de arquivos
  - Operações de entrada
  - Operações de saída
  - Exemplos
- Manipulação de vetores
  - Declaração e definição de instâncias
  - Vetores multidimensionais
  - A classe Vector

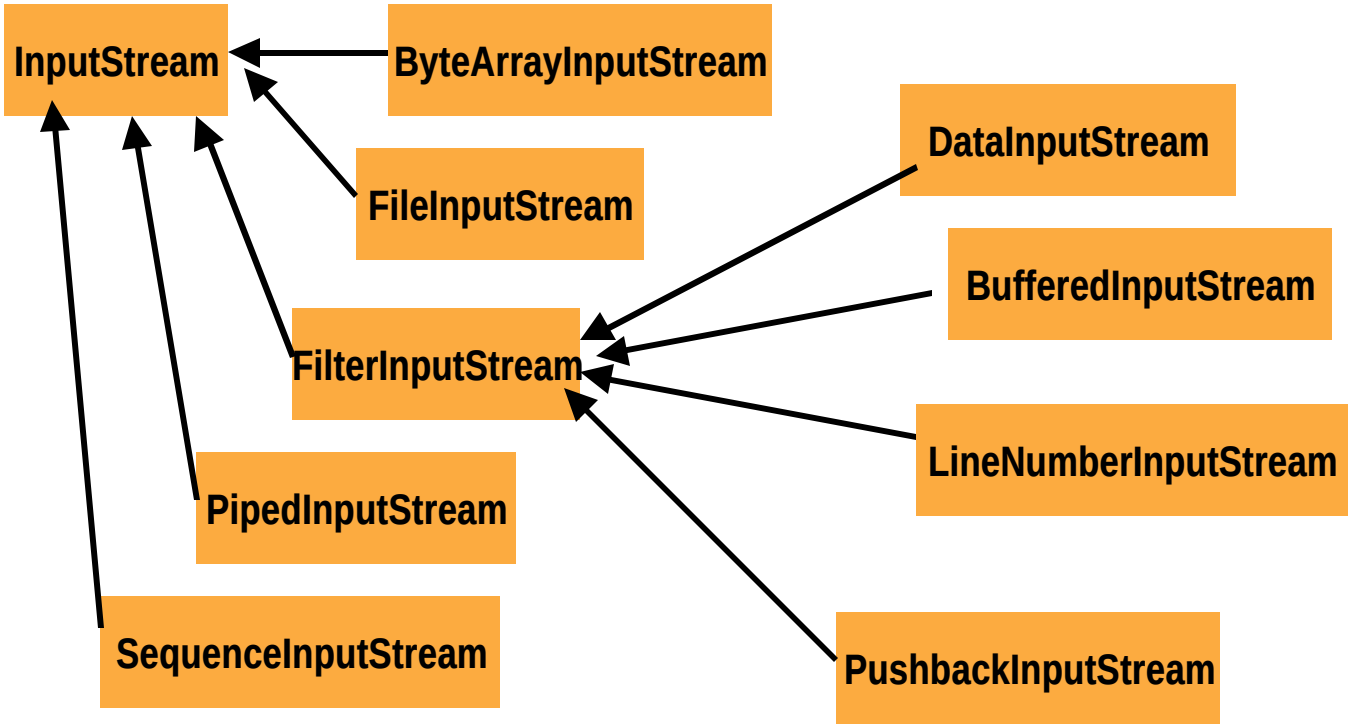
Arquivos em Java

- Para que as aplicações Java possam armazenar e recuperar dados através de arquivos, deve-se lançar mão do pacote **java.io**
- Utilizando objetos de classes do pacote **java.io**, podemos realizar operações de entrada e saída de arquivos!

Operações de entrada

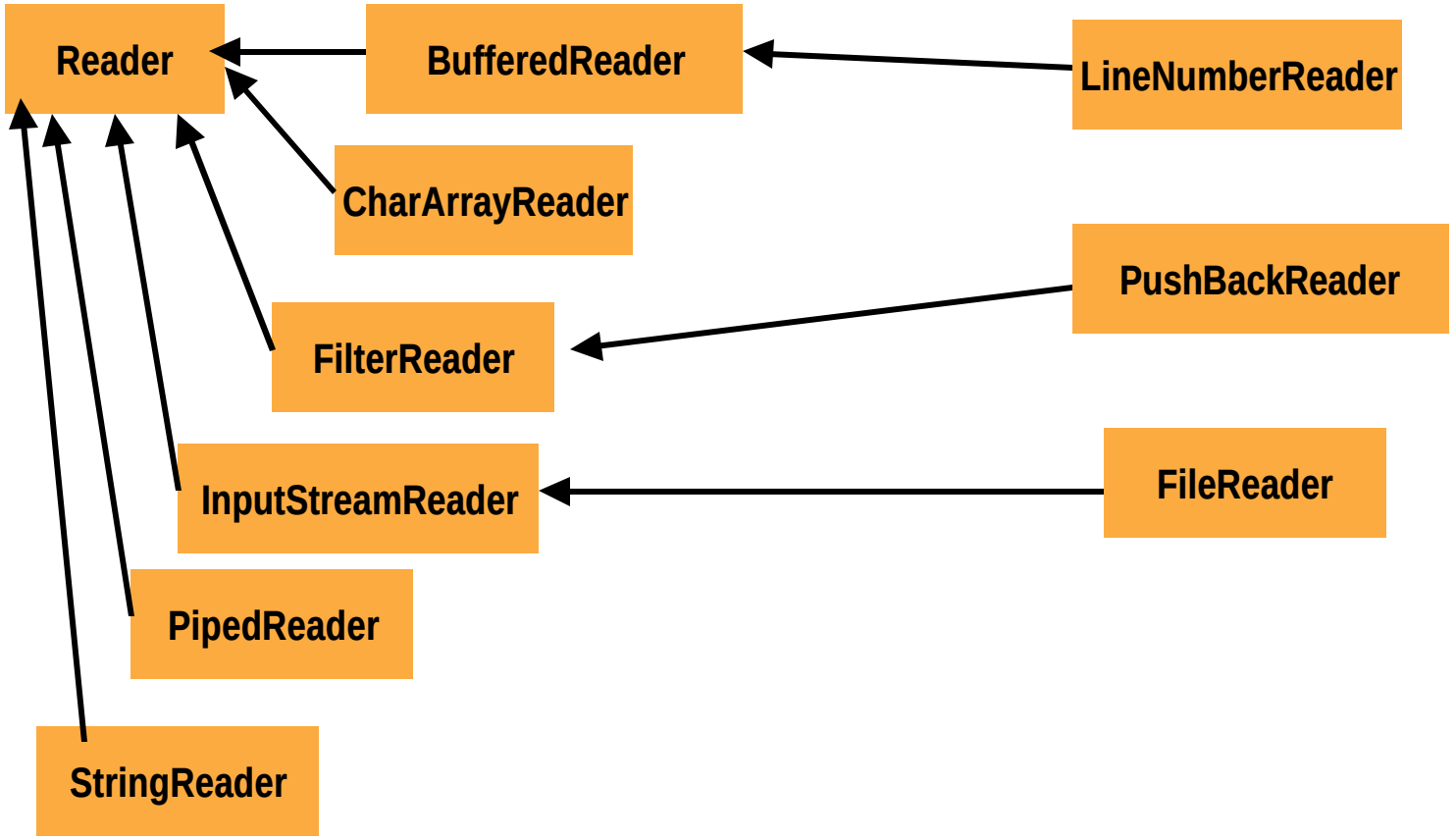
- Todas as operações de entrada são realizadas por classes derivadas das classes java.io.InputStream e java.io.Reader

Hierarquia da classe java.io.InputStream



Operações de entrada

Hierarquia da classe java.io.Reader



Operações de entrada

- Alguns métodos possíveis de serem utilizados para operações de entrada são:

Método	Descrição
available()	Indica a quantidade de bytes disponíveis para leitura sem bloqueio
close()	Fecha a Stream e libera recursos do sistema
read()	Lê o próximo byte disponível
read(byte[])	Preenche o vetor de bytes fornecido como argumento
skip(long)	Descarta a quantidade de bytes especificada
read(byte[],int,int)	Preenche o vetor de bytes fornecido como argumento entre as posições especificadas

Exemplo de manipulação de arquivos em Java

- Exibindo o conteúdo de um arquivo :
  - Utilização de classes derivadas de Reader

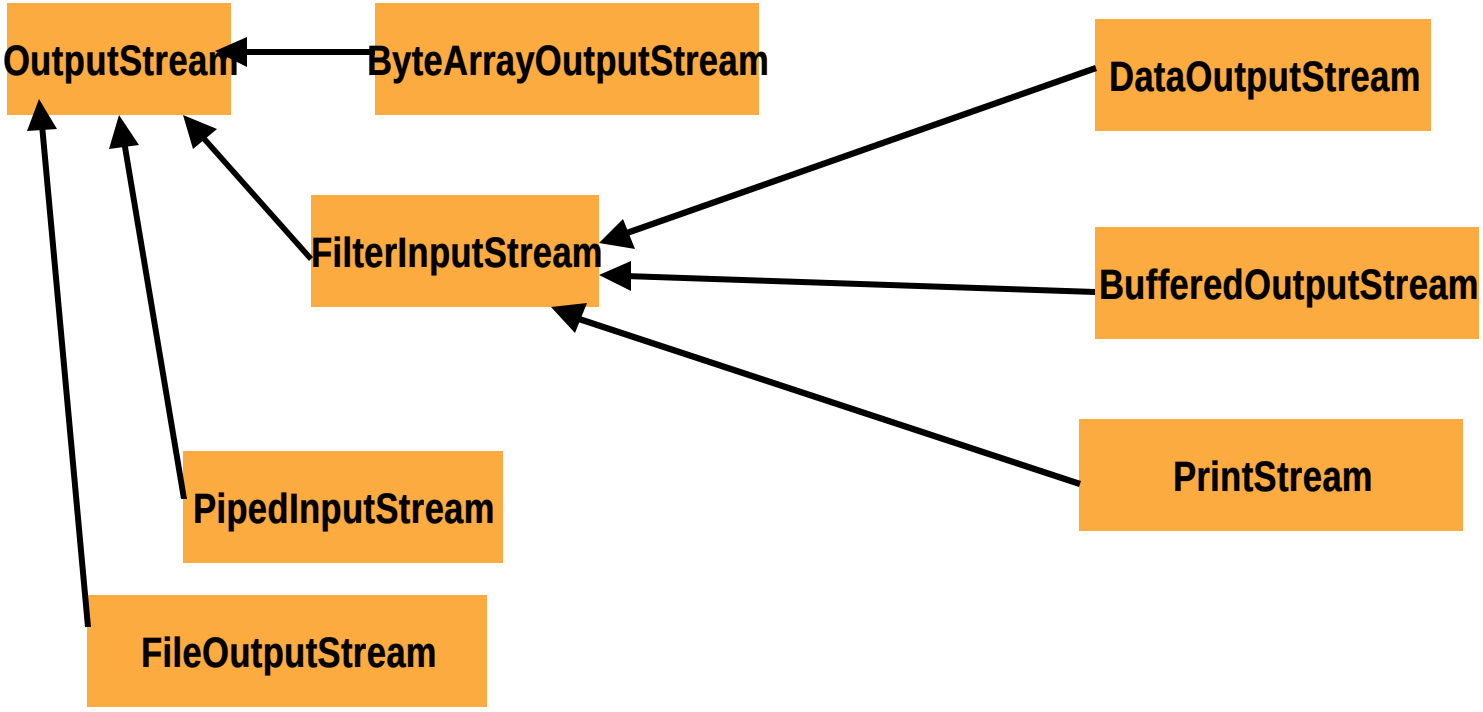
```
import java.io.*;

public class Arquivo3{
     public static void main(String args[])
     {
         try{
           BufferedReader in = new BufferedReader(new FileReader("teste.txt"));
           String teste;
                    while ((teste=in.readLine())!=null)
                         System.out.println("Valor lido do arquivo"+teste);

                     in.close();


        } catch (IOException exc){ System.out.println("Erro de IO");  }
     }
}
```

Nesse programa, o que acontece se, por exemplo, o arquivo “teste.txt” não existe?

Operações de saída

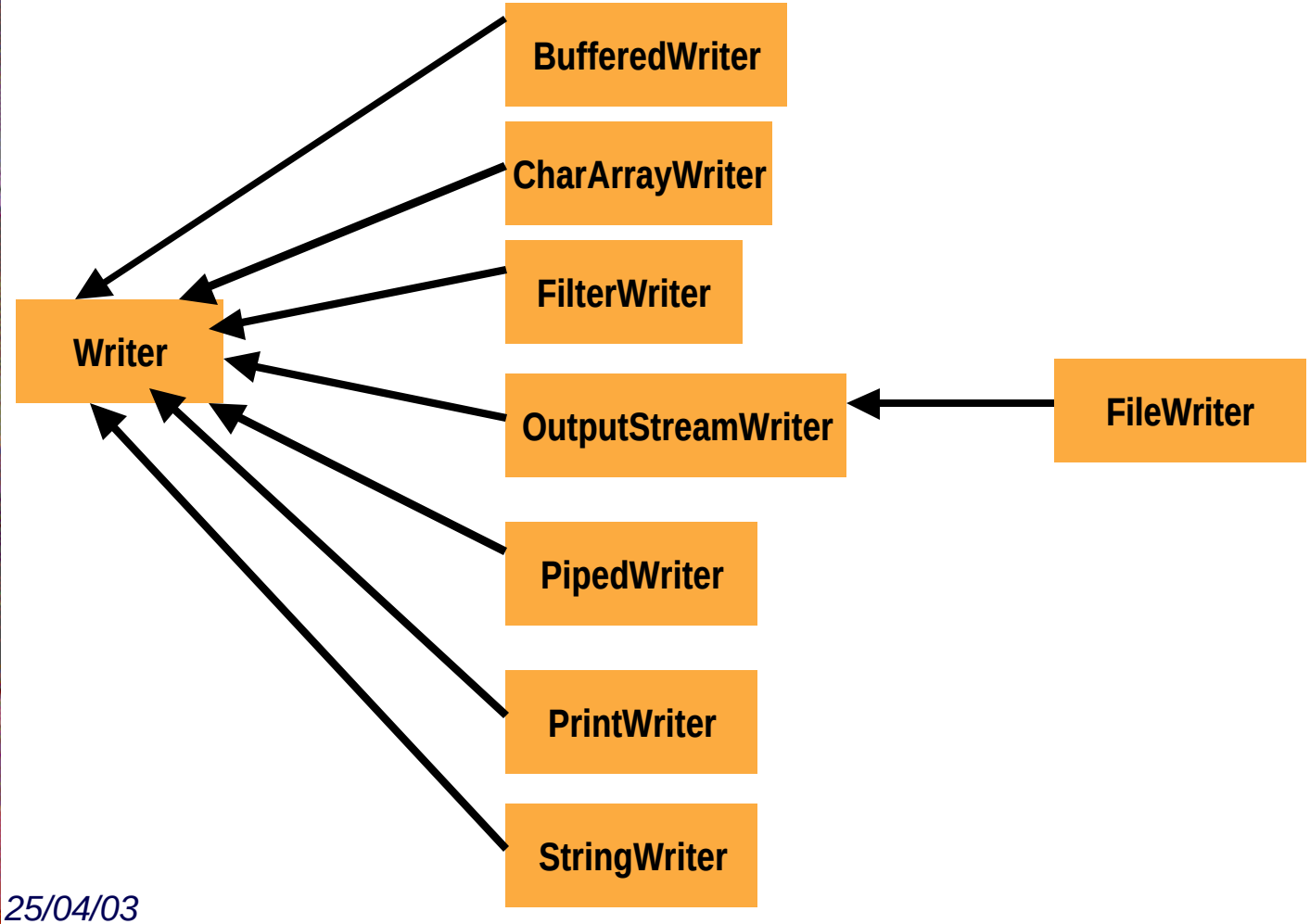
- Todas as operações de entrada são realizadas por classes derivadas das classes java.io.OutputStream e/ou java.io.Writer

Hierarquia da classe java.io.OutputStream



Operações de saída

Hierarquia da classe java.io.Writer



Operações de saída

- Alguns métodos possíveis de serem utilizados para operações de saída são:

Método	Descrição
flush()	Força a escrita em disco de qualquer dado que esteja no *buffer* da *stream*
close()	Fecha a Stream e libera recursos do sistema
write(byte[])	Escreve o vetor de bytes na *stream*
write(byte[],int,int)	Escreve as posições especificadas no vetor de bytes na *stream*

Exemplo de manipulação de arquivos em Java

- Gravando linha a linha em um arquivo :
  - Utilização de classes derivadas de Writer

```
import java.io.*;

public class Arquivo4{
  public static void main(String args[]){
    try{
       PrintWriter out = new PrintWriter(new FileWriter("teste4.txt",true));
       if (args.length>0)
          out.println("Valor especificado na linha de comando"+args[0]);
       else
          out.println("Nenhum valor foi especificado na linha de comando");
       out.close();
    } catch (IOException exc){ System.out.println("Erro de IO"); }
  }
}
```

Nesse programa, o 2º parâmetro do construtor FileWriter indica que o conteúdo do arquivo deve ser adicionado e não sobrescrito!

Exemplo de manipulação de arquivos em Java

- Copiando o conteúdo de um arquivo para outro:
  - Utilização de classes derivadas de InputStream

```
import java.io.*;
public class Arquivo2{
    public static void main(String args[])
    {
     try{
        BufferedInputStream  in=new BufferedInputStream ( new FileInputStream("teste.txt"));
        BufferedOutputStream out=new BufferedOutputStream (new FileOutputStream("teste2.txt"));
        while (in.available()>0) {
           byte texto[]=new byte[in.available()];
           in.read(texto,0,in.available());
           out.write(texto,0,texto.length);
        }
        out.flush();
        in.close();
        out.close();
     } catch (IOException exc){
        System.out.println("Erro de IO");   }
    }
}
```

Exercício

- Oferecer mais uma funcionalidade para o programa que manipula listas:
  - Gravar uma lista em arquivo
  - Recuperar uma lista de arquivo

Vetores

- Em Java vetores são objetos
  - criados na memória dinâmica (*heap*) - referências
  - embora tenham tamanho fixo, pode-se alterar sua referência para um *array* de outro tamanho
- Declaração de vetores

  int numeros[ ]; int[ ] v1,v2;   char mc[ ][ ];
- Instanciação do vetor usando o operador *new*:

  numeros = new int[5];
- Declaração, instanciação e inicialização

  int numeros[ ] = { 10, 20, 30, 40 };

Inicialização Abreviada

- Exemplo: a declaração
  - String [ ] cores = { “verde”, “azul”, “preto” };
- equivale a
  - String [ ] cores = new String [3];
  - cores [0] = “verde”;
  - cores [1] = “azul”;
  - cores [2] = “preto”;

Obtendo o tamanho de vetores

- Se **a** é um identificador de um vetor, **a.length** dá o tamanho de **a**
- Exemplo: o método a seguir imprime um vetor de inteiros de tamanho arbitrário

```
static void imprime (int [] a) {
   for (int i = 0; i < a.length; i = i + 1)
        System.out.println (a[i]);
}
```

Exemplo – Declarando vetores

```
public class Vetores {
   public static void main(String [] args) {
       int [] vetor1, vetor2;
       int vetor3[] = {1,2,3,4,5,6,7,8,9,10};

       vetor1 = vetor3;

       for(int i = 0; i < vetor1.length; i++) {
             System.out.println("Elemento " + i +
                                "igual a " +vetor1[i]);
       }
   }
 }
```

Vetores bidimensionais

- Exemplo
  - int twoDim [ ] [ ] = new int [4] [ ]
  - twoDim[0] = new int [5]
  - twoDim[1] = new int [5]
  - twoDim[2] = new int [5]
  - twoDim[3] = new int [5]

- A sintaxe a seguir **não** é válida
  - int twoDim [ ] [ ] = new int [ ] [4]

Vetores bidimensionais

- Exemplo – Tamanhos variáveis
  - int twoDim [ ] [ ] = new int [5] [ ]
  - twoDim[0] = new int [2]
  - twoDim[1] = new int [4]
  - twoDim[2] = new int [6]
  - twoDim[3] = new int [8]
  - twoDim[3] = new int [10]

- Há também a possibilidade de definir diretamente os tamanhos do vetor:
  - int twoDim [ ] [ ] = new int [4] [5]

Exemplo - vetores multidimensionais

```
public class VetorMulti {
 public static void main(String args[]){
   int myarray[][] = new int[3][5];
   for (int i=0; i < myarray.length; i++){
      for (int k=0; k < myarray[i].length; k++){
         myarray[i][k] = i*10 + k;
         System.out.println("pos["+i+"]["+k+"]=
                                "+myarray[i][k]);
      }
   }
  }
}
```

número de linhas

número de colunas da linha i

Classe Vector

- A classe Vector possibilita que estruturas de dados do tipo vetor possam redimensionar dinamicamente
- Um objeto da classe Vector armazena referências a outros objetos não importando o tipo desse objeto; Com isso, pode-se armazenar qualquer tipo de informação em um objeto Vector
  - Para armazenar valores primitivos, utiliza-se objetos das classes: **Integer, Long, Float, String**

Classe Vector

- Diferente dos vetores tradicionais, todas as operações sobre os elementos (inclusão, remoção, acesso a elementos, etc.) é realizada através de métodos
- Um Vector é automaticamente redimensionado toda vez que o número de elementos ultrapassa a capacidade atual
  - Se não for especificado um incremento de capacidade, o sistema dobrará o tamanho do Vector toda vez que capacidade adicional for necessária

Classe Vector

- Alguns métodos são:

Método	Descrição
addElement(valor)	Adiciona **valor** ao final do Vector. Outro método relacionado é **insertElementAt**
removeElement(valor)	Remove a primeira ocorrência de **valor** do Vector. Retorna **true** se o valor for encontrado. Outros métodos relacionados são: **removeAllElements** e **removeElementAt**
firstElement()	Retorna o valor do primeiro elemento do Vector
lastElement()	Retorna o valor do último elemento do Vector
isEmpty()	Retorna **true** se não houver nenhum elemento no Vector

Classe Vector

- Alguns métodos são (cont.):

Método	Descrição
contains(valor)	Retorna **true** se o Vector contém valor
indexOf(valor)	Retorna o índice da primeira ocorrência de valor em Vector. Retorna –1 se o elemento não foi localizado no Vector
trimToSize()	Reduz a capacidade do Vector para o número atual de elementos
size()	Retorna o número atual de elementos do Vector
capacity()	Retorna o número de elementos que pode ser armazenado no Vector sem a necessidade de alocar mais memória

Classe Vector

Exemplos:

```
import java.util.*;

public class VectorTeste{
  public static void main(String args[])
  {
     Vector v=new Vector(1);
     Integer x=new Integer(10);
     v.addElement(x);
     x=new Integer(20);
     v.addElement(x);
     x=new Integer(30);
     v.addElement(x);
     System.out.println("Primeiro elemento:"+v.firstElement());
     System.out.println("Ultimo elemento:"+v.lastElement());
  }
}
```

Classe Vector

Exemplos:

```
import java.util.*;

public class VectorTeste2{
  public static void main(String args[])
  {
     Vector v=new Vector(1);
     Integer x=new Integer(10);
     String n=new String(“Teste");
     v.addElement(x);
     x=new Integer(20);
     v.addElement(x);
     v.addElement(n);
     System.out.println(“Tamanho do vetor:"+v.capacity());
     System.out.println(“Número atual de elementos:"+v.size());
  }
}
```

Capacidade inicial