INE5603 Introdução à POO

Prof. A. G. Silva

25 de setembro de 2017

Vetores *Arrays*

Baseado nos materiais de

Omero Francisco Bertol – UTFPR

Vetores (*arrays*) (1/2)

Vetores são estruturas de dados que armazenam usualmente uma quantidade fixa de dados de um certo tipo; por esta razão, também são conhecidos como estruturas homogêneas de dados.

Internamente, um vetor armazena diversos valores, cada um associado a um número que se refere à posição de armazenamento, e é conhecido como índice. Os vetores são estruturas indexadas, em que cada valor que pode ser armazenado em uma certa posição (índice) é chamado de elemento do vetor.

Cada elemento do vetor pode ser utilizado individualmente de forma direta, ou seja, pode ser lido ou escrito diretamente, sem nenhuma regra ou ordem preestabelecida, fazendo dos vetores estruturas de dados de acesso aleatório.

Vetores (*arrays*) (2/2)

O número de posições de um vetor corresponde ao seu tamanho. Um vetor de tamanho 10 tem este número de elementos, isto é, pode armazenar até dez elementos distintos. Os diferentes elementos de um vetor são distinguidos unicamente pela posição que ocupam no vetor. Cada posição de um vetor é unicamente identificada por um valor inteiro positivo, linear e sequencialmente numerado. Ou seja:

vetor[i] "i-ésimo" elemento do vetor, sendo que o valor da variável "i" deve pertencer ao intervalo do índice do vetor, ou seja, $0 \leq i \leq (\text{vetor.length}-1)$

O Java é uma linguagem com vetores *zero-based*, isto é, as posições do vetor iniciam a numeração a partir do valor "0". Portanto, um vetor de tamanho 10 tem índices iniciados em 0 prosseguindo até o 9.

Em Síntese: Características Básicas

- Estrutura de dados homogênea e indexada
- Todos os elementos da estrutura são igualmente acessíveis
  - Tempo e tipo de procedimento para acessar qualquer um dos elementos do vetor são "iguais"
- Cada elemento componente deste tipo de estrutura de dados tem um nome próprio que é o nome do vetor seguido da posição, ou índice

  nomeDaVariávelVetor[posição]

Declarando Variáveis do Tipo Vetor

Na declaração de vetores deverão ser fornecidas três informações: o nome do vetor, o número de posições do vetor (seu tamanho) e o tipo de dado que será armazenado no vetor. A declaração de um vetor para "inteiros", de nome "vetor", em Java:

```
int vetor[];                    // declaração do vetor
```

Podemos notar que as declarações de vetores são semelhantes às declarações de variáveis. Os elementos sintáticos que diferenciam as variáveis do tipo vetor das outras variáveis são os colchetes. Embora declarado, o vetor não está pronto para uso, sendo necessário reservar espaço para seus elementos (uma operação de alocação de memória).

```
vetor = new int[10];          // alocação de espaço para vetor
```

Na alocação de espaço, utilizamos o operador new (uma das palavras reservadas da linguagem) para reservar espaço para 10 (dez) elementos do tipo **int**. As duas declarações podem ser combinadas em um única, mais compacta:

```
int vetor[] = new int[10];  // declaração combinada
```

Declarando Vetores

Na linguagem Java, um vetor é uma "classe" e, portanto, deve-se utilizar o método new, que ativa o método construtor correspondente, para criar instâncias, ou exemplares, da classe vetor do tipo selecionado (int, byte, long, char, float, …)

Exemplos de declarações:

```
int vetor = new int[10];        // um vetor de 10 nº inteiros
long vetor = new long[10];      // um vetor de 10 nº inteiros (longo)

float vetor = new float[10];    // um vetor de 10 nº reais (ponto
                                // flutuante simples)

double vetor = new double[10];      // um vetor de 10 nº reais
                                    // (ponto flutuante duplo)
```

Vetores: individualizando valores

x[Expressão]

onde:

x nome da variável do tipo vetor

Expressão **posição** que define qual o elemento da estrutura de dados está sendo referenciado. ***Atenção***: *deve ser um valor pertencente ao intervalo do índice da variável*.

Observação: Os elementos de um vetor tem todas as características de uma variável comum e podem aparecer livremente em expressões e atribuições.

```
vetor[5] = vetor[0] * vetor[i+j];
```

Em Síntese:

```
int n = 5; // tamanho do vetor

// declaração e alocação de espaço
// para o vetor "v"
int v[] = new int[n];

int i; // índice ou posição

// processando os "n" elementos do vetor "v"
for (i=0; i<n; i++) {
  System.out.println(v[i]); // i-ésimo
                            elemento do vetor "v"
}
// representação interna:
```

v[0]	v[1]	v[2]	v[3]	v[4]

```java
import java.util.Scanner;

public class _01 {

  public static void main(String[] args) {
    Scanner ler = new Scanner(System.in);

    int n = 5; // tamanho do vetor
    int v[] = new int[n]; // declaração do vetor "v"
    int i; // índice ou posição

    for (i=0; i<n; i++) {
      System.out.printf("Informe %do. elemento de %d: ",
        (i+1), n);
      v[i] = ler.nextInt();
    }

    System.out.printf("\n");
    for (i=0; i<n; i++) {
      System.out.printf("v[%d] = %d\n", i, v[i]);
    }
  }

}
```

```
Saída - String (run) #4
run:
Informe 1o. elemento de 5: 11
Informe 2o. elemento de 5: 3
Informe 3o. elemento de 5: 10
Informe 4o. elemento de 5: 7
Informe 5o. elemento de 5: 2

v[0] = 11
v[1] = 3
v[2] = 10
v[3] = 7
v[4] = 2
CONSTRUÍDO COM SUCESSO (tempo total: 12 segundos)
```

```java
// Soma os elementos de um vetor, de tamanho 10, armazenados
// aleatoriamente.
public class SomaVetor {

  public static void main(String args[]) {
// declaração da variável do tipo "vetor" de inteiros longo (long)
// com tamanho n = 10
    int n = 10;
    long vetor[] = new long[n];

// Entrada- alimenta o vetor com nros aleatórios entre 0 e 10.
    for (int i=0; i<n; i++)
      vetor[i] = Math.round(Math.random() * 10);

// Processamento- percorre e soma os elementos do vetor.
    int sm = 0;
    for (int i=0; i<n; i++)
      sm = sm + vetor[i];

// Saída- imprime os elementos do vetor e o somatório calculado.
    for (int i=0; i<n; i++)
      System.out.println("Vetor[" + i + "] = " + vetor[i]);

    System.out.println("-------------");
    System.out.println("Soma = " + sm);
  }
}
```

Inicialização de Vetores

Java permite a inicialização de vetores no momento da declaração:

```
int v[] = {3, 2, 7, 5, 0};
```

Isso significa que **v[0]** terá o valor 3, **v[1]** terá o valor 2, **v[2]** terá o valor 7, **v[3]** terá o valor 5 e **v[4]** terá o valor 0.

Ainda é possível conhecer o tamanho do vetor por meio do campo **length**.

```
public class Exemplo {

  public static void main(String[] args) {
    int i, sm = 0, v[] = {3, 2, 7, 5, 0};

    for (i=0; i<v.length; i++) {
      System.out.printf("v[%d] = %d\n", i, v[i]);
      sm = sm + v[i];
    }

    System.out.printf("\nSoma = %d\n", sm);
  }

}
```

```
Saída - Vetores (run)
run:
v[0] = 3
v[1] = 2
v[2] = 7
v[3] = 5
v[4] = 0

Soma = 17
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Estrutura “for” Aprimorada

Utilizada para percorrer os elementos de um vetor sem utilizar um contador:

```
for (parâmetro: nomeDoVetor)
   instrução;
```

Onde “parâmetro” tem duas partes: um tipo (deve corresponder ao tipo dos elementos no vetor) e um identificador, por exemplo: **`int nro`**. O identificador representa os valores sucessivos do vetor nas sucessivas iterações da instrução **`for`**.

```
public class Exemplo {

  public static void main(String[] args) {
    int i = 0, sm = 0, v[] = {3, 2, 7, 5, 0};

    for (int nro: v) {
      System.out.printf("v[%d] = %d\n", i, nro);
      sm = sm + nro;
      i++;
    }

    System.out.printf("\nSoma = %d\n", sm);
  }

}
```

```
Saída - Vetores (run)
run:
v[0] = 3
v[1] = 2
v[2] = 7
v[3] = 5
v[4] = 0

Soma = 17
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Vetor Multidimensional

Na linguagem Java, um vetor pode ser declarado e ter qualquer tipo de base, sendo possível criar vetores de vetores (de vetores etc.), alcançando assim o efeito da multidimensionalidade.

A declaração de um vetor bidimensional para "inteiros", de nome "matriz" em Java:

```
int matriz[][] = new int[3][5];    // 3 linhas x 5 colunas
```

Fazendo referência a um elemento do vetor bidimensional:

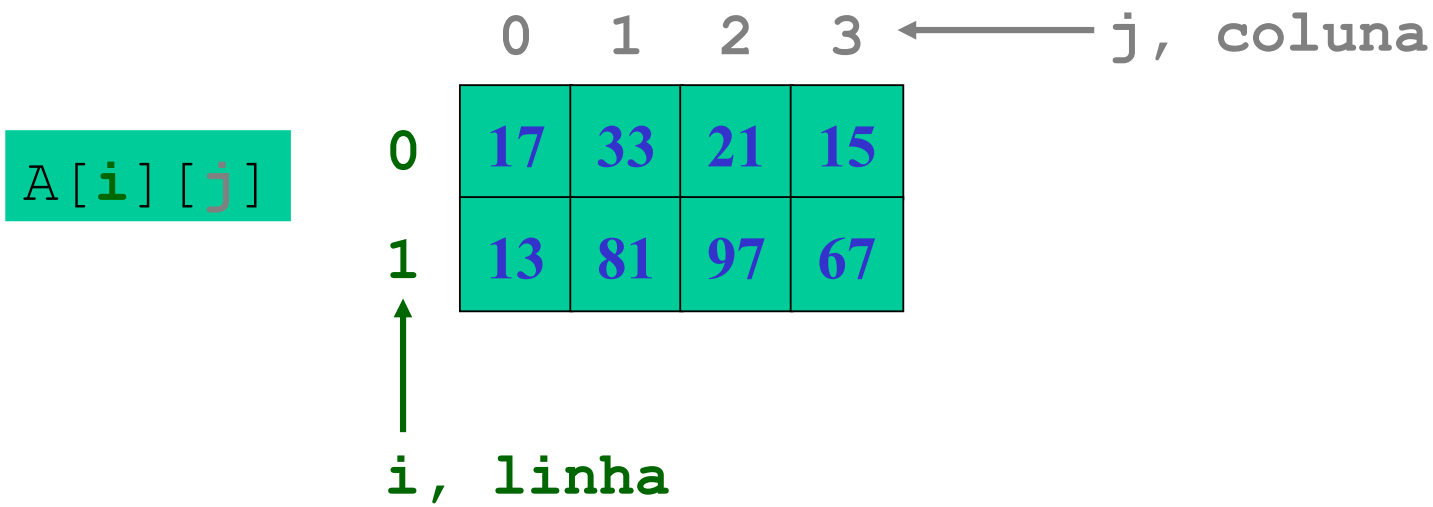
```
matriz[0][2] = 0;  // o terceiro elemento da primeira linha,
                   // ou ainda, o terceiro elemento do
                   // primeiro vetor; recebe o valor 0 (zero).
```

Declarando um vetor bidimensional ou matriz:

```
int A[][] = new int[2][4];
```

Referenciando as posições da matriz:

1a. Linha	2a. Linha
`A[0][0] = 17;`	`A[1][0] = 13;`
`A[0][1] = 33;`	`A[1][1] = 81;`
`A[0][2] = 21;`	`A[1][2] = 97;`
`A[0][3] = 15;`	`A[1][3] = 67;`



```java
import java.util.Scanner;
public class Exemplo {
  public static void main(String[] args) {
    Scanner ler = new Scanner(System.in);
    int i, j, m[][] = new int[2][4];

    for (i=0; i<2; i++) {
      System.out.printf("%da. linha.............\n", (i+1));
      for (j=0; j<4; j++) {
        System.out.printf("m[%d][%d] = ", i, j);
        m[i][j] = ler.nextInt();
      }
      System.out.printf("\n");
    }

    System.out.printf("\n");
    for (i=0; i<2; i++) {
      for (j=0; j<4; j++) {
        System.out.printf("%d ", m[i][j]);
      }
      System.out.printf("\n");
    }
  }
}
```

```
Saída - String (run) #4
run:
1a. linha.............
m[0][0] = 17
m[0][1] = 33
m[0][2] = 21
m[0][3] = 15

2a. linha.............
m[1][0] = 13
m[1][1] = 81
m[1][2] = 97
m[1][3] = 67


17 33 21 15
13 81 97 67
CONSTRUÍDO COM SUCESSO (tempo total: 54 segundos)
```

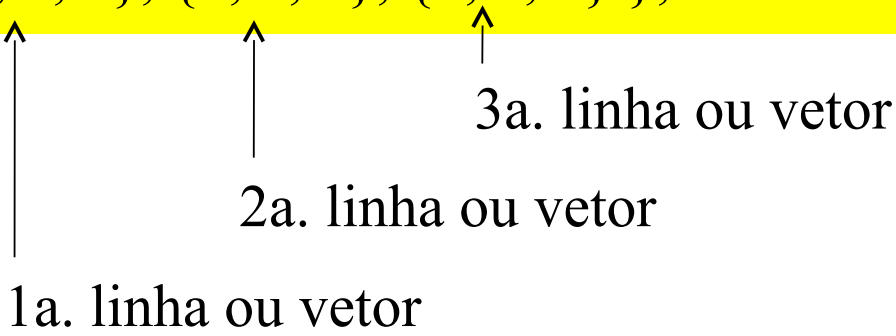
Declarando Vetor Bidimensional (1/2)

1) Com expressões de criação de vetores:

```
int m[][] = new int[3][3]; // matriz quadrada 3 linhas X 3 colunas
```

2) Declarando e inicializando:

```
int m[][] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```



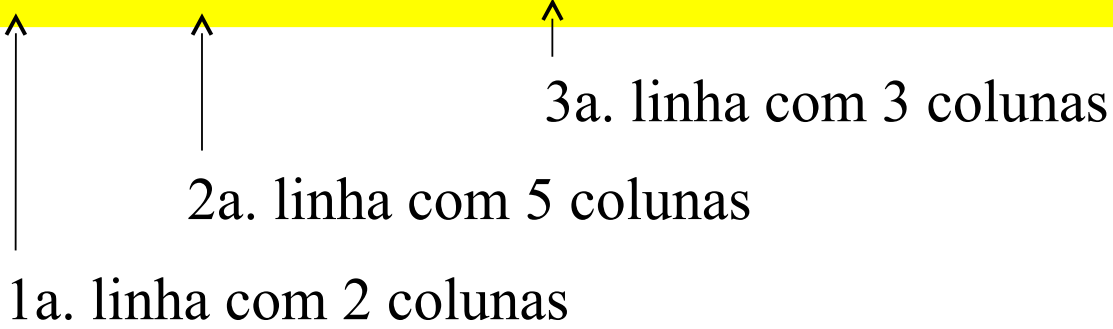
3) Com linhas de diferentes tamanhos:

```
int m[][] = new int[2][];        // cria 2 linhas
m[0] = new int[5];               // cria 5 colunas para a linha 0
m[1] = new int[3];               // cria 3 colunas para a linha 1
```

Declarando Vetor Bidimensional (2/2)

4) Declarando e inicializando linhas de diferentes tamanhos:

```
int m[][] = { {1, 2}, {4, 5, 6, 7, 8}, {9, 10, 11} };
```



Para conhecer os tamanhos dos vetores deve-se utilizar o campo **length**:

- m.length determina o número de linhas
- m[i].length determina o número de colunas da i-ésima linha

Percorrendo Vetores Bidimensionais

```
int m1[][] = { {1, 2, 3}, {4, 5, 6} };
int m2[][] = { {1, 2}, {3}, {4, 5, 6} };
```

m.length	determina o número de linhas
m[i].length	determina o número de colunas da i-ésima linha

```
for (int i=0; i<m.length; i++) {
  System.out.printf("%da. linha: ", (i+1));
  for (int j=0; j<m[i].length; j++) {
    System.out.printf("%d ", m[i][j]);
  }
  System.out.printf("\n");
}
```

Resultado com "m1":
1a. linha: 1 2 3
2a. linha: 4 5 6

Resultado com "m2":
1a. linha: 1 2
2a. linha: 3
3a. linha: 4 5 6

String (1/2)

Todos os tipos utilizados na linguagem Java, com exceção dos tipos primitivos (int, double, char e boolean), são "objetos". O tipo **String**, com **S** maiúsculo, é um dos objetos mais utilizados.

Ao contrário do que ocorre em C e C++, strings em Java não são tratadas como sequências de caracteres terminadas por NULL. São objetos, instâncias da classe **java.lang.String**.

Em resumo, as strings correspondem a uma sequência ou cadeia de caracteres delimitados por **aspas duplas**, que são armazenadas em "instâncias" da classe **String**, por exemplo:

```
String vazia = "";

String ola = "Alô Mundo Java !";
```

A **Tabela ASCII** (*American Standard Code for Information Interchange*) é usada pela maior parte da indústria de computadores para a troca de informações.

Decimal	Caractere
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

Decimal	Caractere
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

Decimal	Caractere
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z

Cada caractere na tabela ASCII estendida é representado por um código de 8 bits, ou seja, um byte.

$2^8 = 256$ caracteres:

0	31	Caracteres de Controle
32	127	ASCII Normal
128	255	ASCII Estendida

String (2/2)

Strings são objetos em Java, portanto, devem ser declarados e instanciados. Por exemplo:

```
// declaração
   String ola;

// instanciação
   ola = new String("Alô Mundo Java !");

// declaração e instanciação (mais prático)
   String ola = "Alô Mundo Java !";
   String nome = "Prof. Omero Francisco Bertol.";

// concatenação (\n = pula linha)
   String aula = ola + "\nby " + nome;
   System.out.println(aula);
// resultado
   Alô Mundo Java !
   by Prof. Omero Francisco Bertol.
```

Lendo uma String

Para ler uma String deve-se utilizar o método “nextLine” da classe **Scanner**:

```
import java.util.Scanner;

public class Exemplo {

  public static void main(String[] args) {
    Scanner ler = new Scanner(System.in);

    String s;

    System.out.printf("Informe uma String:\n");
    s = ler.nextLine();

    System.out.printf("\nString informada:\n");
    System.out.printf("%s\n", s);
  }

}
```

Lendo um Caractere

Para ler um caractere deve-se utilizar o método “read” do pacote de classes **System.in**:

```
public class Exemplo {

  public static void main(String args[])
     throws Exception {

     char sexo;

     System.out.println("Informe o Sexo (M/F): ");

     sexo = (char)System.in.read();

     if ((sexo == 'M') || (sexo == 'm'))
        System.out.println("(masculino).");
     else System.out.println("(feminino).");
  }
}
```

Métodos do Objeto String (1/4)

O objeto String em Java tem mais de 50 métodos.

char charAt(int pos)

retorna o caractere da posição "pos" (0 é a primeira posição)

String concat(String s)

retorna uma string com os caracteres deste objeto concatenados (no final) com os caracteres do argumento "s"

boolean contains(String s)

retorna verdadeiro se a sequência de caracteres do argumento "s" existe no objeto e falso caso contrário.

boolean equals(String s)

true, se as strings forem "exatamente" iguais

boolean equalsIgnoreCase(String s)

true, se as strings forem iguais (ignorando se os caracteres são maíusculos ou minúsculos)

Métodos do Objeto String (2/4)

`int indexOf(int ch)`
retorna o índice dentro da sequência de caracteres da primeira ocorrência do caractere especificado (ch).

`int indexOf(String s)`
retorna o índice dentro da sequência de caracteres da primeira ocorrência da substring especificada (s).

`int lastIndexOf(int ch)`
retorna o índice dentro da sequência de caracteres da última ocorrência do caractere especificado (ch).

`int lastIndexOf(String s)`
retorna o índice dentro da sequência de caracteres da última ocorrência da substring especificada (s).

obs: o valor -1 como retorno indica que não existe uma ocorrência.

Métodos do Objeto String (3/4)

`int length()`
retorna o tamanho da string, ou seja, a quantidade de caracteres da string

`String toLowerCase()`
converte os caracteres da string em "minúsculos"

`String toUpperCase()`
converte os caracteres da string em "maiúsculos"

`String trim()`
remove os espaços em branco do início e do final da string

`String replace(char oldChar, char newChar)`
troca o caractere indicado (oldChar) por um novo caractere (newChar) em "toda" a string

Métodos do Objeto String (4/4)

`String substring(int ini, int fim)`

retorna a substring com início na posição "ini" e final na posição "fim-1"

`static String valueOf(double d)`

retorna a representação string do argumento double.

`static String valueOf(float f)`

retorna a representação string do argumento float.

`static String valueOf(int i)`

retorna a representação string do argumento int.

`static String valueOf(long l)`

retorna a representação string do argumento long.

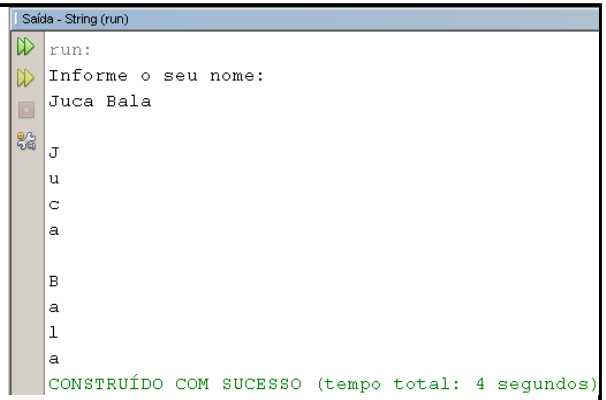
Documentação da Classe String

http://download.oracle.com/javase/1.5.0/docs/api/java/lang/String.html

Percorrendo os "n" caracteres de uma String (1/2):

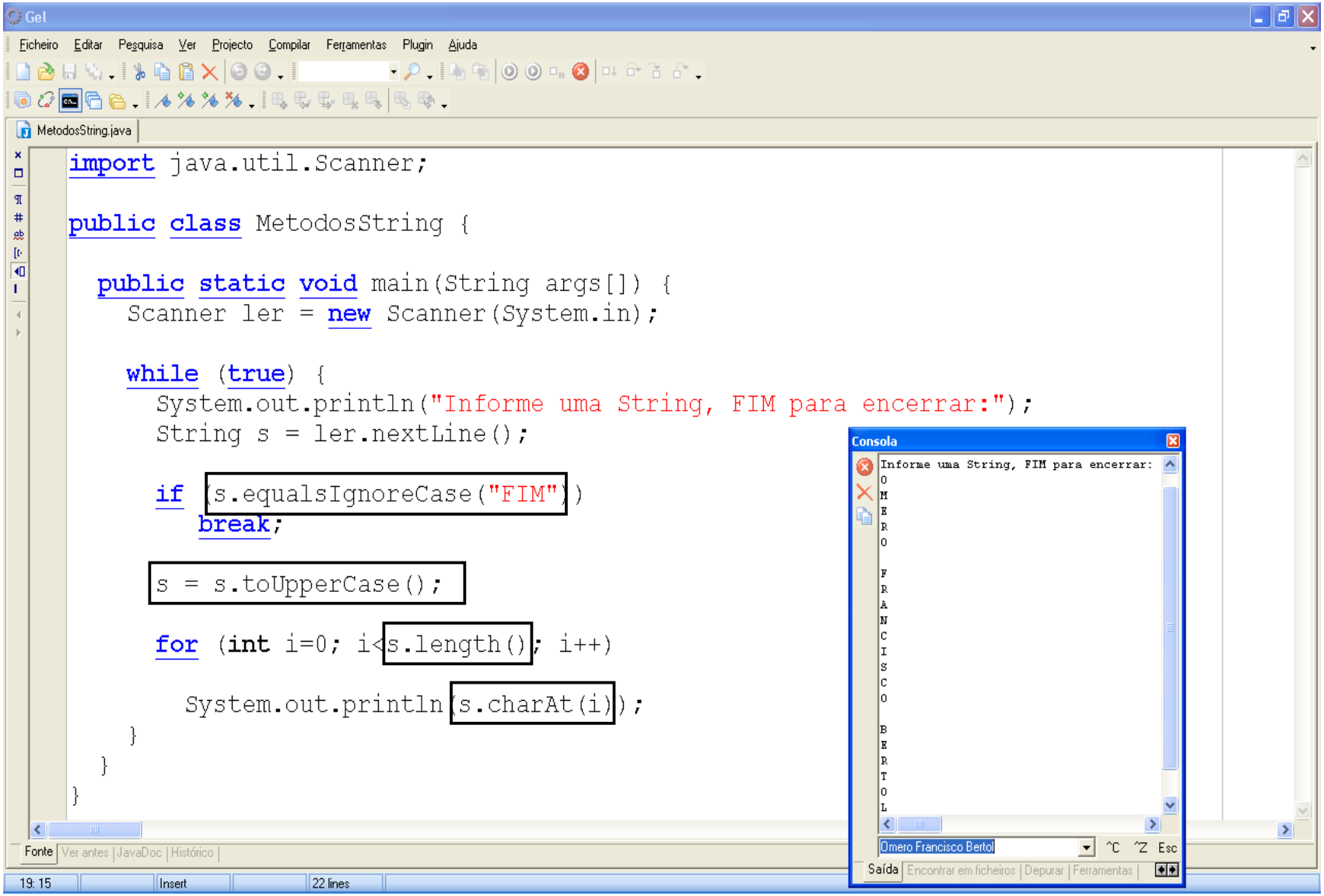
nome.charAt(i) indica o i-ésimo caractere da cadeia "nome".

```
import java.util.Scanner;

public class Exemplo {

  public static void main(String[] args) {
    Scanner ler = new Scanner(System.in);
    int i, n;
    String nome;

    System.out.printf("Informe o seu nome:\n");
    nome = ler.nextLine();

    n = nome.length(); // tamanho da String
    for (i=0; i<n; i++) {
      System.out.printf("\n%c", nome.charAt(i));
    }

    System.out.printf("\n");
  }

}
```



Percorrendo os "n" caracteres de uma String (2/2):

s.charAt(i) indica o i-ésimo caractere da cadeia "s".



Processando um caractere de uma cadeia (1/4):

`s.charAt(i)` indica o i-ésimo caractere da cadeia "s".

```
String s;
int ctmaius = 0, ctminus = 0, ctalfa = 0;

1) verificando se é um caractere alfabético maiúsculo
   if ((s.charAt(i) >= 'A') && (s.charAt(i) <= 'Z'))
      ctmaius = ctmaius + 1;

2) verificando se é um caractere alfabético minúsculo
   if ((s.charAt(i) >= 'a') && (s.charAt(i) <= 'z'))
      ctminus = ctminus + 1;

3) verificando se é um caractere alfabético
   if (((s.charAt(i) >= 'A') && (s.charAt(i) <= 'Z')) ||
       ((s.charAt(i) >= 'a') && (s.charAt(i) <= 'z')))
      ctalfa = ctalfa + 1;
```

Processando um caractere de uma cadeia (2/4):

`s.charAt(i)` indica o i-ésimo caractere da cadeia "s".

```
String s;
int ctnum = 0, cta = 0, ctbit = 0, ctespaco = 0;

4) verificando se é um caractere numérico
   if ((s.charAt(i) >= '0') && (s.charAt(i) <= '9'))
      ctnum = ctnum + 1;

5) verificando se é a vogal 'a'
   if ((s.charAt(i) == 'A') || (s.charAt(i) == 'a'))
      cta = cta + 1;

6) verificando se é um bit ('0' ou '1')
   if ((s.charAt(i) == '0') || (s.charAt(i) == '1'))
      ctbit = ctbit + 1;

7) verificando se é o espaço em branco
   if (s.charAt(i) == ' ')
      ctespaco = ctespaco + 1;
```

Processando um caractere de uma cadeia (3/4):

s.charAt(i) indica o i-ésimo caractere da cadeia "s".

```
String s;
int cta = 0, cte = 0, cti = 0, cto = 0, ctu = 0;

8) verificando as vogais separadamente
    if ((s.charAt(i) == 'A') || (s.charAt(i) == 'a'))
       cta = cta + 1;
    else if ((s.charAt(i) == 'E') || (s.charAt(i) == 'e'))
            cte = cte + 1;
        else if ((s.charAt(i) == 'I') || (s.charAt(i) == 'i'))
                cti = cti + 1;
            else if ((s.charAt(i) == 'O') || (s.charAt(i) == 'o'))
                    cto = cto + 1;
                else if ((s.charAt(i) == 'U') || (s.charAt(i) == 'u'))
                        ctu = ctu + 1;
```

Processando um caractere de uma cadeia (4/4):

`s.charAt(i)` indica o i-ésimo caractere da cadeia "s".

```
String s;
int ctcons = 0;

9) verificando se é uma consoante: todo caractere
   alfabético que não é vogal
   if (((s.charAt(i) >= 'A') && (s.charAt(i) <= 'Z')) ||
       ((s.charAt(i) >= 'a') && (s.charAt(i) <= 'z'))) {

      if ((s.charAt(i) != 'A') && (s.charAt(i) != 'a') &&
          (s.charAt(i) != 'E') && (s.charAt(i) != 'e') &&
          (s.charAt(i) != 'I') && (s.charAt(i) != 'i') &&
          (s.charAt(i) != 'O') && (s.charAt(i) != 'o') &&
          (s.charAt(i) != 'U') && (s.charAt(i) != 'u')) {
         ctcons = ctcons + 1;
      }

   }
```

Criando vetores com "valores iniciais"

```
public class NomeDosMeses {

  public static void main(String args[]) {

    String nomeMes[] = {"janeiro", "fevereiro", "marco", "abril",
        "maio", "junho", "julho", "agosto", "setembro", "outubro",
        "novembro", "dezembro"};

    System.out.println("Mes- Nome do Mes");
    System.out.println("----------------");

// Java é uma linguagem com vetores zero-based, ou seja, a
numeração // dos elementos do vetor inicia a partir do valor "0".
Como neste
// caso estamos representanto os nomes dos "12" meses do ano, o mês
// de dezembro corresponde a última posição do vetor, ou seja, "11".

    for (int i=0; i<12; i++)
      System.out.printf("%0,2d- %s\n", (i+1), nomeMes[i]);

    System.out.println("----------------");

  }

}
```

Tamanho de Variáveis do Tipo Vetor

O tamanho, ou quantidade, de elementos de um vetor pode ser determinado em "tempo de execução" através do atributo, ou variável-membro, **length**.

No exemplo abaixo, o vetor "nome" é preenchido com as cadeias de caracteres: joão, maria, ana, luiz e juca; imprimindo o contéudo do vetor percorrendo os índices de "**0**" até o "**tamanho do vetor - 1**". obs. -1, novamente, porque Java é uma linguagem com vetores *zero-based*.

```
public class NomesDosAmigos {

  public static void main(String args[]) {
    String nome[] = {"chico", "maria", "ana", "luiz", "juca"};

    System.out.println("Nomes dos Amigos");
    System.out.println("----------------");
    for (int i=0; i<nome.length; i++)
      System.out.println(nome[i]);

  }
}
```

CPF- Cadastro de Pessoa Física (1/2)

Para evitar erros de digitação de sequências de números de importância fundamental, como matrícula de um aluno, o CPF do Imposto de Renda, o número de conta bancária, geralmente se adiciona ao número um **dígito verificador**.

Por exemplo, o **CPF** 546471429 é usado como 5464714249, onde **49** são os dígitos verificadores, calculados da seguinte maneira:

Para calcular o **(1°)** digito verificador:

1. cada um dos nove primeiros algarismo é multiplicado por um peso começando de 10 e diminuindo de 1 a cada passo:
   5*10, **4***9, **6***8, **4***7, **7***6, **1***5, **4***4, **2***3, **9***2;
2. somam-se as parcelas obtidas:
   S = 50 + 36 + 48 + 28 + 42 + 5 + 16 + 6 + 18 = 249;
   . calcula-se o digito através da seguinte expressão:
   11 - (S % 11) = 11 - (249 % 11) = 11 - 7 = **4**

obs. se o resto encontrado for 10 ou 11, o dígito verificador será 0; nos outros casos, o dígito verificador é o próprio resto encontrado.

CPF- Cadastro de Pessoa Física (2/2)

Para calcular o **(2°)** digito verificador:

1. cada um dos dez primeiros algarismo é multiplicado por um peso começando de 11 e diminuindo de 1 a cada passo:
   5*11, **4***10, **6***9, **4***8, **7***7, **1***6, **4***5, **2***4, **9***3, **4***2;

2. somam-se as parcelas obtidas:
   55 + 40 + 54 + 32 + 49 + 6 + 20 + 8 + 27 + 8 = 299;

3. calcula-se o digito através da seguinte expressão:
   11 - (S % 11) =
   11 - (299 % 11) =
   11 - 2 = **9**

obs. se o resto encontrado for 10 ou 11, o dígito verificador será 0; nos outros casos, o dígito verificador é o próprio resto encontrado.

```java
public static boolean isCpf(String cpf) {
  int sm, i, r, num;
  char dig10, dig11;
// calcula o 1o. digito verificador do CPF
  sm = 0;
  for (i=0; i<9; i++) {                // por exemplo: transforma o caracter '0'
    num = (int)(cpf.charAt(i) - 48);   // no inteiro 0 (48 eh a posição de '0'
                                       // na tabela ASCII)
    sm = sm + (num * (10 - i));
  }
  r = 11 - (sm % 11);
  if ((r == 10) || (r == 11))
     dig10 = '0';
  else dig10 = (char)(r + 48);
// calcula o 2o. digito verificador do CPF
  sm = 0;
  for (i=0; i<10; i++) {
    num = (int)(cpf.charAt(i) - 48);
    sm = sm + (num * (11 - i));
  }
  r = 11 - (sm % 11);
  if ((r == 10) || (r == 11))
     dig11 = '0';
  else dig11 = (char)(r + 48);

// compara os dígitos calculados com os dígitos informados
  if ((dig10 == cpf.charAt(9)) && (dig11 == cpf.charAt(10)))
     return(true);
  else
     return(false);
}
```

Referência

- Introdução ao Java.
  - Peter Jandl Junior.
  - São Paulo: Berkeley - 2002.
  - Capítulo 2: Java e sua Sintaxe, pág. 51..54.
  - Capítulo 4: Aplicações de Console, pág. 122.