FICHA PRÁTICA 2

LABORATÓRIO DE ARRAYS

PROF. F. MÁRIO MARTINS

DI/UM

VERSÃO 1.0

2007

FICHA PRÁTICA 2

LABORATÓRIO DE ARRAYS

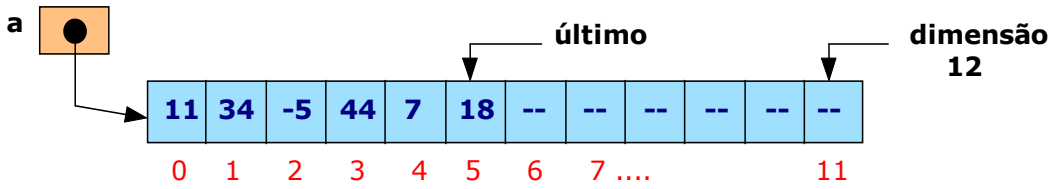
SÍNTESE TEÓRICA

Os ***arrays*** de JAVA são estruturas lineares indexadas, ou seja, cada posição do *array* possui um endereço inteiro para acesso ao elemento nele contido (1º elemento no índice 0). Os *arrays* podem conter valores de tipos primitivos ou objectos. Os *arrays* de JAVA não são objectos, ou seja, não são criados por nenhuma classe nem respondem a mensagens. São no entanto de tipo referenciado, ou seja, a variável que identifica o *array* contém o endereço de memória do *array* (é uma referência).

Sendo estruturas lineares indexadas, os **elementos** de um *array* ocupam **posições** referenciáveis por um **índice** inteiro com valores a partir de 0.

A dimensão física de um *array*, também designada a sua **capacidade**, pode ser definida aquando da sua declaração ou posteriormente, mas é diferente do seu **comprimento**, que se associa ao número efectivo de elementos que, num dado momento, estão armazenados no *array*.

Para um *array* de dimensão d, o seu comprimento actual c será sempre c <= d e o índice do último elemento será sempre c-1. Para um *array* **a**, a instrução **a.length** devolve um inteiro que corresponde à sua dimensão actual, não o actual número de elementos. Para arrays numéricos, inicializados a 0 ou 0.0 há quer ter cuidado com **length** pois os elementos a zero iniciais são contados também, e não correspondem a dados introduzidos. Assim, sempre que o número de elementos não coincida garantidamente com a dimensão, uma variável que conte os elementos efectivos do array deverá ser usada.



A dimensão física de um *array*, bem como o tipo dos seus elementos, são em geral definidos aquando da sua declaração, como em:

```
int[] vector = new int[100];
```

A dimensão pode, no entanto, ser definida posteriormente, usando a construção **new**, não podendo o *array* ser usado enquanto tal dimensão não for especificada.

```
String[] nomes;
nomes = new String[50];
```

A capacidade/dimensão definida para um *array* é fixa, ou seja, é imutável ao longo da execução do programa. A capacidade pode ser também definida de forma implícita e automática através da sua inicialização, sendo, neste caso, a capacidade do *array* igual ao número de elementos introduzidos na sua inicialização, cf. o exemplo:

```
int[] valores = { 12, 56, -6, 45, 56, 8 };  // dim = 6
double[] notas = { 12.5, 15.6, 10.9, 15.2, 6.6, 8.7, 9.0, 11.1 };  // dim = 8
```

Os *arrays* podem ser multidimensionais (linhas, colunas, etc.) e não apenas a uma só dimensão (linha). Os *arrays* monodimensionais são muitas vezes referidos como **vectores**.

Os *arrays* multidimensionais são em geral referidos como **matrizes**. O número de dimensões de um *array* é clarificado na sua definição, pois cada dimensão corresponde sintacticamente a mais um [].

```
int[][] matriz_valores = new int[20][50];          // matriz de 20 linhas por 50 colunas
double[][] notas = new double[3][300];             // matriz 3 testes por 300 alunos
double [][][] temps = new double[15][12][31];      // cidades x meses x dias e temperaturas
```

SINTAXE ESSENCIAL

1.- DECLARAÇÕES, INICIALIZAÇÕES E DIMENSIONAMENTO

```
int lista[];   // estilo C
int[] lista;   // estilo JAVA

int[] turma = new int[100];
double[] medias = new double[50];
byte[] mem = new byte[800*600];

short matriz[][] = new short[10][50];

short matx[][] = new short[10][]; // 2ª dimensão é variável
matx[0] = new short[15]; matx[1] = new short[40];

String[] nomes = new String[20];
String[] alunos = { “Pedro”, “Rita”, “Ana” };
String[][] linhas = { {“A”, “minha”}, {“casa”, “tem”, “um”}, {“rio”} };
String[][] galo = { {“O”, “O”, “X”},
                    {“X”, “X”, “O”},
                    {“O”, “X”, “O”} }
Ponto[] plano = new Ponto[200];
Object obj[] = new Object[100];
```

2.- COMPRIMENTO E ACESSO AOS ELEMENTOS

```
// comprimento
int comp = lista.length;
int numAlunos = alunos.length;

// acesso
int val = lista[0]; int num = lista[val*2];
short snum = matx[5][3];
String nome = nomes[index];
String pal = linhas[l][c];
out.println(lista[i]); out.println(nomes[i]);
out.printf(“Val = %d%n”, lista[i]);
```

3.- VARRIMENTO = ACESSO A TODOS OS ELEMENTOS

```
for(int i = 0; i <= a.length-1; i++) { ...a[i].....}   // por índice

for(IdTipo elem : IdArray) { …elem … }                 // for(each)
```

```
// Imprimir todos os elementos de um array
for(int i=0; i< lista.length; i++) out.println(lista[i]);

for(int elem : lista) out.println(elem);

// Exemplos de somatórios
int soma = 0;
for(int i=0; i< lista.length; i++) soma = soma + lista[i];

int soma1 = 0;
for(int elem : lista) soma1 += elem;

// Exemplos de concatenação de strings
String total = “”;
for(int i=0; I < nomes.length; i++) { total = total + nomes[i]; }

String total = “”;
for(String nome : nomes) { total += nome; }

// Contagem de pares e ímpares num array de inteiros
int par = 0, impar = 0;
for(int i = 0; i < a.lenght; i++)
  if (a[i]%2 == 0) par++; else impar++;
out.printf(“Pares = %d - Impares = %d%n”, par, impar);

// Total de inteiros > MAX de um array de arrays de inteiros
int maiores = 0; int MAX = Integer.MIN_VALUE;
for(int l = 0; l < nums.length; l++) {
   for(int c = 0; c < nums[l].length; c++)
      if (nums[l][c] > MAX) maiores++;
}

// Concatenação de strings de um array bidimensional
String[][] nomes = { {"Rita", "Pedro"}, ..... };
String sfinal = "";
for(int l = 0; l < nomes.length; l++) {
     for(int c = 0; c < nomes[l].length; c++) sfinal += nomes[l][c];
}

// usando for(each)
sfinal = "";
for(String[] lnomes : nomes)
     for(String nome : lnomes) sfinal += nome;
```

4.- Leitura de Valores para um array (usando a classe Input)

```
// Ler um número n, dado pelo utilizador, de valores de dado tipo, e guardá-los
// sequencialmente num array
int valor = 0;
out.print(“Quantos números inteiros quer introduzir ? “);
int n = Input.lerInt();
for(int i = 0; i <= n-1; i++) {
   valor = Input.lerInt(); lista[i] = valor;
}

// ou ainda, de forma mais simples mas equivalente:
int n = Input.lerInt(); int valor = 0;
for(int i = 0; i <= lista.length-1; i++) lista[i] = Input.lerInt();
```

```
// Ler um número não previamente fixado de valores de dado tipo e guardá-los
// num array pela sua ordem de leitura; Terá sempre que existir uma condição
// de paragem da leitura, seja porque foi lido um valor definido como valor de
// paragem (flag), seja porque o array já não tem mais capacidade.

int VALOR_STOP = -9999;          // valor que serve de sentinela/flag para parar a leitura
int[] lista = new int[MAXDIM];   // MAXDIM é uma constante predefinida no programa
boolean stop = false; int conta = 0; int valor;
while(!stop && conta<=MAXDIM-1) {
   valor = Input.lerInt();
   if(valor == VALOR_STOP)
         stop = true;
   else
         { lista[conta] = valor; conta++ }
}
```

5.- Algoritmo de Procura

```
// Procura de um valor lido (chave) num array, dando como resultado a sua
// posição, ou -1 caso não seja encontrado.

int[] lista = new int[MAXDIM];  // MAXDIM é uma constante predefinida no programa
..... // leitura ou inicialização
int chave; boolean encontrada = false;
int indice =0; int pos = -1;
out.print("Qual o valor a procurar no array? : ");
chave = Input.lerInt();
while(!encontrada && indice<=MAXDIM-1) {
   if(lista[indice] == chave) {
         encontrada = true; pos = indice;
   }
}
out.println("Valor " + chave + " encontrado na posição " + pos");
```

6.- Cópia Entre Arrays

System.arraycopy(array_fonte, índice_inicial_f, array_destino, índice_inicial_d, quantos);

```
System.arraycopy(a, 0, b, 0, a.length);  // a.length elementos de a[0] para b desde b[0]
System.arraycopy(lista, 5, lista1, 1, 4);  // 4 elems a partir de lista[5] para lista1 desde 1
```

7.- Métodos da Classe java.util.Arrays (tipo => tipo simples)

```
int binarySearch(tipo[] a, tipo chave);   // devolve índice da chave, se existir, ou < 0;
boolean equals(tipo[] a, tipo[] b);       // igualdade de arrays do mesmo tipo;
void fill(tipo[] a, tipo val);            // inicializa o array com o valor parâmetro;
void sort(tipo[] a);                      // ordenação por ordem crescente;
String toString(tipo[] a);                // representação textual dos elementos;

String deepToString(array_multidim);      // repres. textual para multidimensionais;
boolean deepEquals(array_multi1, array_multi2);   // igualdade de arrays multidim;
```

EXERCÍCIOS:

Ex1: Declarar, inicializar e imprimir os elementos de um *array* de inteiros.

```
// declarar, inicializar e imprimir os elementos de um array //
int[] lista = {12, 2, 45, 66, 7, 23, 99};
System.out.println("-----  ELEMENTOS DO ARRAY  ----------------------");
for(int i = 0; i < lista.length; i++) System.out.println("Elemento "+ i + "= " + lista[i]);
System.out.println("-------------------------------------------------");

// solução alternativa usando método da classe Arrays
int[] lista = {12, 2, 45, 66, 7, 23, 99};
out.println(Arrays.toString(lista));
```

Ex2: Escrever um programa que faça a leitura de N valores inteiros para um *array* e determine qual o maior valor introduzido e qual a sua posição no *array*.

```
import static java.lang.System.out;
public class ExArrays2 {
   public static void main(String[] args) {
      int n; // total de valores a serem lidos
      int[] lista = new int[100];
      int valor; int MAX = Integer.MIN_VALUE;
      int pos = -1;
      out.print("Numero de valores a ler: "); n = Input.lerInt();
      // leitura dos N valores para o array
      for(int i = 0; i <= n-1; i++) {
        out.print("Valor " + (i+1) + " : "); valor = Input.lerInt();
        lista[i] = valor; // lista[i] = Input.lerInt()
      }
      // determinação do MÁXIMO e da sua posição - SOLUÇÃO 1
      for(int i = 0; i < lista.length; i++)
         if(lista[i] > MAX) { MAX = lista[i]; pos = i; }
      out.println("Máximo1 = " + MAX + " - na posição: " + (pos + 1));
      // solução 2
      int ct = 0; MAX = Integer.MIN_VALUE;
      for(int elem : lista) {
         if(elem > MAX) { MAX = lista[ct]; pos = ct; }
         ct++;
      }
      out.println("Máximo2 = " + MAX + " - na posição: " + (pos + 1));
   }
}
```

Ex3: Modificar o programa anterior de modo a que a leitura dos N elementos para um *array* de inteiros seja realizada usando um método auxiliar que recebe o valor de N como parâmetro.

```
import static java.lang.System.out;
public class ExArrays3 {

   public static int[] leArrayInt(int n, int DIM) { // n = nº de elementos a serem lidos
      int[] a = new int[DIM];
      // leitura dos valores para o array
      for(int i = 0; i <= n-1; i++) {
         out.print(" Valor " + (i+1) + " : "); a[i] = Input.lerInt();
      }
      return a;
   }
```

```
    public static void main(String[] args) {
        int DIM = 100; int[] lista = new int[DIM]; int n, pos = 0;
        do {
            out.print("Número de elementos a ler: < " + DIM + " : ");
            n = Input.lerInt();
        }
        while(n > DIM);
        lista = leArrayInt(n, DIM);
        // determinação do MÁXIMO e da sua POSIÇÃO - 2 soluções
        int MAX = Integer.MIN_VALUE;
        for(int i =0; i<= lista.length-1; i++)
            if( lista[i] > MAX ) { MAX = lista[i]; pos = i; }
        out.println("Máximo1 = " + MAX + " na posição: " + (pos +1));
        int i = 0;
        // solução 2 usando foreach
        for(int elem : lista) {
         i++;
         if( elem> MAX) { MAX = lista[i]; pos = i; }
        }
        out.println("Máximo2 = " + MAX + " na posição: " + (pos + 1));
    }
}
```

Ex4: Modificar o programa anterior de modo a que quer a leitura dos N elementos quer a determinação do máximo elemento do *array* sejam realizados em métodos auxiliares do método main().

```
/**
 * Programa que usa um método auxiliar para ler inteiros válidos e inseri-los num array
* de inteiros que é devolvido como resultado. Um outro método auxiliar determina o
* maior elemento de um array com número de elementos dado como parâmetro.
* O programa apresenta-se, deste modo, devidamente estruturado.
 *
 * @author F. Mário Martins
 * @version 1.0/2005
 */

import static java.lang.System.out;
public class ExArrays4 {

  public static final int DIM = 100;

  public static int[] leArray(int num) {
     int[] nums = new int[num];
     int n = 0;
     for(int i = 0; i< num; i++) {
         out.print("Valor " + (i+1) + " : ");
         nums[i] = Input.lerInt();
     }
     return nums;
  }

  public static int max(int[] nums, int total) {
     int max = Integer.MIN_VALUE;
     for(int i = 0; i < total; i++)
       if (nums[i] > max) max = nums[i];
     return max;
  }
```

```
  public static void main(String[] args) {
    int[] arrayNum = new int[DIM];
    out.print("Total de números a ler: "); int dim = Input.lerInt();
    arrayNum = leArray(dim);
    int maximo = max(arrayNum, dim);
    out.println("Máximo = " + maximo);
    Arrays.sort(arrayNum);
    out.println("--  Array Ordenado  --");
    for(int i = 0; i<dim; i++) out.println(arrayNum[i]);
  }
}
```

Ex5: Escrever um programa que faça a leitura de N elementos inteiros para um *array,* mas que os insira de forma a que o *array* se mantenha sempre ordenado por ordem crescente.

```
/**
 * Programa que usa um método auxiliar para ler inteiros válidos e inseri-los num array
 * de inteiros que é mantido permanentemente ordenado.
 * Assim, a inserção de cada elemento é uma inserção ordenada. Quanso o elemento
 * a inserir é menor que algum dos elementos do array, tem que se fazer o "shift" de
todas
 * as posições a partir deste elemento maior, 1 posição para a frente.
 *
 * @author F. Mário Martins
 * @version 1.0/2005
 */

import static java.lang.System.out;
public class ExArrays5 {

  public static final int DIM = 100;

  public static int[] leArray(int num) {
     int[] nums = new int[num+1];
     int valor; int index;
     // i será sempre o número de elementos já inseridos no array
     for(int i = 0; i < num; i++) {
        out.print("Valor " + (i+1) + " : ");
        valor = Input.lerInt();
        // determina a posição para inserir => encontrar o 1º valor maior
        index = 0;
        while(nums[index] <= valor  && index < i ) { index++; }
        //
        if( i == 0) nums[0] = valor;
        else {
                // a partir do último elemento do array, faz shift de todos para a
                // posição seguinte no array
               for(int p = i; p > index; p--) nums[p] = nums[p-1];
               // insere o novo valor mantendo o array ordenado
               nums[index] = valor;
              }
        for(int x = 0; x <= i; x++) out.println(nums[x]);
      }
     return nums;
  }

  public static void main(String[] args) {
    int[] arrayNum = new int[DIM];
    out.print("Total de números a ler: "); int dim = Input.lerInt();
    arrayNum = leArray(dim);
    out.println("--  Array Ordenado  --");
```

```
    for(int i = 0; i < dim; i++) out.println(arrayNum[i]);
  }
}
```

Ex6: Escrever um programa que faça a leitura de N elementos inteiros para um *array,* receba dois índices válidos do *array* lido e crie um *array* apenas com os elementos entre esses dois índices. Usar um método auxiliar.

```
import static java.lang.System.out;
public class ExArrays6 {

  public static final int DIM = 100;

  public static int[] leArray(int num) {
     int[] nums = new int[num];
     for(int i = 0; i< num; i++) {
        out.print("Valor Indice " + i + " : ");
        nums[i] = Input.lerInt();
     }
     return nums;
  }

  public static int[] selecciona(int[] nums, int start, int stop) {
     int[] res = new int[stop-start+1];
     for(int i = 0; i <= stop-start; i++) res[i] = nums[start + i];
     return res;
  }

  public static void main(String[] args) {
    int[] arrayNum = new int[DIM]; int inicio, fim;
    out.print("Total de números a ler: "); int dim = Input.lerInt();
    arrayNum = leArray(dim);

    do {
      out.print("Indice inicial para selecção (0 ..): ");
      inicio = Input.lerInt();
    }
    while(inicio < 0 || inicio > dim -1);

    do {
       out.print("Indice final ( > inicial): ");
       fim = Input.lerInt();
    }
    while(fim < inicio || fim > dim - 1);

    int[] subarray = selecciona(arrayNum, inicio, fim);

    out.println("--  Array Resultado --");
    for(int i = 0; i <= subarray.length - 1; i++) out.println(subarray[i]);
  }
}
```

Ex7: Escrever um programa que leia uma série de palavras terminada por "zzz" para um array, aceite repetidamente uma palavra até que seja introduzida a palavra "xxx" e verifique se tal palavra existe no array. Caso exista o programa deverá removê-la do array.

```
import static java.lang.System.out;
public class Ex7Arrays {

   public static int MAXDIM = 100;
```

```
  public static int lePalavras(String[] palavras) {
     // preenche o array parâmetro e devolve o número de palavras lidas
     String palavra; int conta = 0;
     out.print("Palavra 1 : "); palavra = Input.lerString();
     while(!palavra.equals("zzz") && !palavra.equals("ZZZ") && conta < MAXDIM) {
        palavras[conta] = palavra; conta++;
        out.print("Palavra " + (conta + 1) + " : "); palavra = Input.lerString();
     }
     return conta;
  }

  public static int procuraPal(String[] palavras, String palavra) {
     boolean encontrada = false; int index;
     index = 0;
     while(!encontrada && index <= palavras.length-1) {
        if(palavras[index].equals(palavra)) encontrada = true;
        else index++;
     }
     return (encontrada ? index : 0);
  }

  public static String[] removePal(String[] palavras, int index, String pal) {
    // faz "shift down" desde indice+1 até length-1
    for(int p = index+1; p <=palavras.length-1; p++)
      palavras[p-1] = palavras[p];
    return palavras;
  }

  public static void main(String[] args) {
     String[] dicionario = new String[MAXDIM];
     String palavra, resp; int indice;
     int total = lePalavras(dicionario);
     out.println("----   DICIONÁRIO ----");
     for(int p = 0; p <= total-1; p++) out.println(dicionario[p]);
     out.println("----------------------");
     do {
        out.print("Palavra a remover: "); palavra = Input.lerString();
        indice = procuraPal(dicionario, palavra);
        out.println("Indice " + indice);
        if (indice == 0)
           out.println("PALAVRA NÃO EXISTENTE !!");
        else {
            dicionario = removePal(dicionario, indice, palavra);
            total--;
            out.println("----  DICIONÁRIO ACTUAL  ----");
            for(int i = 0; i <= total-1; i++)
              out.println(dicionario[i]);
            }
       out.println("Pretende remover mais palavras (S/*) ? ");
       resp = Input.lerString();
     }
     while(resp.equals("S") || resp.equals("s"));
     out.println("--- FIM DO PROGRAMA ---");
  }
}
```

Ex8: Escrever um programa que leia para um array os vencimentos mensais brutos (íliquidos) dos 20 funcionários de uma empresa. O programa possuirá uma tabela de retenção de IRS constante, do tipo

Salário Ilíquido	% Retenção de IRS
0 a 500 Euros	5
501 a 1000 Euros	10
1001 a 2000	20
2001 a 4000	30
4001 ou mais	40

Pretende-se que o programa crie um array no qual, para o respectivo funcionário cujo vencimento bruto se encontra no array lido, sejam introduzidos as respectivas retenções para IRS. No final, o programa deve apresentar uma listagem com os vencimento bruto, retenção de IRS e vencimento líquido para os 20 funcionários.

```
/**
 * Calculo de Retenção de IRS sobre vencimentos de funcionários.
 *
 * @author F. Mário Martins
 * @version 3/2007
 */
import static java.lang.System.out;
public class Ex8Arrays {

   public static int MAXDIM = 100;

   public static double[] leSalarios(int total) {
      double[] salarios = new double[total];
      for(int i = 0; i <= total - 1; i++) {
           out.print("Vencimento " + (i+1) + " : " );
           salarios[i] = Input.lerInt();
      }
      return salarios;
   }

   public static double calcula_imposto(int[] escaloes, int[] taxas,
                                double salario) {
      int index = 0;
      while(escaloes[index] < salario && index < escaloes.length) { index++; }
      return ((double)taxas[index])/100*salario;
   }

   public static void mostra_resultados(double[] vencs, double[] impostos) {
      out.println("\n\n-------------------------------------------------------");
      out.println("\tVENCIMENTO BRUTO\t IRS\t\tLÍQUIDO");
      for(int i = 0; i <= vencs.length - 1; i++)
         out.printf("N. %2d\t\t %8.2f\t%6.2f\t\t%8.2f%n",
                  (i+1), vencs[i], impostos[i], vencs[i]-impostos[i]);
      out.println("\n-------------------------------------------------------");
   }

   public static void main(String[] args) {
      int totalFunc;
      do {
         out.print("Numero Total de funcionários < " + MAXDIM + " : ");
         totalFunc = Input.lerInt();
      }
      while(totalFunc > MAXDIM);
      // arrays de vencimentos, taxas, escalões e impostos
      double[] vencimentos = new double[totalFunc];
      double[] impostos = new double[totalFunc];
      int[] escaloes = { 500, 1000, 2000, 4000, Integer.MAX_VALUE };
```

```
      int[] taxas = { 5, 10, 20, 30, 40 };
      // cálculos de imposto
      vencimentos = leSalarios(totalFunc);
      for(int v = 0; v <= totalFunc - 1; v++) {
         impostos[v] = calcula_imposto(escaloes, taxas, vencimentos[v]);
      }
      mostra_resultados(vencimentos, impostos);

   }
}
```

Ex9: Escrever um programa que simule o jogo do Euromilhões. O programa gera aleatoriamente uma chave contendo 5 números (de 1 a 50) e duas estrelas (1 a 9).
Em seguida são pedidos ao utilizador 5 números e duas estrelas (a aposta). O programa deverá em seguida apresentar a chave gerada e o número de números e estrelas certos da aposta do utilizador. Naturalmente devem ser usados arrays para guardar os dados.

```
/**
 * Jogo simulador do Euromilhões
* @author F. Mário Martins
 * @version 3/2007
 */
import static java.lang.Math.random;
import static java.lang.System.out;
public class Ex9Arrays {

   public static int[] geraNumeros(int total, int menor, int maior) {
      int[] sorteio = new int[total];
      for(int i = 1; i <= total; i++)
         sorteio[i-1] = menor + (int) (random()*maior);
      return sorteio;
   }

   public static int[] leNums(int total, int inf, int sup) {
      int[] nums = new int[total]; boolean repetido = false; int num;
      for(int i = 0; i <= total - 1; i++) {
        do {
          out.print("Numero " + (i+1) + " (entre " + inf + " e " + sup + " ): ");
          num = Input.lerInt();
          repetido = existe(nums, i+1, num);
          if (repetido) out.println("Não serve. É repetido !");
        }
        while(!(num >= inf && num <= sup) || repetido);
        nums[i] = num;
      }
      return nums;
   }

   public static boolean existe(int[] lista, int dim, int elem) {
      // verifica se elem existe na lista dada como parâmetro
      int index = 0; boolean existe = false;
      while(index <= dim-1 && !existe) {
         existe = (lista[index] == elem); index++;
      }
      return existe;
   }
   public static int comparaArrays(int[] lista1, int[] lista2) {
       // quantos elementos de lista2 existem em lista1
      boolean existe; int conta = 0;
      for(int i = 0; i <= lista2.length-1; i++) {
```

```
            existe = existe(lista1, lista1.length, lista2[i]);
            if (existe) conta++;
        }
        return conta;
    }

    public static void mostra_resultados(int numsOk, int estrelasOk) {
        out.println(" Acertou em " + numsOk + " números e "
                    + estrelasOk + " estrelas.");
        out.println("-----------------------------------------------------");
        if( numsOk == 5 && estrelasOk == 2 )
          out.println(" VOCÊ É MULTIMILIONÁRIO !!! EXCÊNTRICO !!!");
    }

    public static void main(String[] args) {
        int[] numeros = new int[5];
        int[] estrelas = new int[2];
        int[] palpiteNums = new int[5];
        int[] palpiteEstrelas = new int[2];
        String resp; int numCertos, estrelasCertas;

        do {
          numeros = geraNumeros(5, 1, 50);
          estrelas = geraNumeros(2, 1, 9);
          palpiteNums = leNums(5, 1, 50);
          palpiteEstrelas = leNums(2, 1, 9);
          //
          numCertos = comparaArrays(numeros, palpiteNums);
          out.println("--------  CHAVE DO EUROMILHÕES  -----------");
          for(int i = 0; i < 5; i++) out.print(numeros[i] + " ");
          out.print(" -- ");
          for(int i = 0; i < 2; i++) out.print(estrelas[i] + " ");
          out.println();
          out.println("-------------------------------------------");
          estrelasCertas = comparaArrays(estrelas, palpiteEstrelas);
          mostra_resultados(numCertos, estrelasCertas);
          //
          out.println("Novo Jogo ? (S/*) : ");
          resp = lnput.lerString();
        }
        while(resp.equals("S"));
        out.println("------ SEMPRE A CRIAR EXCÊNTRICOS ..... ");
    }
}
```

Ex10: Modifique o programa do exemplo 9 de modo a que no final o programa apresente o somatório de todos os vencimentos e de todos os impostos retidos aos funcionários.