



Universidade

Cruzeiro do Sul

Estrutura de Dados I

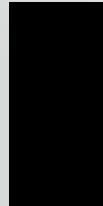
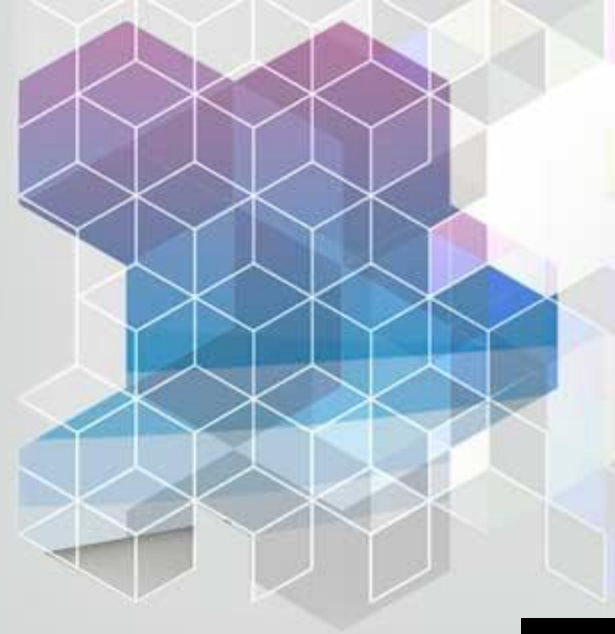
Curso Ciência da Computação

Aula 02

Prof. Claudio Benossi

1. Unidade


**Vetores, Matrizes, TAD e
Alocação Dinâmica de
Memória**





Estrutura de Dados

O que são algoritmos?

- Um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída.
 - Uma ferramenta para resolver um problema computacional bem especificado.
 - O enunciado do problema especifica em termos gerais a entrada e a saída do problema
- 

Estrutura de Dados

Entrada:

$V[] = \{31, 41, 59, 26, 41, 58\}$

Saída esperada:

$V[] = \{26, 31, 41, 41, 58, 59\}$

Instância do problema



Entrada que satisfaz a
quaisquer restrições impostas
no enunciado do problema.

Um algoritmo é dito **CORRETO** se gerar uma saída correta para cada instância do problema.

Estrutura de Dados

Algoritmos eficientes.

Medida de eficiência: velocidade (quanto tempo um algoritmo demora para produzir seu resultado).





Estrutura de Dados


Problemas de instância

Encontrar a média dos elementos de um vetor $A[1 \dots n]$ de números.

Uma das instâncias deste problema consiste em encontrar a média dos elementos do vetor $A[] = \{876, -145, 323, 112, 221\}$.

Outra instância deste problema consiste em encontrar a média dos elementos do vetor $B[] = \{100, 120, 350, 440, 50, 600, 200\}$.

O tamanho de uma instância de um problema é a quantidade de dados necessária para descrever a instância. Neste caso, diz-se que $A = |5|$ e $B = |7|$






Arranjos – Array (Vetores ou Matrizes)

Em computação um Vetor (Array) ou arranjo é o nome de uma matriz unidimensional considerada a mais simples das estruturas de dados.

Vetor (array unidimensional) é uma variável que armazena várias variáveis do mesmo tipo.

No problema apresentado anteriormente, nós podemos utilizar um vetor de 50 posições para armazenar os nomes dos 50 alunos.





Arranjos – Array (Vetores ou Matrizes)

Matriz (array multidimensional) é um vetor de vetores. Imagine uma matriz para armazenar as 4 notas de 50 alunos.

Ou seja, um vetor de 50 posições, e em cada posição do vetor, há outro vetor com 4 posições. Isso é uma matriz.

Cada item do vetor (ou matriz) é acessado por um (ou dois) número(s) chamado(s) de índice(s).

Tipos Primitivos e Tipos Abstratos de Dados

Primitivos

- ☐ int
- ☐ boolean
- ☐ double
- ☐ float
- ☐ char

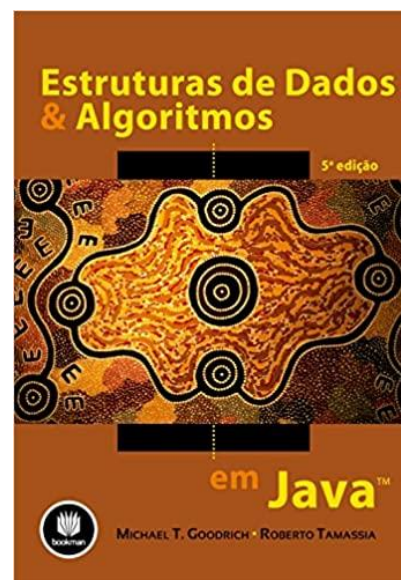
Abstratos

- ☐ Integer
- ☐ Boolean
- ☐ Double
- ☐ Float
- ☐ String
- ☐ Estudante
- ☐ Professor
- ☐ Pessoa
- ☐ Pizza

Arranjos – Array (Vetores ou Matrizes)

“Array (ou vetor) é a estrutura de dados mais simples que existe.

Um vetor armazena uma sequência de valores onde todos são do mesmo tipo”



Loiane Groner

Arranjos – Array (Vetores ou Matrizes)

Imagine a necessidade de armazenar a temperatura média diária de todos os dias de um ano ...

Double tempDia001 = 25.7;

Double tempDia002 = 27.4;

Double tempDia003 = 29.3;

...

Double tempDia365 = 22.8;

**Imagine o tamanho do
código desse programa a
ser realizado,
os recursos que serão
utilizados???**



Arranjos – Array (Vetores ou Matrizes)

Solução:

```
Double[] temperatura = new double[365]
```

```
temperatura[0] = 25.7;
```

```
temperatura[1] = 27.4;
```

```
temperatura[2] = 29.3;
```

...

```
temperatura[364] = 22.8;
```



Arranjos – Array (Vetores ou Matrizes)



Implementação em Java

Importante:

- O uso da estrutura de repetição for
- O Uso do comando length



```
1 package projeto_revisao_ed;
2
3 public class Projeto_revisao_ED {
4
5     public static void main(String[] args) {
6         double[] temperatura = new double[365];
7         temperatura[0] = 25.7;
8         temperatura[1] = 27.4;
9         temperatura[2] = 29.3;
10        temperatura[3] = 31.4;
11        temperatura[4] = 30.2;
12        temperatura[5] = 29.6;
13
14        System.out.println("O valor da temperatura do dia 3 é:" + temperatura[2]);
15        System.out.println("O tamanho do vetor: " + temperatura.length);
16
17        for (int i=0; i<temperatura.length; i++){
18            System.out.println("O valor da temperatura do dia " + (i+1) + " é " + temperatura[i]);
19        }
20
21        for (double temp : temperatura){
22            System.out.println(temp);
23        }
24    }
25 }
```

Arranjos – Array (Vetores ou Matrizes)

Vamos falar sobre a manipulação de dados em um vetor



A **manipulação de dados** nada mais é que o método de coletar, limpar, processar e consolidar os **dados** para uso na análise. Simplificando, ele enriquece os **dados**, os transforma e melhora a precisão do resultado analítico. É uma etapa do processo analítico que consome uma quantidade significativa de tempo e esforço.

Arranjos – Array (Vetores ou Matrizes)



Vamos fazer uma atividade com o conceito de Vetor:

- Criar uma aplicação na linguagem Java para solicitar ao usuário a média diária das temperaturas de um período de 7 dias;
- Verificar e informar:
 - Média da temperatura durante o período;
 - Quantos dias a temperatura ficou acima da média;
 - Quantos dias a temperatura ficou abaixo da média.

INICIO

```
VAR flo_temperatura: ARRAY [7] numerico;  
VAR flo_soma, flo_media, int_contador, int_acima, int_abaixo: numerico;  
flo_soma = 0;
```

```
PARA (int_contador = 1 ATÉ int_contador = 7, int_contador ++ ) FAÇA  
    ESCREVER "Digite a " + int_contador + "ª temperatura";  
    LER flo_temperatura[int_contador];  
    flo_soma = flo_soma + flo_temperatura[int_contador];
```

FIM PARA

```
flo_media = flo_soma / 7;
```

```
int_acima = 0;
```

```
int_abaixo = 0;
```

```
PARA (int_contador = 1 ATÉ int_contador = 7, int_contador ++ ) FAÇA  
    SE (flo_temperatura[int_contador] > flo_media) ENTÃO  
        int_acima = int_acima + 1;
```

FIM SE

```
    SE (flo_temperatura[int_contador] < flo_media) ENTÃO  
        int_abaixo = int_abaixo + 1;
```

FIM SE

FIM PARA

```
ESCREVER "Total de dias que a temperatura ficou acima da média" + int_acima;
```

```
ESCREVER "Total de dias que a temperatura ficou abaixo da média" + int_abaixo;
```

FIM



```
1 package projeto_temperatura;
2
3 import javax.swing.JOptionPane;
4
5 public class Projeto_temperatura {
6     public static void main(String[] args) {
7         double[] temperatura = new double[7];
8         double soma, media;
9         int contador, dias_acima, dias_abaixo;
10        soma = 0;
11        for(contador = 0; contador<7; contador++){
12            temperatura[contador] = Double.parseDouble(JOptionPane.showInputDialog("Digite a " + (contador + 1)
13                + "° temperatura: "));
14            soma = soma + temperatura[contador];
15        }
16        media = soma / 7;
17        dias_acima = 0;
18        dias_abaixo = 0;
19        for(contador = 0; contador<7; contador++){
20            if (temperatura[contador] > media){
21                dias_acima = dias_acima + 1;
22            }
23            if (temperatura[contador] < media){
24                dias_abaixo = dias_abaixo + 1;
25            }
26        }
27        JOptionPane.showMessageDialog(null, "A média das temperaturas é " + media);
28        JOptionPane.showMessageDialog(null, "A quantidade de dias acima da média é " + dias_acima);
29        JOptionPane.showMessageDialog(null, "A quantidade de dias abaixo da média é " + dias_abaixo);
30    }
31 }
```



```
Projeto_temperatura.java x
Código-Fonte Histórico
1 package projeto_temperatura;
2
3 import java.util.Scanner;
4
5 public class Projeto_temperatura {
6     public static void main(String[] args) {
7         double[] temperatura = new double[7];
8         double soma, media;
9         int contador, dias_acima, dias_abaixo;
10        soma = 0;
11        Scanner dados = new Scanner(System.in);
12        for(contador = 0; contador<7; contador++){
13            System.out.println("Digite a " + (contador + 1) + "° temperatura:");
14            temperatura[contador] = dados.nextDouble();
15            soma = soma + temperatura[contador];
16        }
17        media = soma / 7;
18        dias_acima = 0;
19        dias_abaixo = 0;
20        for(contador = 0; contador<7; contador++){
21            if (temperatura[contador] > media){
22                dias_acima = dias_acima + 1;
23            }
24            if (temperatura[contador] < media){
25                dias_abaixo = dias_abaixo + 1;
26            }
27        }
28        System.out.println("A média das temperaturas é " + media);
29        System.out.println("A quantidade de dias acima da média é " + dias_acima);
30        System.out.println("A quantidade de dias abaixo da média é " + dias_abaixo);
31    }
32 }
```



Arranjos – Array (Vetores ou Matrizes)

Para entender os principais tópicos sobre manipulação de vetores vamos criar uma classe Vetor com alguns métodos.





Arranjos – Array (Vetores ou Matrizes)

Um **método em Java** é equivalente a uma função, subrotina ou procedimento em outras linguagens de programação.

Não existe em **Java** o conceito de **métodos** globais.

Todos os **métodos** devem sempre ser definidos dentro de uma classe.





Arranjos – Array (Vetores ou Matrizes)

Vamos criar uma classe em Java que além da estrutura de dados com o nosso **vetor**, vamos criar métodos:

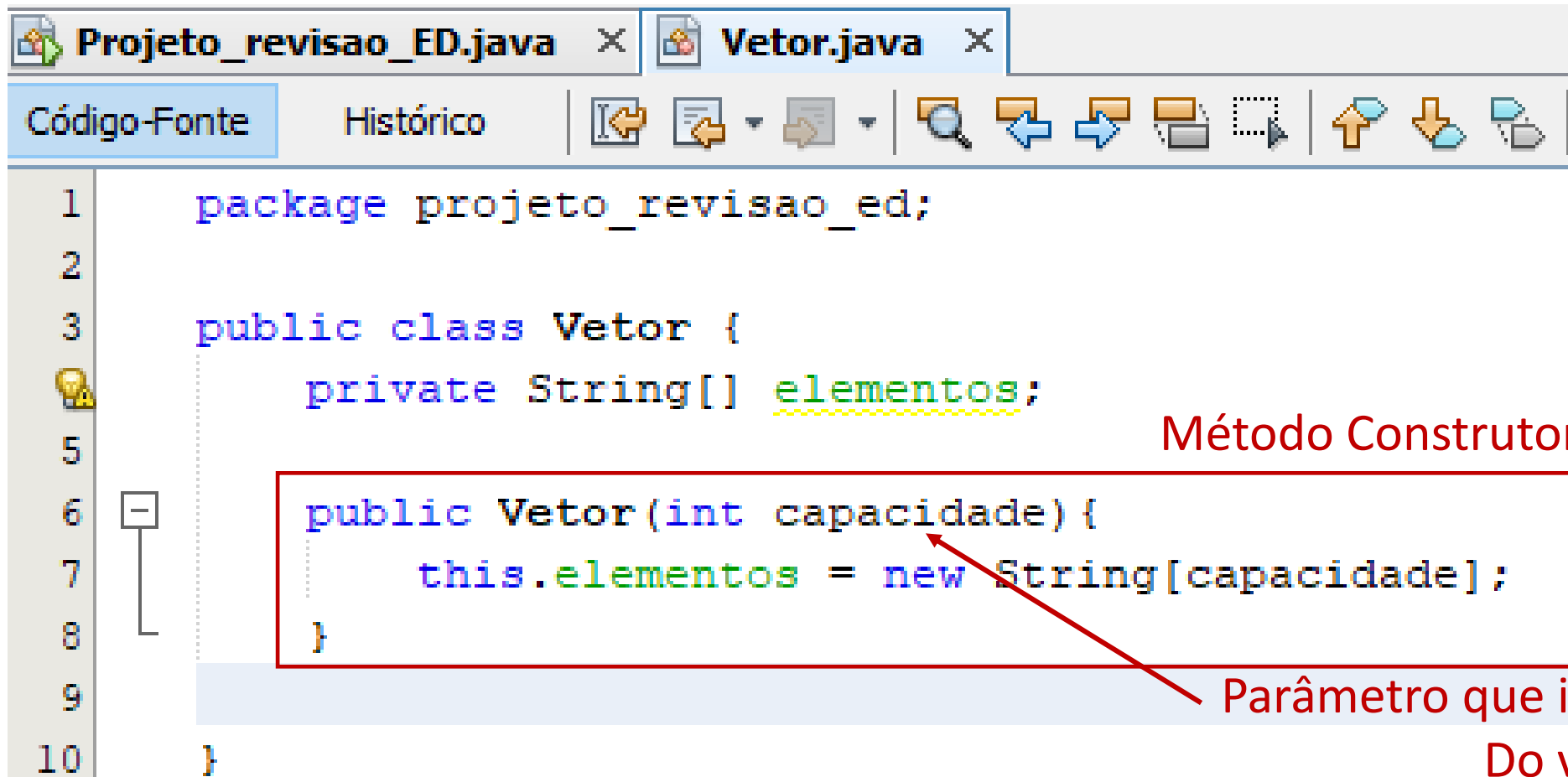
- Método para adicionar um elemento (nesse caso será inserido no final do vetor);
- Também podemos criar um método para adicionar um elemento em uma posição específica;
- Método para remover um elemento;
- Método para buscar um elemento (Pesquisa);
- Método para identificar o tamanho real do nosso vetor;
- E por fim um método para exibir o conteúdo do nosso vetor

```
Projeto_revisao_ED.java x Vetor.java x
Código-Fonte Histórico
1 package projeto_revisao_ed;
2
3 public class Vetor {
4     private String[] elementos;
5     public Vetor(int capacidade){
6         elementos = new String[capacidade];
7     }
8     public void adiciona(String elemento){
9
10    }
11    public void adiciona(int posicao, String elemento){
12
13    }
14    public void remove(int posicao){
15
16    }
17    public String busca(int posicao){
18
19    }
20    public int busca (String elemento){
21
22    }
23    public int tamanho() {
24
25    }
26    public String toString(){
27
28    }
29 }
```

Podemos ter um método com o mesmo, isso é uma **sobreposição**, o que vai diferenciar são os parâmetros passados pelo método

Arranjos – Array (Vetores ou Matrizes)

Primeiro vamos criar a nossa classe Vetor:



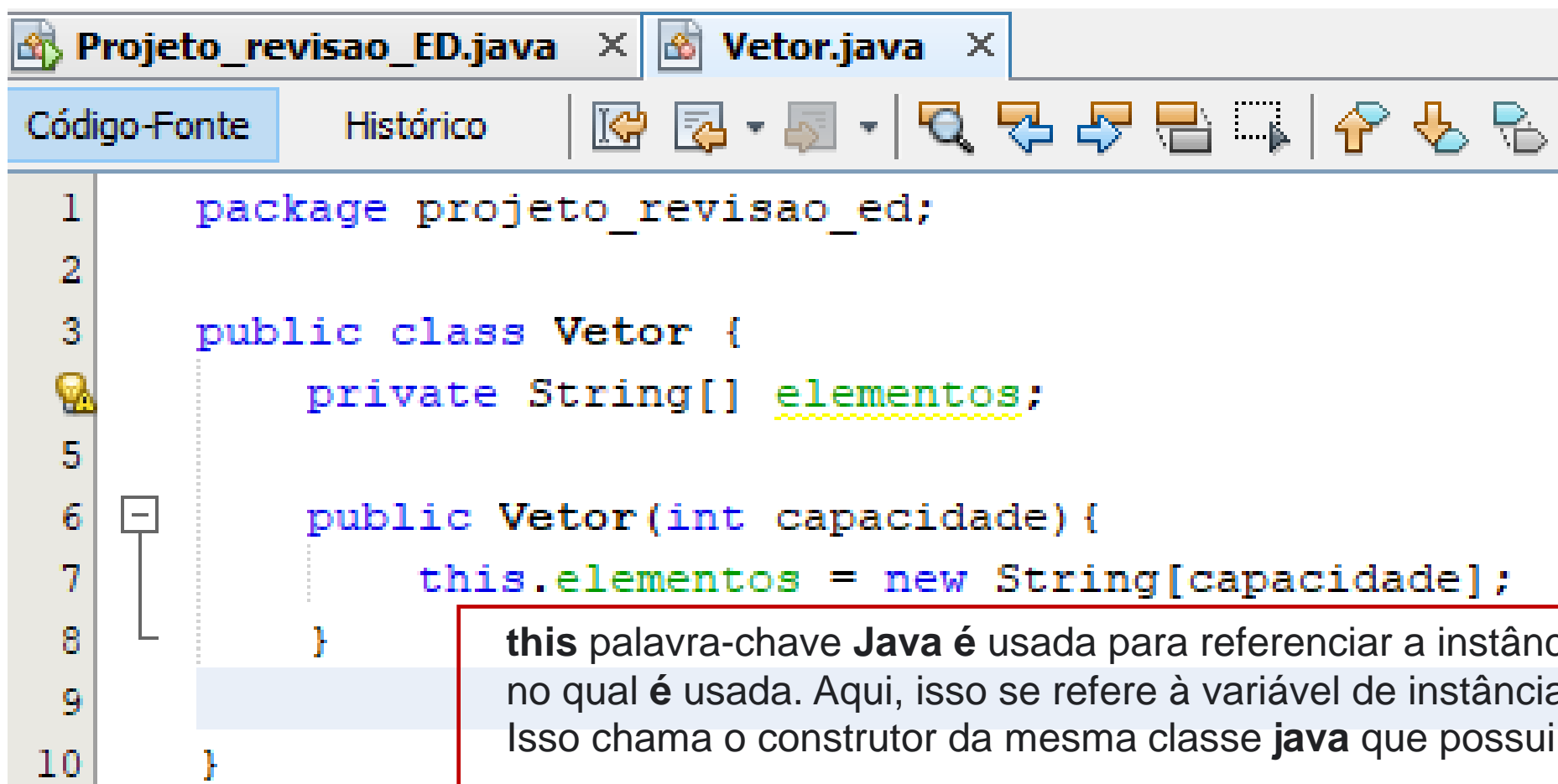
```
1 package projeto_revisao_ed;
2
3 public class Vetor {
4     private String[] elementos;
5
6     public Vetor(int capacidade) {
7         this.elementos = new String[capacidade];
8     }
9
10 }
```

Método Construtor

Parâmetro que indica o tamanho Do vetor

Arranjos – Array (Vetores ou Matrizes)

Primeiro vamos criar a nossa classe Vetor:

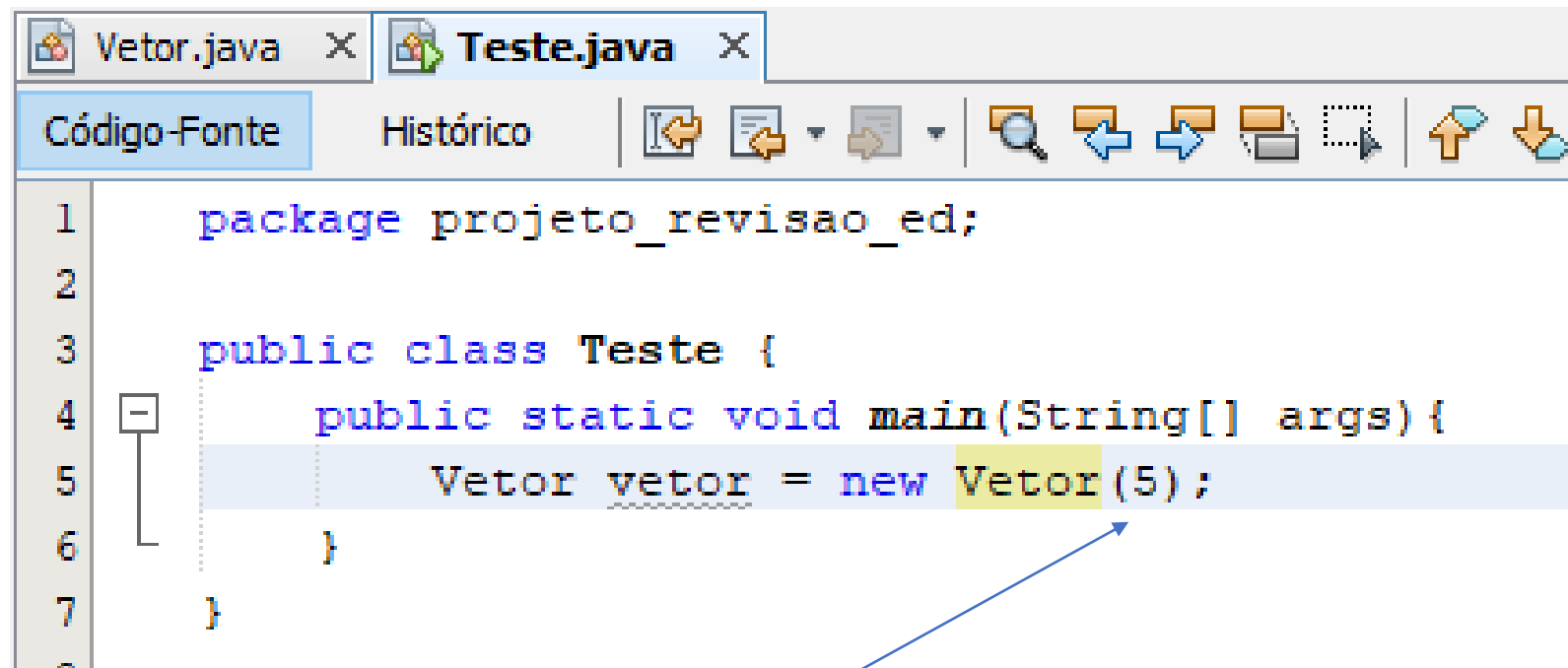


```
1 package projeto_revisao_ed;
2
3 public class Vetor {
4     private String[] elementos;
5
6     public Vetor(int capacidade) {
7         this.elementos = new String[capacidade];
8     }
9
10 }
```

this palavra-chave **Java** é usada para referenciar a instância atual do método no qual é usada. Aqui, isso se refere à variável de instância. ... Isso chama o construtor da mesma classe **java** que possui um parâmetro .

Arranjos – Array (Vetores ou Matrizes)

Vou criar um novo programa para testar a nossa classe através do método void main (ponto de partida para executar a nossa aplicação).



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6     }
7 }
```

Nesse caso vamos criar um vetor com 5 posições, somente.

Arranjos – Array (Vetores ou Matrizes)

Vetor

null	null	null	null	null
Posição 0	Posição 1	Posição 2	Posição 3	Posição 4

Lembrando que quando eu crio um vetor do tipo **String** ele inicia todas as posições como **null** o valor inicial de uma String no Java, se for um vetor do tipo **boolean** terá o valor inicial igual a **False**, se for do tipo **int** o valor inicial será de **0**.



Arranjos – Array (Vetores ou Matrizes)

Agora vamos adicionar elementos no final do nosso vetor, para isso vamos identificar a posição do vetor que está disponível.

Vamos voltar a nossa classe Vetor:

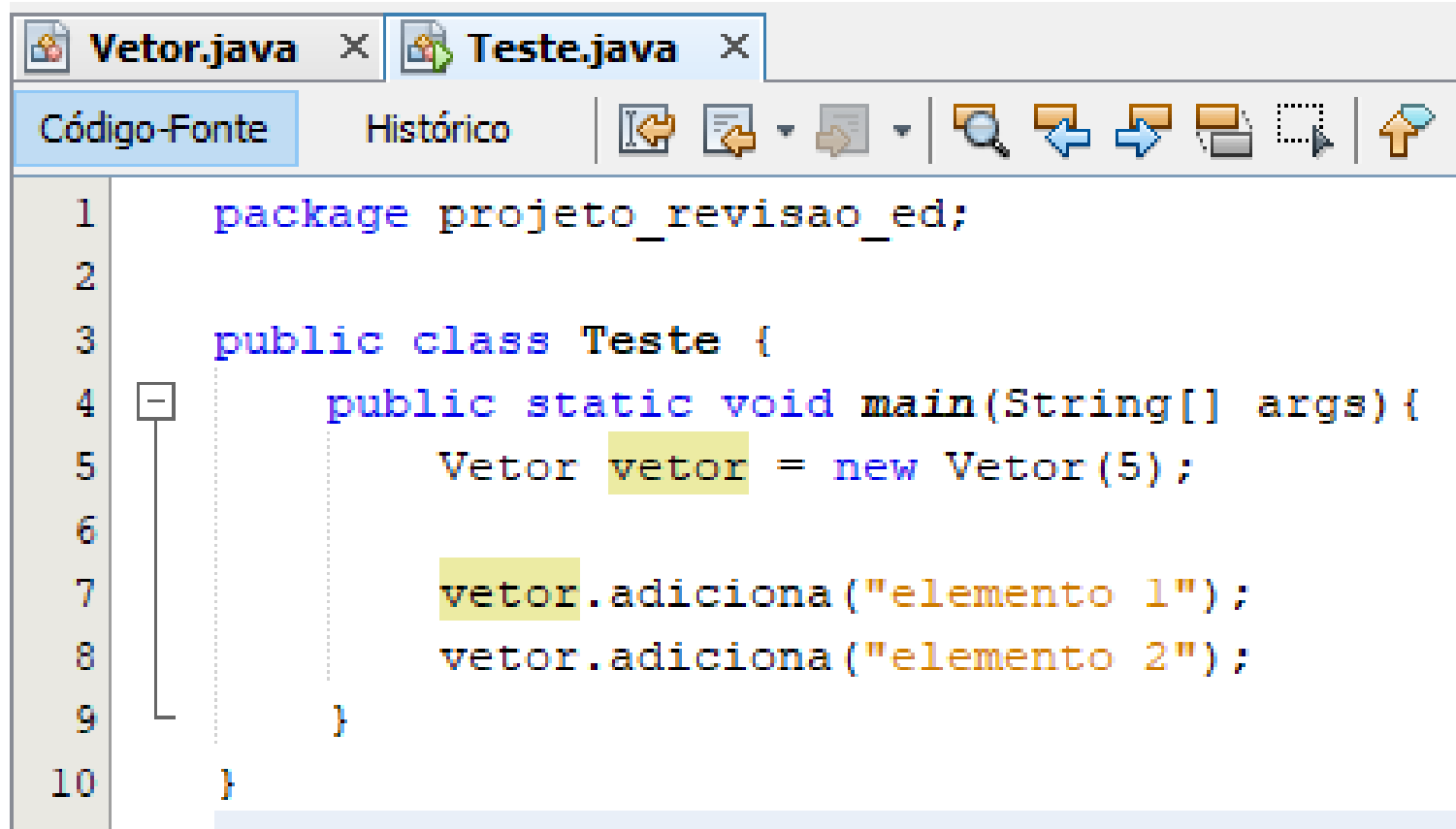
```
1 package projeto_revisao_ed;
2
3 public class Vetor {
4     private String[] elementos;
5
6     public Vetor(int capacidade) {
7         this.elementos = new String[capacidade];
8     }
9
10    public void adiciona(String elemento) {
11        for(int i=0; i<this.elementos.length; i++) {
12            if(this.elementos[i] == null) {
13                this.elementos[i] = elemento;
14                break;
15            }
16        }
17    }
18
19 }
```

Estrutura de Repetição

Estrutura de decisão

Arranjos – Array (Vetores ou Matrizes)

Vamos Testar:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6
7         vetor.adiciona("elemento 1");
8         vetor.adiciona("elemento 2");
9     }
10 }
```

Emento 1	Elemento 2	null	null	null
Posição 0	Posição 1	Posição 2	Posição 3	Posição 4



Arranjos – Array (Vetores ou Matrizes)

Ok, **funciona**, porém quando trabalhamos com um vetor muito grande será necessário fazer uma varredura do vetor isso pode ficar demorado e **não tão eficiente**, nesse caso é recomendado *criar um atributo na minha classe para identificar a ultima posição ocupada*, dessa forma vamos **otimizar o processo**, nesse caso vamos ter o tamanho do vetor e o tamanho real ocupado no nosso vetor.



```
1 package projeto_revisao_ed;
```

```
2
```

```
3 public class Vetor {
```

```
4     private String[] elementos;
```

```
5     private int tamanho;
```

```
6
```

```
7     public Vetor(int capacidade){
```

```
8         this.elementos = new String[capacidade];
```

```
9         this.tamanho = 0;
```

```
10     }
```

```
11
```

```
12     public void adiciona(String elemento) throws Exception{
```

```
13         if (this.tamanho < this.elementos.length){
```

```
14             this.elementos[this.tamanho] = elemento;
```

```
15             this.tamanho++;
```

```
16         } else {
```

```
17             throw new Exception("O Vetor já está cheio, "
```

```
18                 + "não é possível adicionar novos elementos");
```

```
19         }
```

```
20     }
```

```
21
```

Variável para determinar o
tamanho atual do vetor





```
1 package projeto_revisao_ed;
2
3 public class Vetor {
4     private String[] elementos;
5     private int tamanho;
6
7     public Vetor(int capacidade){
8         this.elementos = new String[capacidade];
9         this.tamanho = 0;
10    }
11
12    public void adiciona(String elemento) throws Exception{
13        if (this.tamanho < this.elementos.length){
14            this.elementos[this.tamanho] = elemento;
15            this.tamanho++;
16        } else {
17            throw new Exception("O Vetor já está cheio, "
18                               + "não é possível adicionar novos elementos");
19        }
20    }
21 }
```

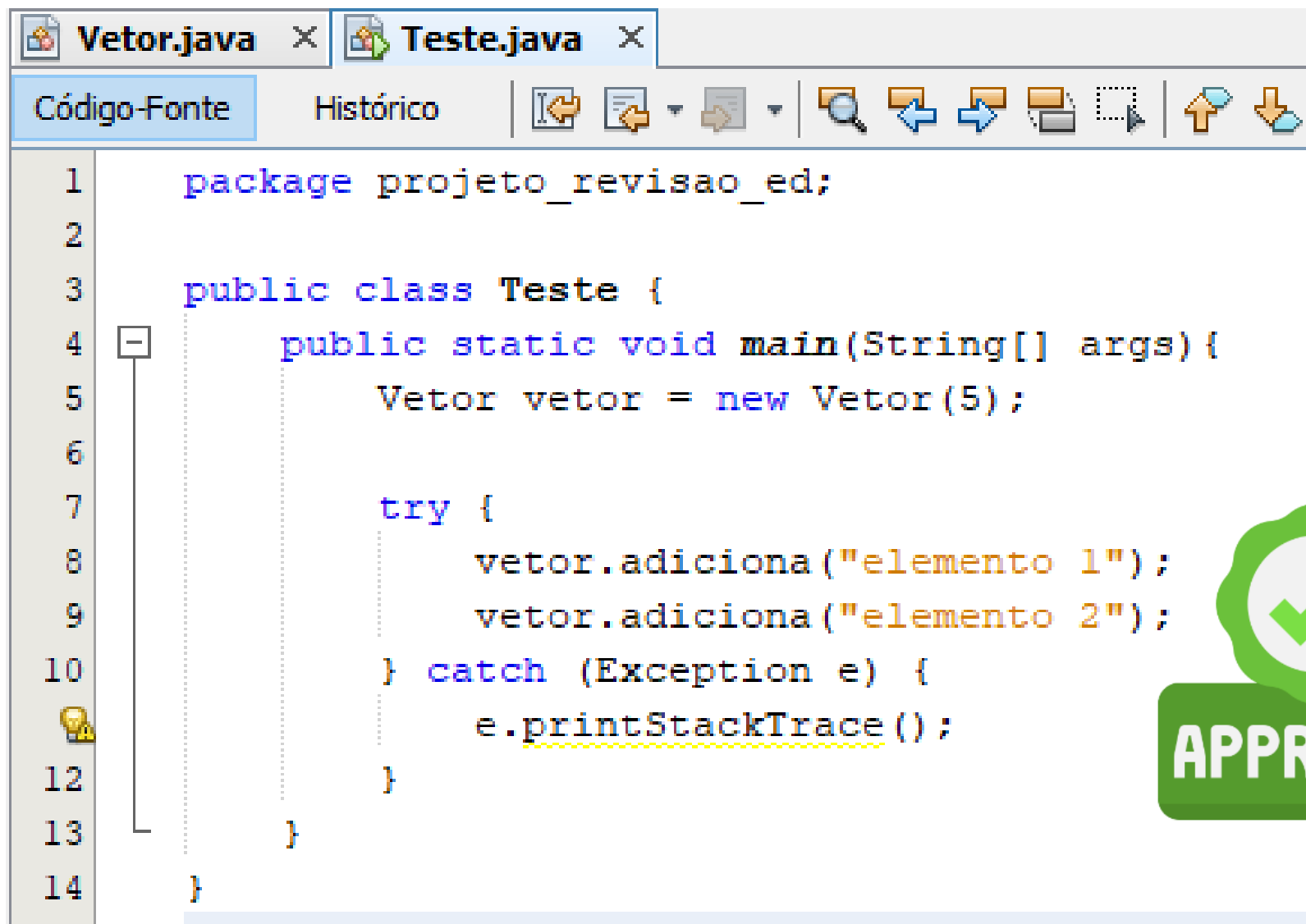
Comandos **throw** e **throws**

Imagine uma situação em que não é desejado que uma exceção seja tratada na própria classe ou método, mas sim em outro que venha lhe chamar.

Para solucionar tal situação utilizamos o comando **throws** na assinatura do método com a possível exceção que o mesmo poderá a vir lançar.

Arranjos – Array (Vetores ou Matrizes)

Vamos Testar:

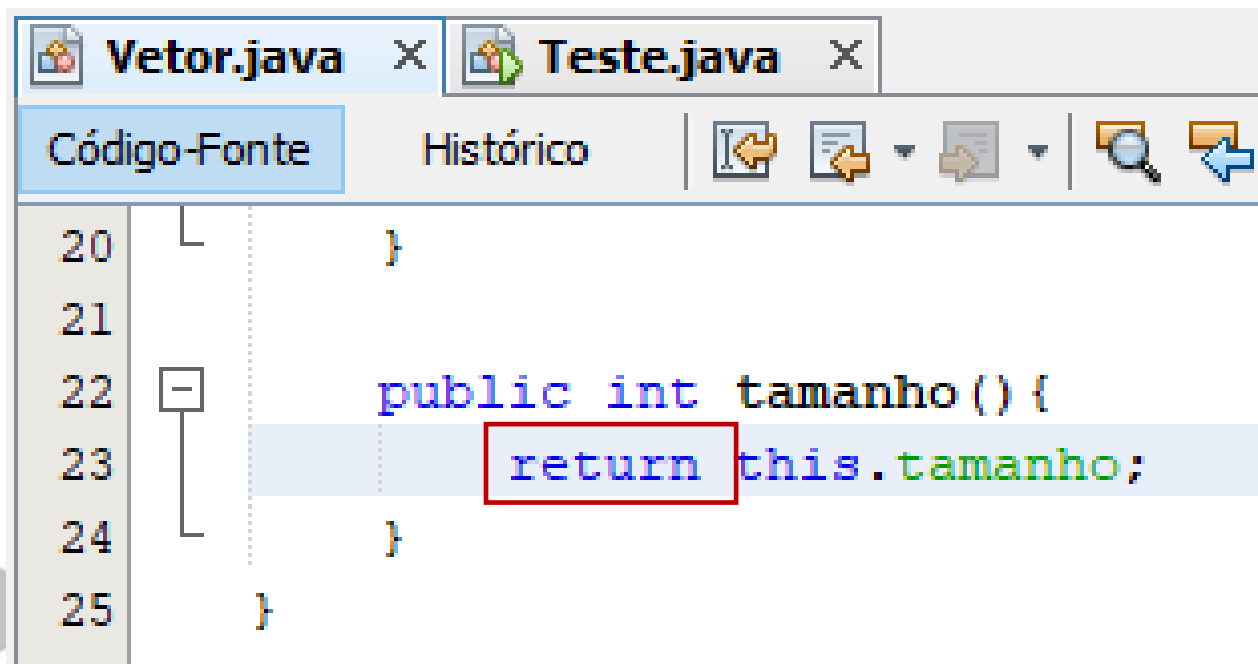


```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13    }
14 }
```



Arranjos – Array (Vetores ou Matrizes)

Agora vamos implementar os métodos para informar o tamanho real de um vetor e como imprimir os elementos do nosso vetor:



```
20 }
21
22 public int tamanho() {
23     return this.tamanho;
24 }
25 }
```

The screenshot shows an IDE with two tabs: 'Vetor.java' and 'Teste.java'. The 'Código-Fonte' (Source Code) tab is active. The code in 'Vetor.java' shows a method 'tamanho()' that returns 'this.tamanho'. The 'return' keyword is highlighted with a red box.

Como o próprio nome diz, o **return** serve para retornar algo de dentro do método!

Sendo assim, todo método que não seja void está informando ao **Java** que ele vai retornar um valor e, por isso, obrigatoriamente deverá utilizar o **return** para devolver um valor!

Arranjos – Array (Vetores ou Matrizes)

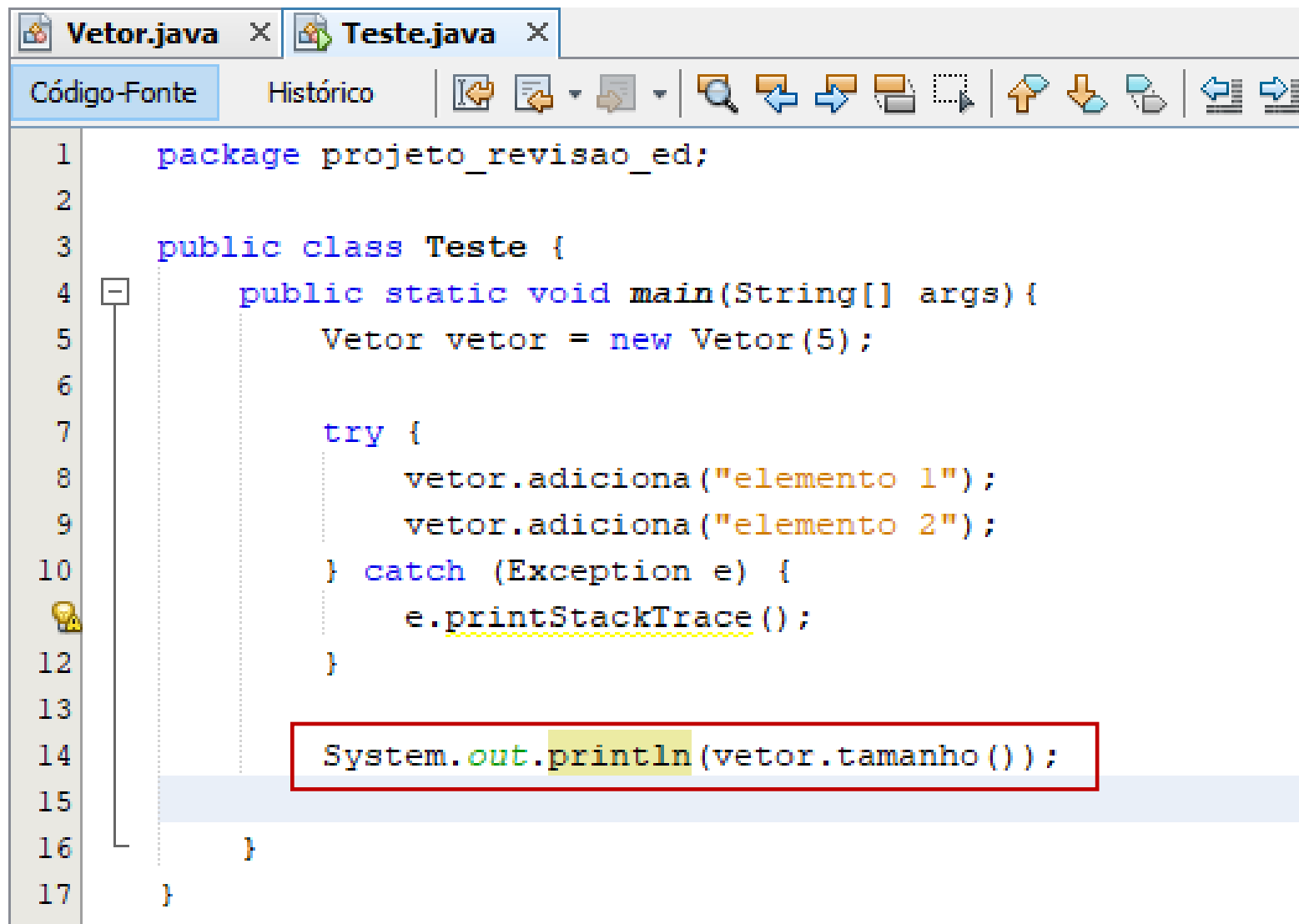
Observe que não criamos métodos de acesso como o **Get** e **Set** para forçar o controle interno da classe e evitar o usuário que vai utilizar a nossa classe não modifique essa informação.

Isso será visto em linguagem de programação, não em estrutura de dados!



Arranjos – Array (Vetores ou Matrizes)

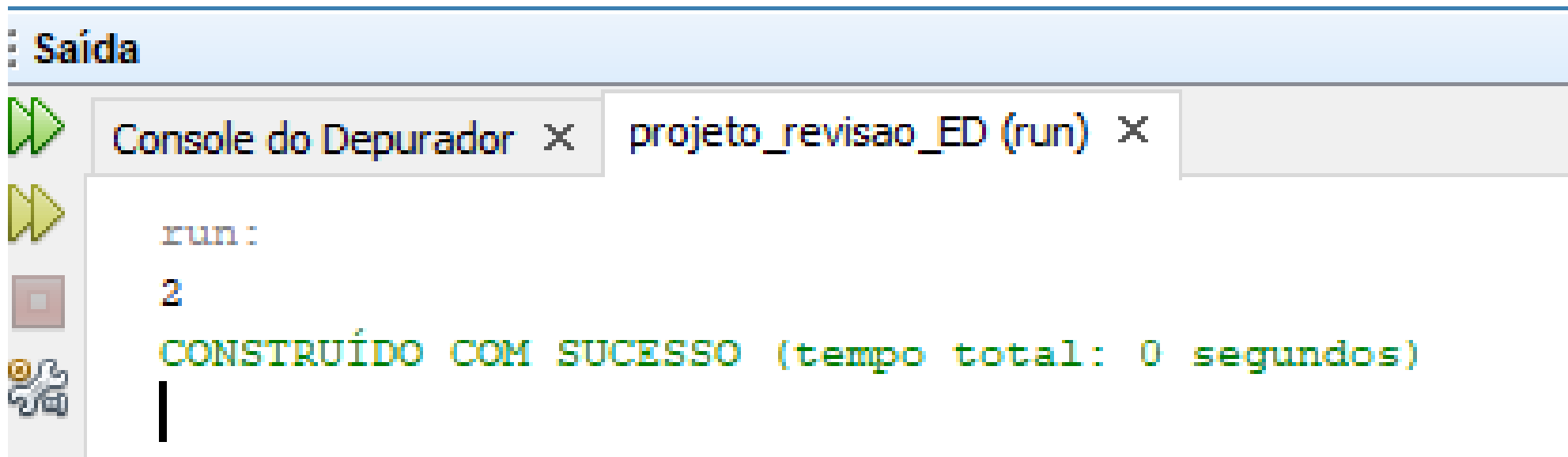
Vamos testar:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16    }
17 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:



The screenshot shows a Python IDE's output console. The title bar of the console window is labeled "Saída". Below the title bar, there are two tabs: "Console do Depurador" and "projeto_revisao_ED (run)". The "projeto_revisao_ED (run)" tab is active, displaying the output of the program. The output consists of three lines: "run:", "2", and "CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)". A cursor is visible at the end of the third line.

```
run:
2
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Ok, mas agora queremos imprimir as informações armazenadas em nosso vetor nas posições ocupadas:

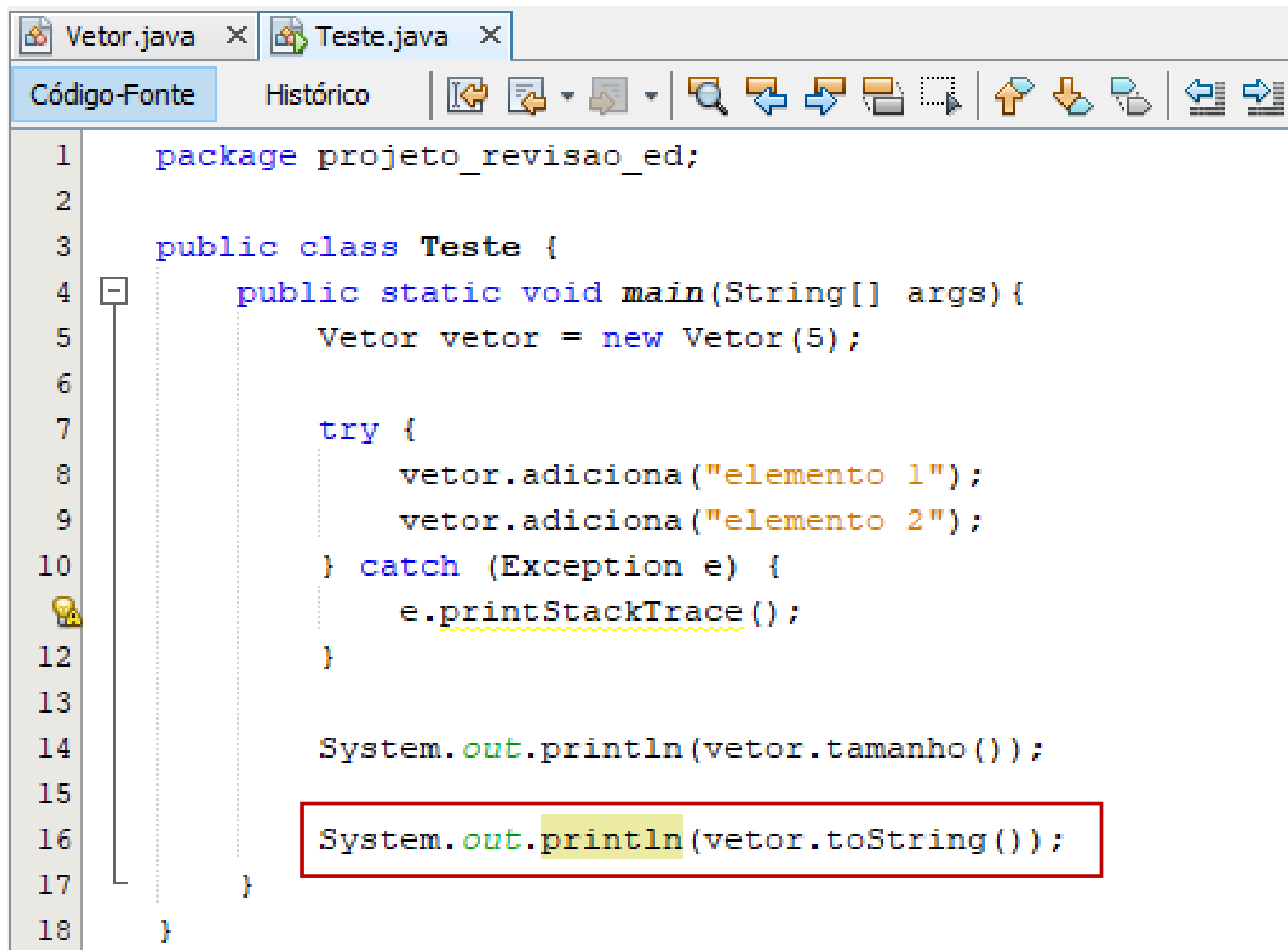
Importando a Classe Arrays que tem uma série de métodos prontos para manipular as informações contidas em um vetor ou matriz.

```
1 package projeto_revisao_ed;
2
3 import java.util.Arrays;
4
5 public class Vetor {
6     private String[] elementos;
7     private int tamanho;
8
9     public Vetor(int capacidade) {...4 linhas }
10
11     public void adiciona(String elemento) throws Exception {...9 linhas }
12
13     public int tamanho() {...3 linhas }
14
15     public String toString() {
16         return Arrays.toString(elementos);
17     }
18 }
```

Importando a Classe Arrays que tem uma série de métodos prontos para manipular as informações contidas em um vetor ou matriz.

Arranjos – Array (Vetores ou Matrizes)

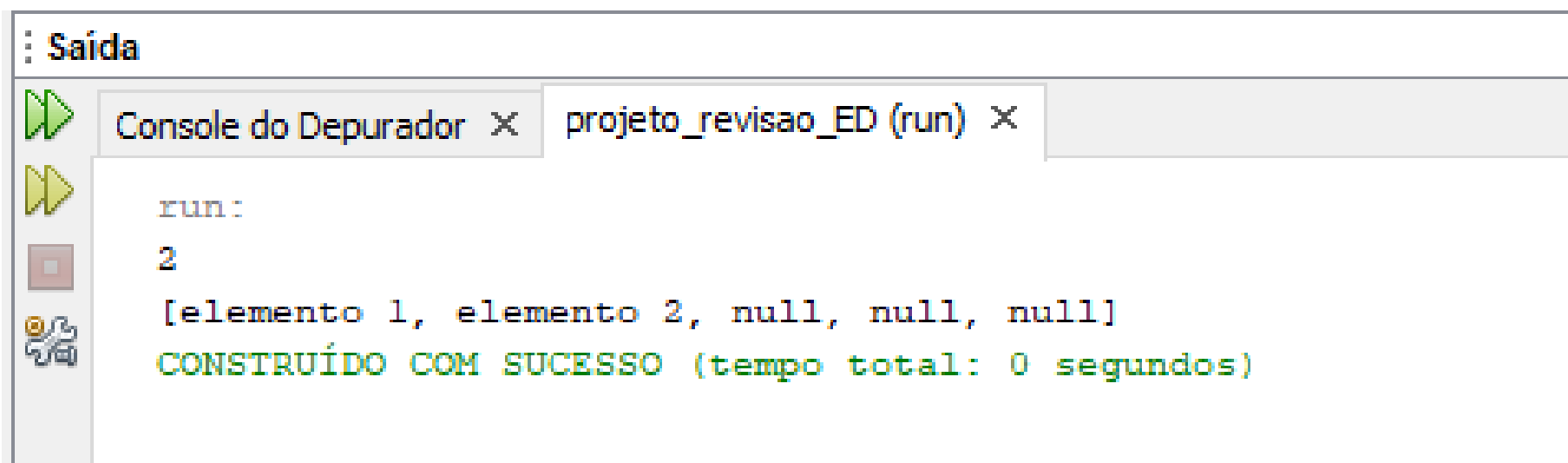
Vamos testar:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17    }
18 }
```


Arranjos – Array (Vetores ou Matrizes)

Resultado:



The screenshot shows a Java IDE's output console. At the top, there's a tab labeled "Saída". Below it, there are two tabs: "Console do Depurador" and "projeto_revisao_ED (run)". The "projeto_revisao_ED (run)" tab is active, displaying the following output:

```
run:
2
[elemento 1, elemento 2, null, null, null]
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

OK, porém, ele apresenta também os valores do vetor que estão com null, então vamos reescrever nosso método public String toString() para mostrar somente as posições com valores



`@Override`

```
public String toString(){
    StringBuilder s = new StringBuilder();
    s.append("[");

    for(int i=0; i<this.tamanho-1; i++){
        s.append(this.elementos[i]);
        s.append(", ");
    }

    if(this.tamanho> 0){
        s.append(this.elementos[this.tamanho-1]);
    }

    s.append("]");

    return s.toString();
}
```

O método `append` serve para adicionar mais conteúdo ao final de um `StringBuffer`.

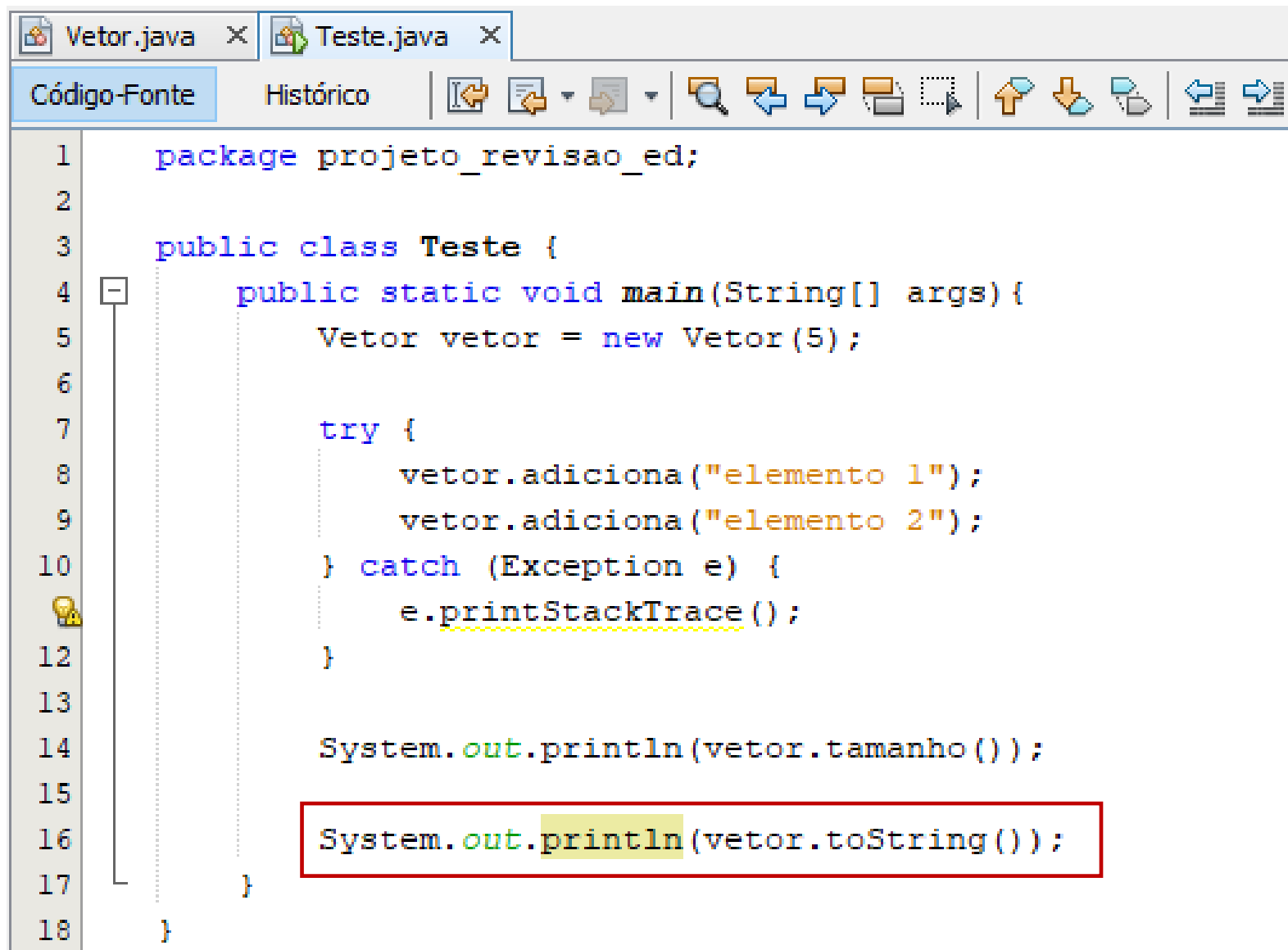
@Override. É uma anotação marcadora que deve ser usada apenas com métodos. Serve para indicar que o método anotado está sobrescrevendo um método da superclasse.

A classe `StringBuilder` faz parte do pacote `java`. Essa classe permite criar e manipular dados de `Strings` dinamicamente, ou seja, podem criar variáveis de `String` modificáveis.

Arranjos – Array (Vetores ou Matrizes)

Vamos testar:

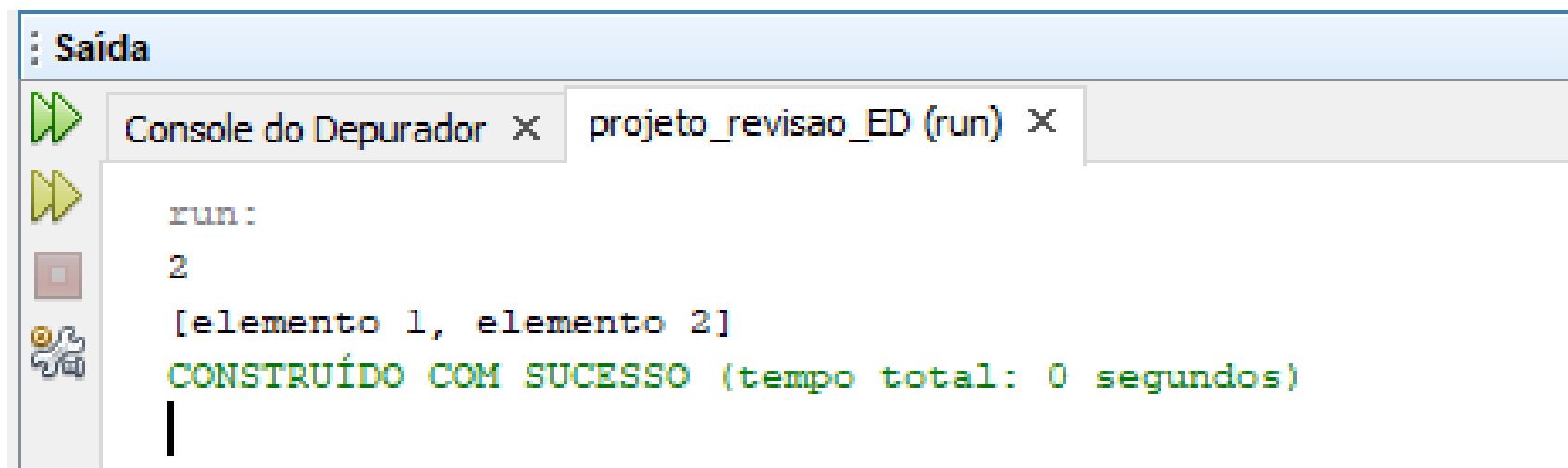
Outra Vez...



```
Vetor.java x Teste.java x
Código-Fonte Histórico
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) {
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17    }
18 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:



The screenshot shows a web browser's developer console with the 'Saída' (Output) tab selected. The console displays the following output:

```
run:  
2  
[elemento 1, elemento 2]  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)  
|
```

Nesse caso o resultado fica muito melhor e mais *clean* sem apresentar um monte de valores *null*.

Arranjos – Array (Vetores ou Matrizes)

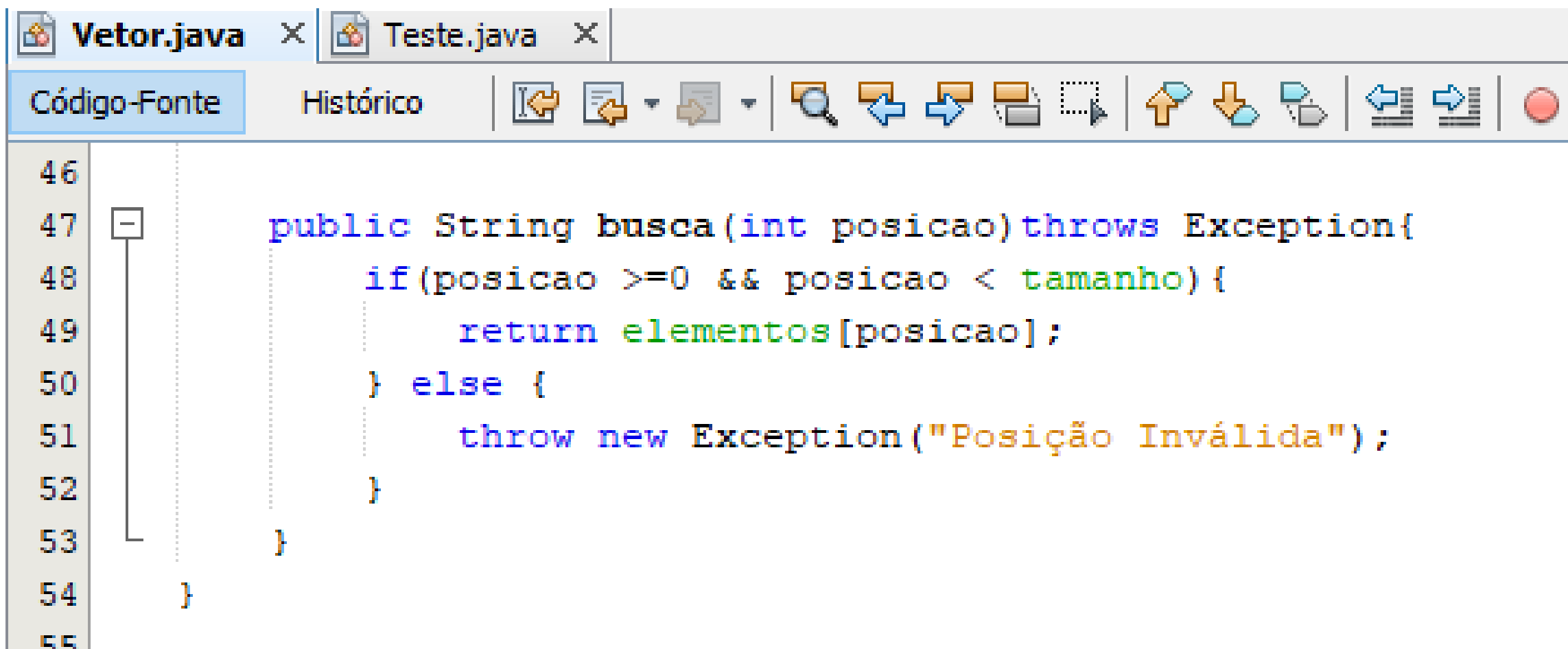
OK estamos no caminho certo...



Agora vamos determinar como obter um elemento de uma determinada posição, para isso vamos criar o método de **busca** na classe **Vetor**.

Arranjos – Array (Vetores ou Matrizes)

Método de Busca:



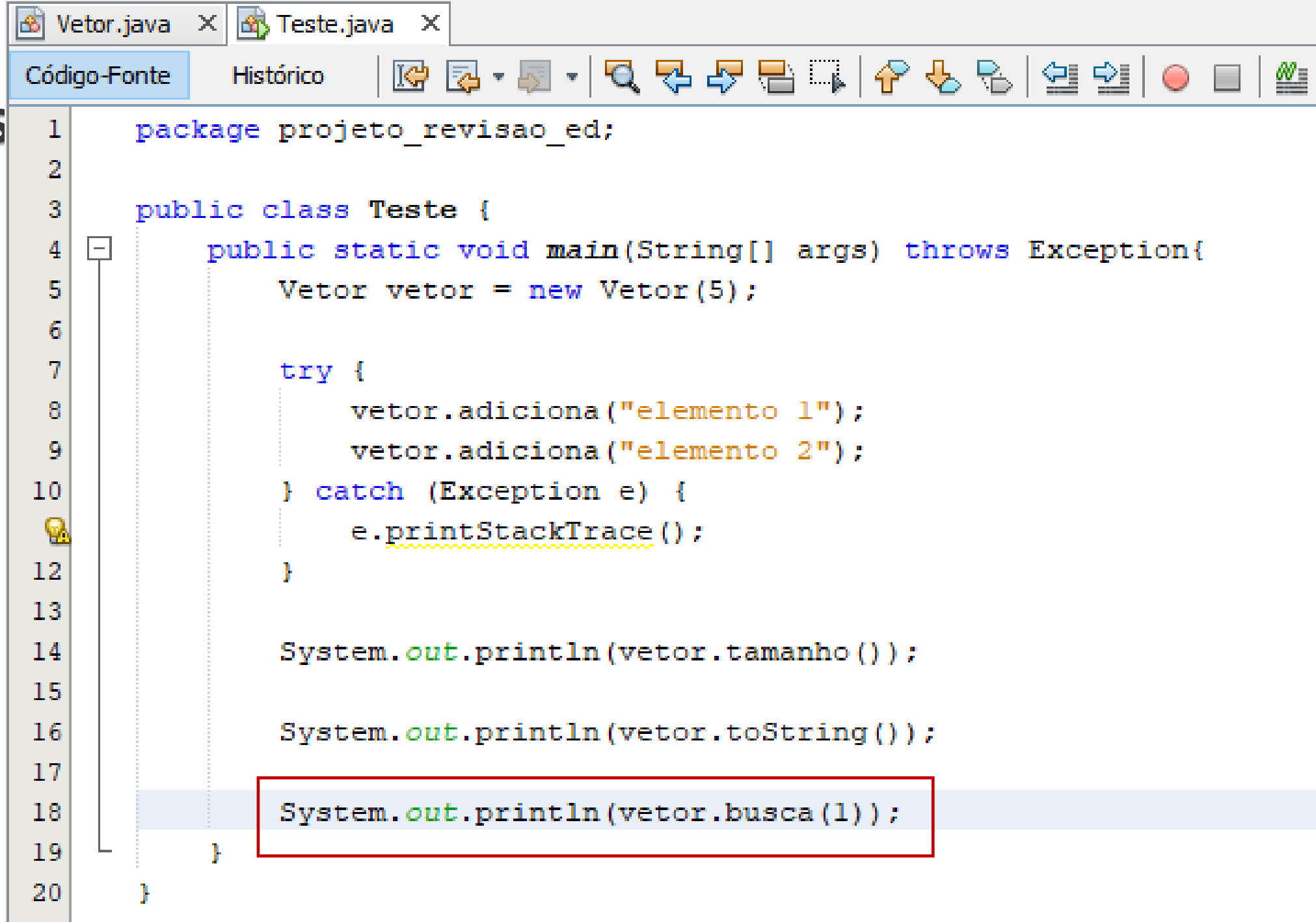
The screenshot shows an IDE window with two tabs: 'Vetor.java' and 'Teste.java'. The 'Vetor.java' tab is active, showing a code editor with a search method. The code is as follows:

```
46  
47 public String busca(int posicao) throws Exception {  
48     if (posicao >= 0 && posicao < tamanho) {  
49         return elementos[posicao];  
50     } else {  
51         throw new Exception("Posição Inválida");  
52     }  
53 }  
54  
55
```

The code is color-coded: keywords like 'public', 'String', 'throws', 'Exception', 'if', 'return', 'else', 'throw', and 'new' are in blue; variables like 'posicao', 'tamanho', and 'elementos' are in green; and literals like 'Posição Inválida' are in orange. The editor has a toolbar with various icons for navigation and editing. A line number margin is on the left, and a 'Histórico' button is next to the 'Código-Fonte' button.

Arranjos

Testando:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17
18        System.out.println(vetor.busca(1));
19    }
20 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:

⋮ Saída - projeto_revisao_ed (run)



run :



2

[elemento 1, elemento 2]



elemento 2



CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

|

Arranjos – Array (Vetores ou Matrizes)

Vamos provocar um erro propositalmente para testar a exceção quando solicitamos uma posição que não está preenchida.



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17        System.out.println(vetor.busca(3));
18    }
19 }
20 }
```



Proj... x Arquivos Serviços

projeto_revisao_ed

- Pacotes de Códigos-fonte
 - projeto_revisao_ed
 - Teste.java
 - Vetor.java
- Bibliotecas

Navegador x

Membros <vazio>

Teste

- main(String[] args)

Código-Fonte Histórico

```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17        System.out.println(vetor.busca(3));
18    }
19 }
20
21
```

Saída - projeto_revisao_ed (run)

```
run:
2
Exception in thread "main" java.lang.Exception: Posição Inválida
[elemento 1, elemento 2]
    at projeto_revisao_ed.Vetor.busca(Vetor.java:51)
    at projeto_revisao_ed.Teste.main(Teste.java:18)
C:\Users\WIN10\AppData\Local\NetBeans\Cache\8.2rc\executor-snippets\run.xml:53: Java returned: 1
FALHA NA CONSTRUÇÃO (tempo total: 0 segundos)
```



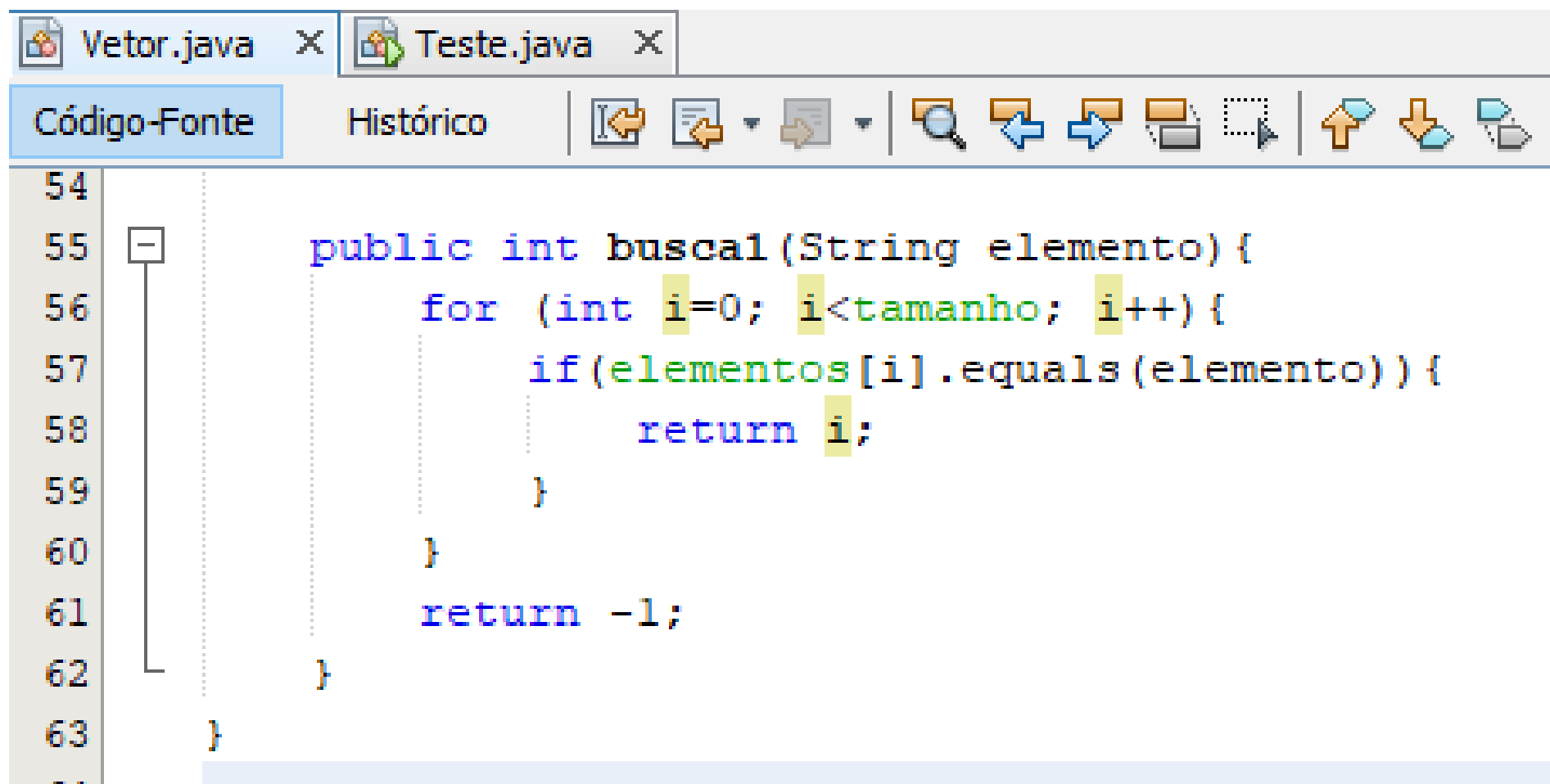
Arranjos – Array (Vetores ou Matrizes)

Vamos fazer uma verificação para determinar se um determinado elemento existe no vetor, e em caso de positivo, ele deve informar a posição no vetor, para isso, vamos modificar o nosso método de busca.

Sabemos que existe vários algoritmos que podem ser utilizados para busca, porém vamos usar o mais simples nesse exemplo, que é a busca sequencial.

Arranjos – Array (Vetores ou Matrizes)

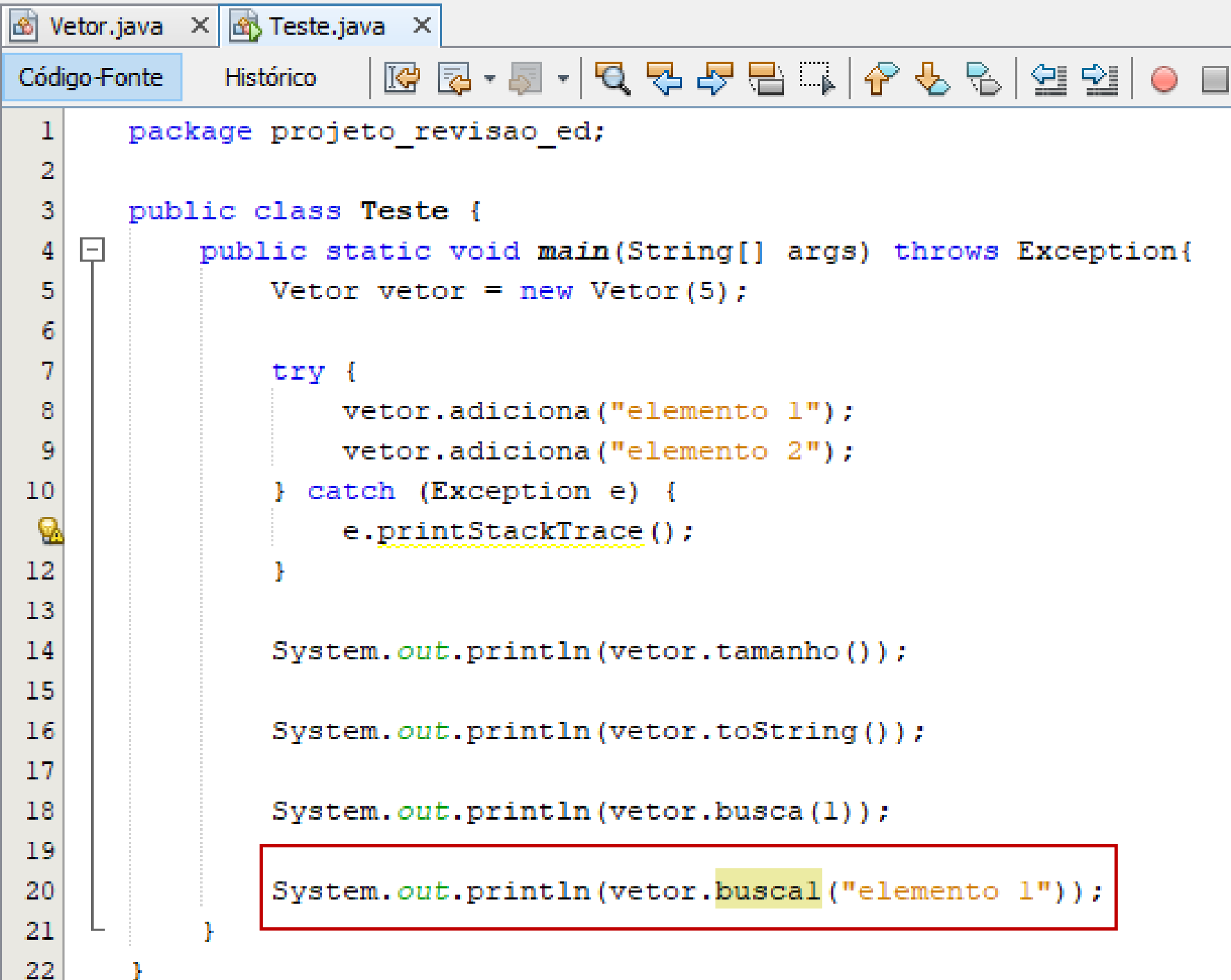
Novo método de busca:



The screenshot shows an IDE window with two tabs: 'Vetor.java' and 'Teste.java'. The 'Código-Fonte' (Source Code) tab is active. The code is written in Java and implements a search method named 'busca1'. The code is as follows:

```
54  
55 public int busca1(String elemento) {  
56     for (int i=0; i<tamanho; i++) {  
57         if(elementos[i].equals(elemento)) {  
58             return i;  
59         }  
60     }  
61     return -1;  
62 }  
63 }
```

The code is color-coded: keywords are in blue, strings in green, and variables/numbers in black. The 'tamanho' variable is highlighted in yellow in the original image. The 'elementos' array is highlighted in green in the original image. The 'i' variable is highlighted in yellow in the original image. The 'return i;' statement is highlighted in yellow in the original image. The 'return -1;' statement is highlighted in yellow in the original image. The '}' at the end of the method is highlighted in yellow in the original image. The '}' at the end of the class is highlighted in yellow in the original image.



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.tamanho());
15
16        System.out.println(vetor.toString());
17
18        System.out.println(vetor.busca(1));
19
20        System.out.println(vetor.busca("elemento 1"));
21    }
22 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:

```
 Saída - projeto_revisao_ed (run)

run:
2
[elemento 1, elemento 2]
elemento 2
0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
|
```

Arranjos – Array (Vetores ou Matrizes)

Vamos Adicionar um novo elemento em qualquer posição, imagine que eu queira inserir um valor na primeira posição, por exemplo:

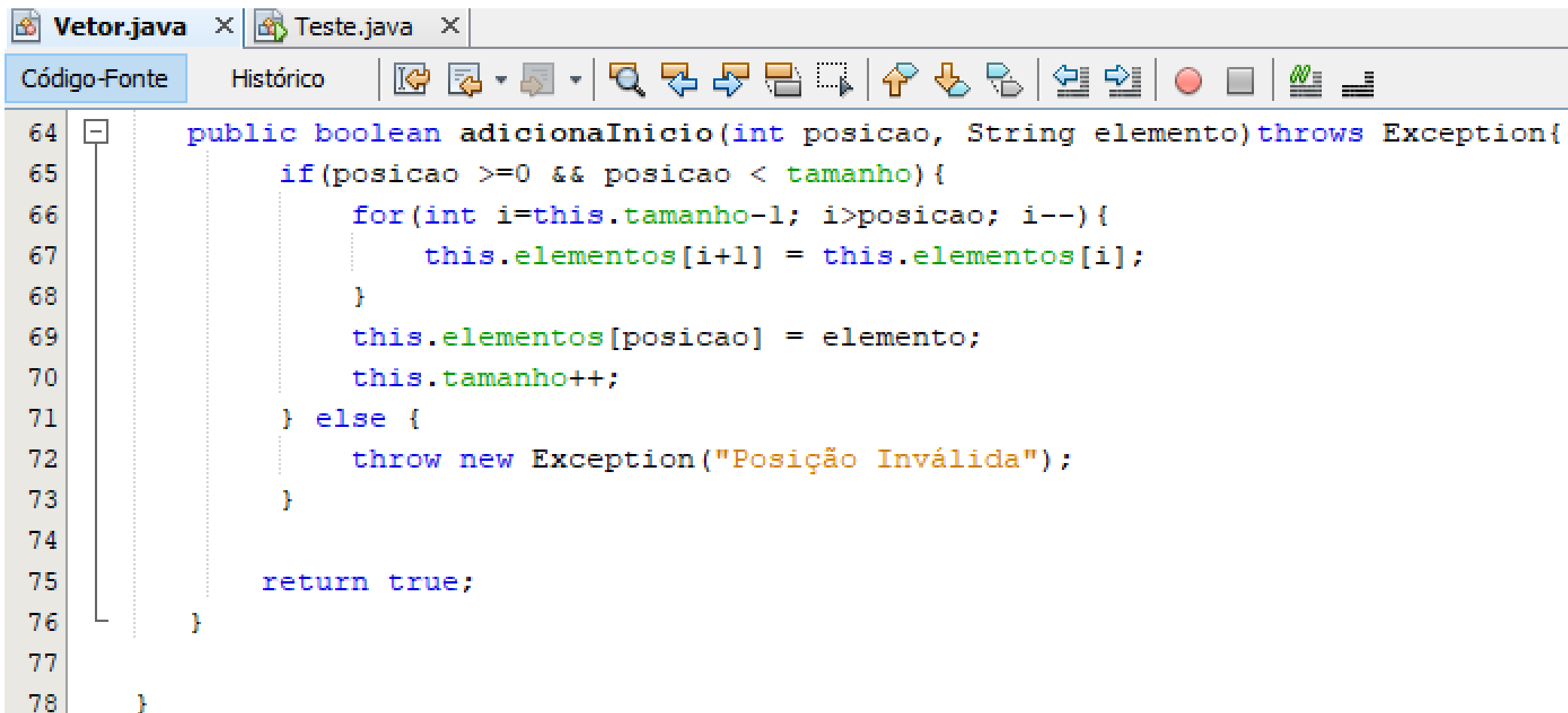
Elemento 1	Elemento 2	null	null	null
Posição 0	Posição 1	Posição 2	Posição 3	Posição 4

Caso eu queira inserir uma informação como “elemento 0” na posição 0 e mover demais valores para as posições seguintes, ficando como na imagem abaixo:

Elemento 0	Elemento 1	Elemento 2	null	null
Posição 0	Posição 1	Posição 2	Posição 3	Posição 4

Arranjos – Array (Vetores ou Matrizes)

Vamos criar um novo método na classe Vetor:



The screenshot shows an IDE with two tabs: 'Vetor.java' and 'Teste.java'. The 'Vetor.java' tab is active, showing the source code. The code is in Java and implements the 'adicionaInicio' method. The method signature is 'public boolean adicionaInicio(int posicao, String elemento) throws Exception'. The logic checks if 'posicao' is within the bounds of the array. If it is, it shifts all elements from 'posicao' to the end of the array one position to the right, then inserts 'elemento' at 'posicao' and increments the 'tamanho' attribute. If 'posicao' is invalid, it throws an 'Exception' with the message 'Posição Inválida'. The method returns 'true' if successful.

```
64 public boolean adicionaInicio(int posicao, String elemento) throws Exception{
65     if(posicao >=0 && posicao < tamanho){
66         for(int i=this.tamanho-1; i>posicao; i--){
67             this.elementos[i+1] = this.elementos[i];
68         }
69         this.elementos[posicao] = elemento;
70         this.tamanho++;
71     } else {
72         throw new Exception("Posição Inválida");
73     }
74
75     return true;
76 }
77
78 }
```



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10        } catch (Exception e) {
11            e.printStackTrace();
12        }
13
14        System.out.println(vetor.adicionaInicio(0, "elemento 0"));
15
16        System.out.println(vetor.tamanho());
17
18        System.out.println(vetor.toString());
19    }
20 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:

```
⋮ Saída - projeto_revisao_ed (run)
run:
true
3
[elemento 0, elemento 2, elemento 2]
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
|
```

Arranjos – Array (Vetores ou Matrizes)

Agora imagine que durante a execução de suas tarefas você identificou uma necessidade de armazenar mais informações, ou seja, tem que aumentas o tamanho do vetor.



Arranjos – Array (Vetores ou Matrizes)

Normalmente quando trabalhamos com vetor, nem sempre sabemos o tamanho exato, para isso, precisamos desenvolver códigos que possamos determinar o tamanho do nosso vetor de forma dinâmica, de acordo com a nossa necessidade.



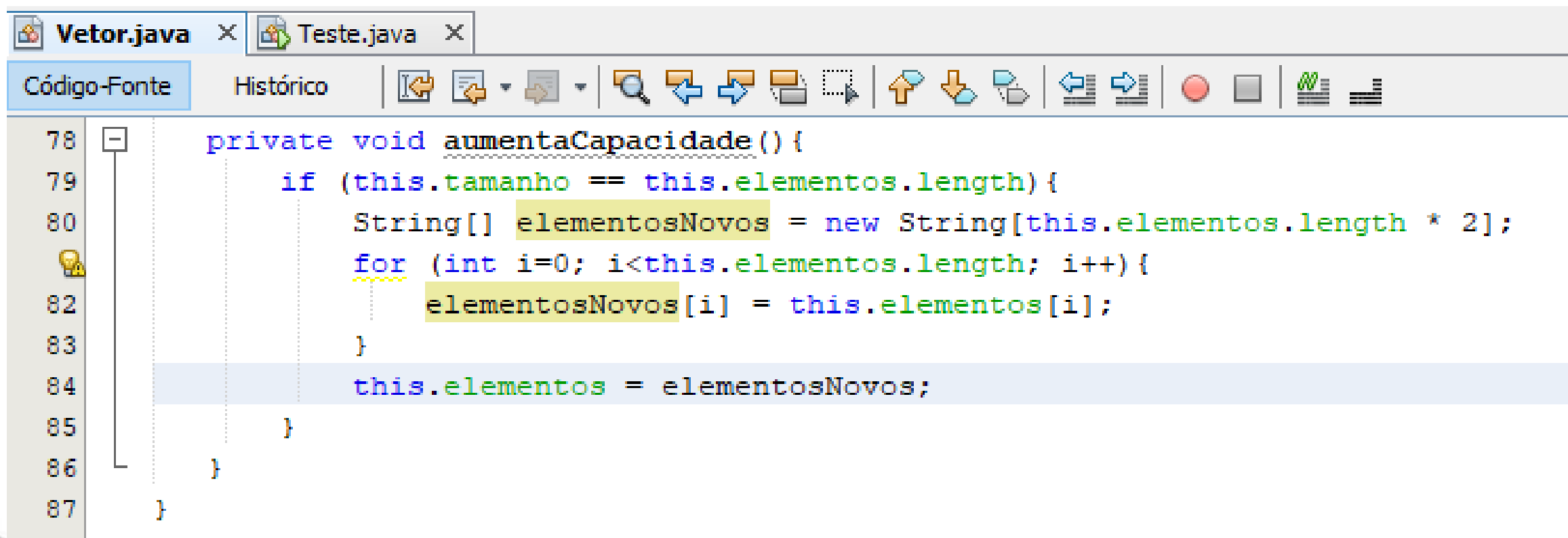


Arranjos – Array (Vetores ou Matrizes)

Como **não podemos modificar o nosso vetor depois de inicializado**, vamos ter que **criar um processo** para criar um novo vetor com maior capacidade (recomenda-se dobrar o tamanho quando ele estourar) e fazer uma copia dos elementos do vetor que acabou a capacidade para o novo vetor e depois vamos atribuir ao vetor inicial o novo vetor (sobrepor).

Arranjos – Array (Vetores ou Matrizes)

Vamos criar um novo método na classe Vetor chamado **aumentaCapacidade()**:



```
78 private void aumentaCapacidade() {
79     if (this.tamanho == this.elementos.length) {
80         String[] elementosNovos = new String[this.elementos.length * 2];
81         for (int i=0; i<this.elementos.length; i++){
82             elementosNovos[i] = this.elementos[i];
83         }
84         this.elementos = elementosNovos;
85     }
86 }
87 }
```

Arranjos – Array (Vetores ou Matrizes)

Lembrando que esse código só vai ser executado quando atingir a capacidade máxima do vetor, então precisamos inserir esse método dentro dos métodos de adição de novos elementos.



Arranjos – Array (Vetores ou Matrizes)

```
public void adiciona(String elemento) throws Exception{  
    this.aumentaCapacidade();  
    if (this.tamanho < this.elementos.length){  
        this.elementos[this.tamanho] = elemento;  
        this.tamanho++;  
    } else {  
        throw new Exception("O Vetor já está cheio, "  
            + "não é possível adicionar novos elementos");  
    }  
}
```

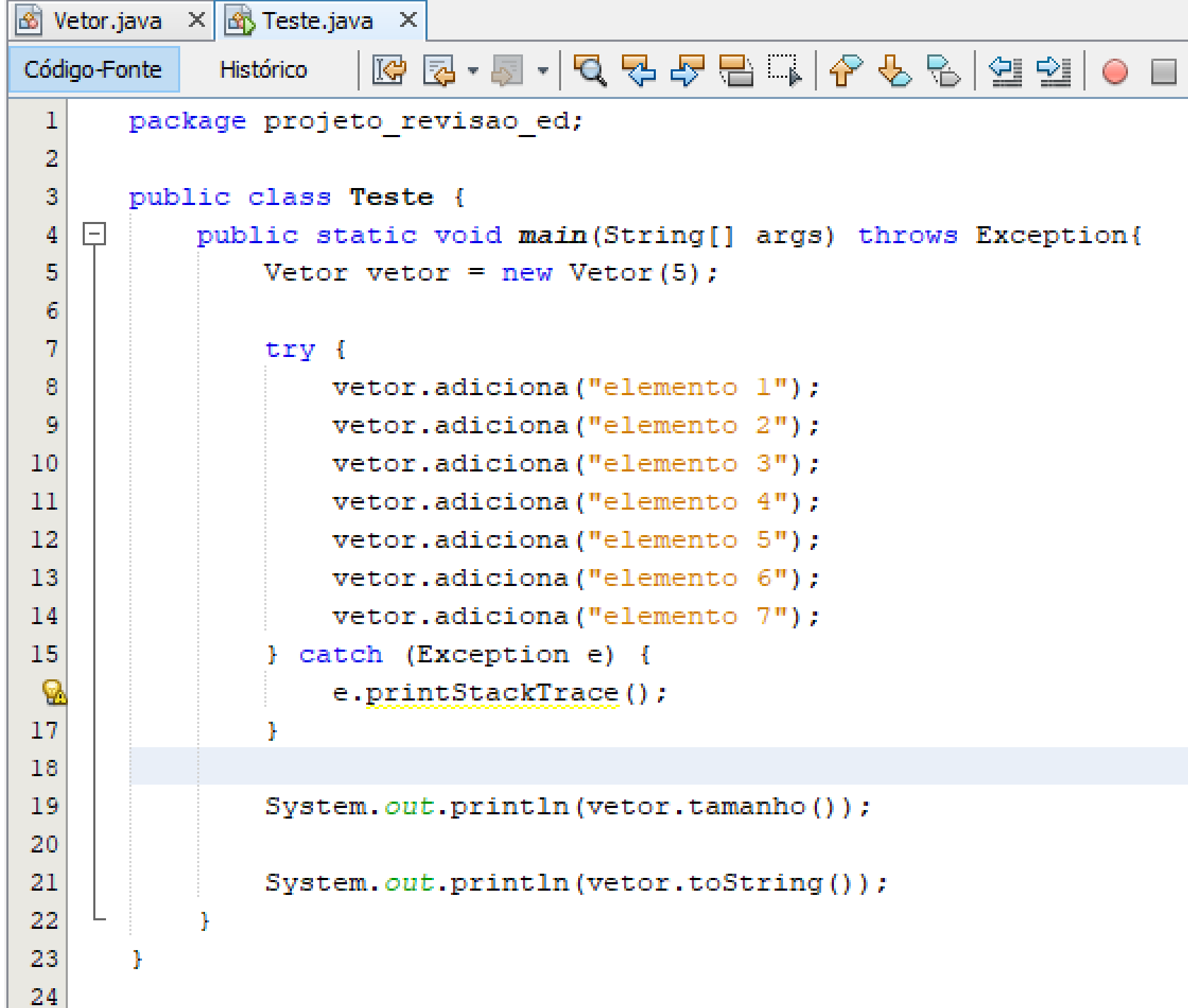
Arranjos – Array (Vetores ou Matrizes)

E também:

```
public boolean adicionaInicio(int posicao, String elemento) throws Exception{
    this.aumentaCapacidade();
    if(posicao >=0 && posicao < tamanho){
        for(int i=this.tamanho-1; i>posicao; i--){
            this.elementos[i+1] = this.elementos[i];
        }
        this.elementos[posicao] = elemento;
        this.tamanho++;
    } else {
        throw new Exception("Posição Inválida");
    }

    return true;
}
```

Teste:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10            vetor.adiciona("elemento 3");
11            vetor.adiciona("elemento 4");
12            vetor.adiciona("elemento 5");
13            vetor.adiciona("elemento 6");
14            vetor.adiciona("elemento 7");
15        } catch (Exception e) {
16            e.printStackTrace();
17        }
18
19        System.out.println(vetor.tamanho());
20
21        System.out.println(vetor.toString());
22    }
23 }
24
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:

⋮ Saída - projeto_revisao_ed (run)



run:



7

[elemento 1, elemento 2, elemento 3, elemento 4, elemento 5, elemento 6, elemento 7]



CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)



Arranjos – Array (Vetores ou Matrizes)

Vamos agora remover um elemento do vetor:

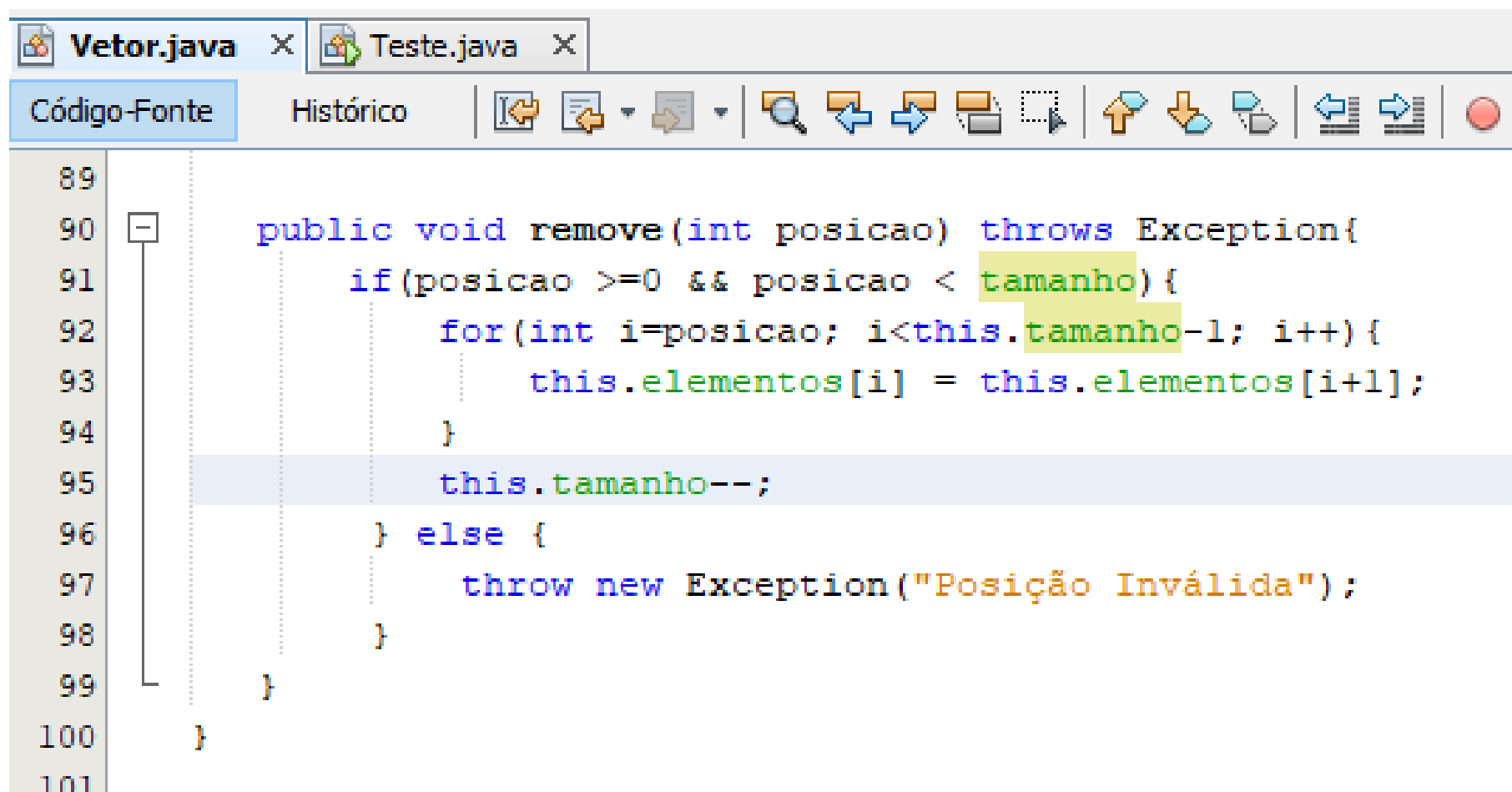
Imagine que precisamos remover um elemento do vetor abaixo, no caso o “**Elemento 3**” que está na posição 2.

vetor

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5	Elemento 6	Elemento 7	null	null	null
0	1	2	3	4	5	6	7	8	9

Arranjos – Array (Vetores ou Matrizes)

Vamos criar o método remove():

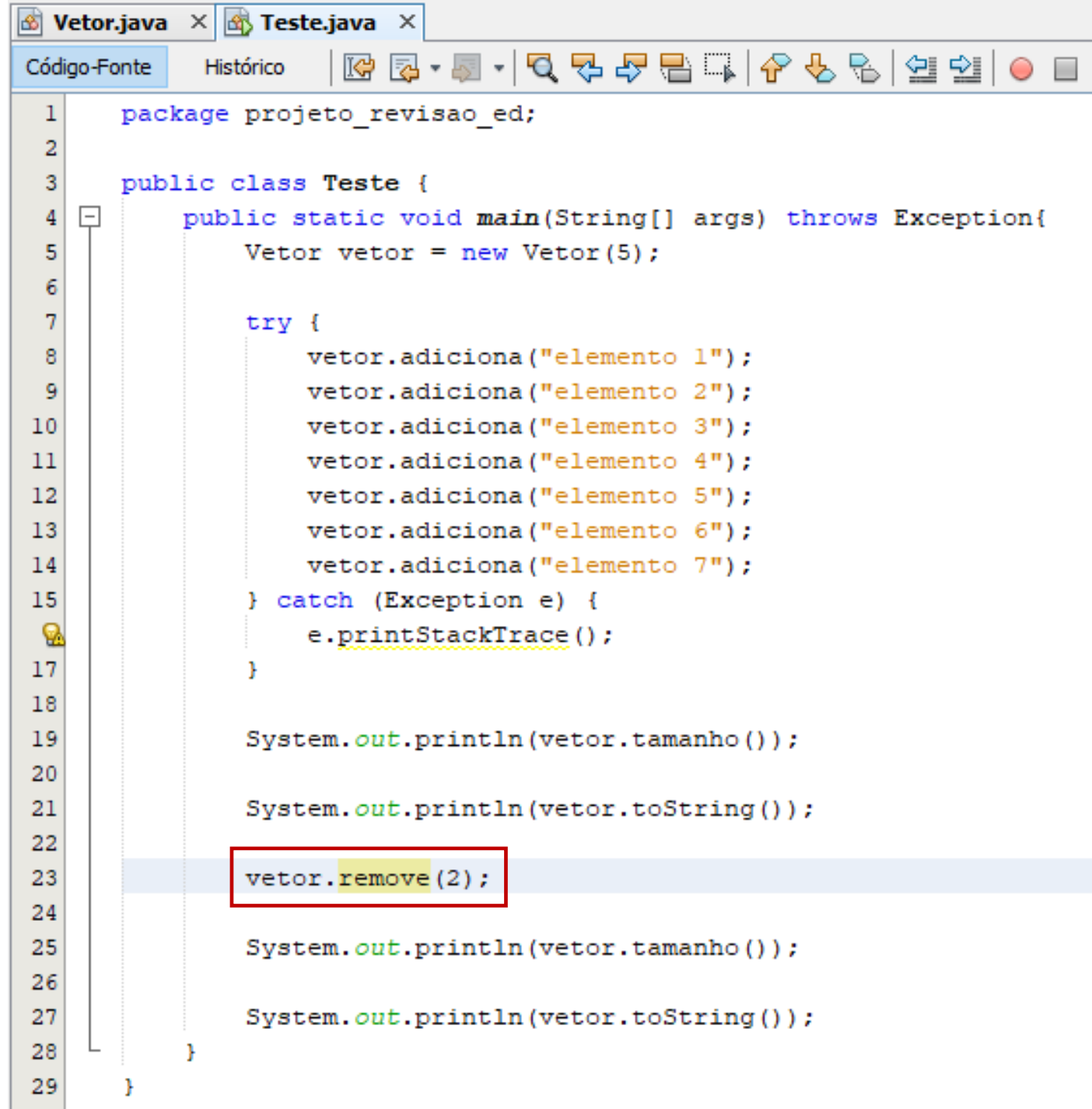


The screenshot shows an IDE with two tabs: 'Vetor.java' and 'Teste.java'. The 'Código-Fonte' (Source Code) tab is active. The code is as follows:

```
89  
90 public void remove(int posicao) throws Exception{  
91     if(posicao >=0 && posicao < tamanho){  
92         for(int i=posicao; i<this.tamanho-1; i++){  
93             this.elementos[i] = this.elementos[i+1];  
94         }  
95         this.tamanho--;  
96     } else {  
97         throw new Exception("Posição Inválida");  
98     }  
99 }  
100 }  
101
```

The line `this.tamanho--;` on line 95 is highlighted in blue. The IDE interface includes a toolbar with various icons for navigation and editing.

Teste:



```
1 package projeto_revisao_ed;
2
3 public class Teste {
4     public static void main(String[] args) throws Exception{
5         Vetor vetor = new Vetor(5);
6
7         try {
8             vetor.adiciona("elemento 1");
9             vetor.adiciona("elemento 2");
10            vetor.adiciona("elemento 3");
11            vetor.adiciona("elemento 4");
12            vetor.adiciona("elemento 5");
13            vetor.adiciona("elemento 6");
14            vetor.adiciona("elemento 7");
15        } catch (Exception e) {
16            e.printStackTrace();
17        }
18
19        System.out.println(vetor.tamanho());
20
21        System.out.println(vetor.toString());
22
23        vetor.remove(2);
24
25        System.out.println(vetor.tamanho());
26
27        System.out.println(vetor.toString());
28    }
29 }
```

Arranjos – Array (Vetores ou Matrizes)

Resultado:

Saída - projeto_revisao_ed (run)



```
run:
```

```
7
```

```
[elemento 1, elemento 2, elemento 3, elemento 4, elemento 5, elemento 6, elemento 7]
```

```
6
```

```
[elemento 1, elemento 2, elemento 4, elemento 5, elemento 6, elemento 7]
```

```
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

```
|
```


"O sucesso é a soma de
pequenos esforços
repetidos dia após dia"



Robert Collier

Obrigado!

Se precisar ...

Prof. Claudio Benossi

cbenossi@cruzeirodosul.edu.br

