AULAS 7 e 8

Objectivos

Nestas duas aulas práticas pretende-se que os alunos utilizem arrays como estrutura de dados para o armazenamento de colecções de dados.

No final da 1ª aula os alunos deverão ser capazes de definir e utilizar arrays unidimensionais. Devem saber atribuir valores aos elementos de um array, ler os valores armazenados num array e fazer a distinção entre o comprimento de um array e o nº de elementos efectivamente válidos num array, i.e., reconhecer que um array construído inicialmente com n valores poderá num dado momento encontrar-se apenas parcialmente preenchido.

Na 2ª aula os alunos devem praticar a ordenação de arrays unidimensionais e a utilização de tabelas. Neste último caso, utiliza-se uma versão simplificada das tabelas relativamente à funcionalidade disponibilizada pela linguagem Java, uma vez que nos exercícios aqui propostos todas a tabelas têm o mesmo nº de linhas e colunas.

1. Copiar, compilar e executar o programa abaixo que permite ler e imprimir a cotação em bolsa de uma empresa ao longo de 5 dias no máximo.

```
import p1.*;
class aula07_cotacoes extends P1App {
  static void main(String [] args) {
    int op, numElem;
    double [] vector;

    vector = new double [5];
    numElem = 0;

    do {
      op = menu();
      switch(op) {
        case 1:
          numElem = lerCotacoes(vector, numElem);
        case 2:
          imprimirCotacoes(vector, numElem);
          break;
        case 0:
          break;
        default:
          println(" > Opção errada!");
      }
    }while(op != 0);
    println("bye...");
  }

  static int menu() {
    int opcao;
    println();
    println("=>  Menu  <=");
    println("1. Inserir novas cotações");
    println("2. Listar cotações");
    println("0. Terminar");
    opcao = readInt(" > Opção: ");
    return opcao;
  }                                                    // continua
```

```
  static int lerCotacoes(double [] v, int n) {
    double cotacao;

    println(" > Leitura de cotações");
    do {
      cotacao = readDouble("   . Cotação no dia " + (n+1) + ": ",
0, 100);
      if(cotacao != 0) {
        v[n] = cotacao;
        n++;
      }
    } while(cotacao != 0 && n < v.length);
    return n;
  }

  static void imprimirCotacoes(double [] v, int n) {
    int i;
    println(" > Listagem das cotações");
    for(i=0; i<n; i++) {
      print("   . dia " + (i+1) + " :");
      printfd(6,2, v[i]);
      if((i+1) % 3 == 0)
        println();
    }
    println();
  }
}
```

Parte 1:

a) Testar o programa introduzindo a cotação da empresa (opção 1) nos 4 primeiros dias.

b) Tentar introduzir a cotação da empresa (opção 1) nos dois dias seguintes. Identificar no código do programa o motivo pelo qual a leitura termina após a introdução da cotação no 5º dia.

c) Neste momento o *array* deverá ter 5 cotações; usar a opção 2 do menu para confirmar, se necessário. Escolher novamente a opção 1 do menu para introduzir cotações. O que acontece?
Corrigir o problema encontrado.

d) A condição `i < n` do ciclo `for` em `imprimirCotacoes` poderia ser substituída por `i < v.length`? Justifique a resposta usando um caso de teste elucidativo.

e) Alterar o programa de forma a permitir ler e imprimir a cotação da empresa ao longo de um ano (220 dias úteis).

Parte 2: acrescentar opções ao menu e escrever subprogramas que permitam resolver os seguintes pedidos:

f) Calcular a cotação máxima e mínima das acções da empresa. Deve fazer um subprograma para o determinar o máximo e outro para o mínimo.

g) Calcular a valorização das acções da empresa. A valorização é igual à cotação actual (a última inserida pelo utilizador) menos a cotação no primeiro dia. Notar que se o valor for negativo, corresponde efectivamente a uma desvalorização.

h) Indicar qual o dia em que a variação da cotação das acções relativamente ao dia anterior foi mais elevada.

i) Antes de proceder à análise estatística das cotações das acções os analistas financeiros podem pretender realizar o alisamento do *array* de cotações de forma a reduzir os efeitos de variações fortuitas. O *array* pode ser alisado aplicando a regra:

$$\forall x_i \in \vec{v} \setminus \{x_1, x_n\}, x_i = \frac{x_{i-1} + x_i + x_{i+1}}{3}$$, em que $\vec{v}$ designa o *array* das cotações.

Construir um subprograma que permita imprimir o *array* alisado (do elemento `2` até `numElem-1`).

j) Determinar o nº máximo de dias consecutivos em que a cotação das acções foi crescente, isto é, o valor da cotação num dia é igual ou superior ao valor da cotação no dia anterior.

k) Acrescentar uma opção que permita ao utilizador indicar e guardar internamente numa variável, qual o dia da semana (2 – 2ª feira, 3 – 3ª feira, …, 6 – 6ª feira) correspondente à 1ª cotação no *array* de cotações. Esta opção só deve aparecer no menu se o *array* de cotações não estiver vazio. Nota: esta variável será utilizada nas alíneas seguintes.

l) Imprimir as cotações das acções da empresa no seguinte formato:

```
semana|   S      T      Q      Q      S
------|-----------------------------------
   1  |        12.25  12.45  12.30  12.10
   2  | 11.85  11.70  11.92  12.20  12.05
```

Notar que neste caso o subprograma deve receber como parâmetro o dia da semana correspondente à 1ª cotação processado na alínea anterior, além do *array* e do nº de cotações introduzidas pelo utilizador. No final, este subprograma deve ainda prever a possibilidade de o dia da semana referido não ter sido indicado pelo utilizador.

A listagem só deve incluir os dias úteis e deve ignorar a existência de feriados e períodos de férias.

m) Acrescentar uma opção para imprimir a cotação das acções no final de cada semana (6ª feira).

2. Escrever um programa que leia a partir do teclado um conjunto de registos de PH disponibilizados por instrumentação apropriada e apresente no ecrã a média ($\bar{x}$) e o desvio padrão (σ) dos valores lidos e o nº de valores que não pertencem ao intervalo $[\bar{x} - \sigma, \bar{x} + \sigma]$. O nº máximo de registos é 10; no entanto o utilizador poderá terminar a leitura antes, introduzindo um valor negativo ou um valor superior a 14.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \text{ e } \sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})}{n-1}}$$; tal que $x_i \in \vec{v}$.

Utilizar um subprograma para calcular cada um dos valores pretendidos.

3. Escrever um programa para registar as temperaturas observadas diariamente às 23h00, num depósito de combustível de uma refinaria de petróleo, durante a primeira quinzena do mês. O programa deve permitir introduzir registos, indicando o dia do mês e a temperatura observada (opção 1), e mostrar todos os registos (opção 2). As temperaturas variam entre]0 e 40ºC[. Utilizar um subprograma para a leitura e outro para a impressão dos valores O programa deve funcionar da seguinte forma:

```
[Menu]
Opção? 1
 > Dia: 5
 > Temperatura: 24.2
[Menu]
Opcao? 1
 > Dia: 3
 > Temperatura: 20.9
[Menu]
Opção? 1
 > Dia: 35
 Erro! Dia incorrecto
[Menu]
Opção? 1
 > Dia:  5
 > registo anterior: 24.2
 > Temperatura: 22.1
[Menu]
Opção? 2
Dia  3 - 20.9ºC
Dia  5 - 22.1ºC
...
```

No exemplo à esquerda substituir `[Menu]` por:

```
1. Registar temperatura
2. Listar registos
0. Terminar
```

4. Escrever um programa para correcção de testes de resposta múltipla. Cada teste tem 10 questões com 5 respostas possíveis (a, b, c, d ou e). Cada resposta correcta conta um valor; cada resposta errada desconta `0,25` valores. Eventuais classificações negativas devem ser transformadas em zero.

   a) O programa deve começar por ler e armazenar a chave de correcção num *array*. Em seguida, deve permitir introduzir as respostas dos alunos, prevendo a possibilidade de o aluno não ter respondido a algumas questões. Depois de introduzidas as 10 respostas do aluno o programa deve apresentar a classificação final obtida e permitir ao utilizador continuar, inserindo as respostas de um novo aluno ou terminar a execução do programa.

   b) Alterar o programa anterior para que o programa passe a apresentar no final um resumo com a seguinte informação:

      i. Nº total de alunos processados

      ii. Nº total de alunos aprovados (classificação igual ou superior a `9,5`)

      iii. A média das notas de todos os alunos

iv. O nº de alunos nas seguintes condições: RPNM se a classificação for inferior a 5 valores; senão, reprovado com acesso ao exame final se a classificação for inferior a 9,5 valores; senão, aprovado se a classificação for inferior a 16 valores; e aprovado com distinção nos outros casos.

v. Um histograma das classificações obtidas com o formato abaixo. A coluna à esquerda lista as notas de 0 a 20, e a coluna à direita tem uma estrela por cada aluno que obteve a classificação indicada na linha.

```
nota | nº alunos
-----------------------------------------------------
  0  | *
 ...
 10  | ************
 ...
 20  | *****
```

5. Escrever um programa que lê a partir do teclado uma sequência de valores reais positivos até o utilizador introduzir o valor zero. A sequência pode ter no máximo 20 valores. No final o programa deve imprimir os valores lidos ordenados por ordem crescente.

6. Construir um programa didáctico para ensinar crianças a ordenar uma sequência de 8 números inteiros entre 1 e 20. O programa deve começar por gerar e armazenar a sequência de números num vector. Em seguida, deve listar o vector mostrando o nº armazenado em cada posição do vector. A criança deve indicar um par de valores a trocar de posição. Em seguida, o programa deve listar o vector resultante e a operação repete-se até o vector estar ordenado por ordem crescente. No final deve mostrar o nº de tentativas realizadas.

   O exemplo abaixo mostra como gerar 2 números aleatórios inteiros entre 1 e 10.

```
import p1.*;
import java.util.*;
class random extends P1App {
  static void main(String [] args) {
      int n1, n2;
      Random r = new Random(); // inicializar a sequência aleatória
      n1 = r.nextInt(10);      // gerar o um valor
      println(n1);
      n2 = r.nextInt(10);      // gerar outro valor
      println(n2);
  }
}
```

   Alterar o programa anterior construindo um menu com 3 níveis de dificuldade: 1 – fácil (a sequência terá apenas 5 números entre 1 e 20), 2 – médio (a sequência terá 8 números entre 1 e 20) e 3 – difícil (a sequência terá 15 números entre 1 e 50). O menu deve incluir ainda uma opção 0 – sair.

7. Escrever um programa que permita gerar chaves aleatórias para o preenchimento de boletins do totoloto. O programa deve perguntar qual o nº de cruzes que o apostador pretende (6 a 12 cruzes) e apresentar uma chave ordenada sem números repetidos. O programa termina quando utilizador indicar que pretende 0 cruzes.

   (Nota: para gerar números aleatórios consultar o exemplo do exercício nº 6)

8. Construir um programa que permita ler e armazenar numa tabela o nº de unidades produzidas com defeito em 4 linhas de montagem diferentes para cada dia da semana, e executar os pedidos indicados nas alíneas abaixo. O programa deve incluir um menu com uma opção para cada alínea proposta.

	Segunda	Terça	Quarta	Quinta	Sexta
Linha #1	21	1	7	5	12
Linha #2	12	2	33	6	8
Linha #3	7	1	2	7	11
Linha #4	9	0	44	5	13

   a) Mostrar a tabela.

   b) Mostrar o nº total de peças defeituosas produzidas por linha de montagem.

   c) Mostrar o nº total de peças defeituosas produzidas por dia da semana.

   d) Mostrar o n.º médio de peças defeituosas produzidas por dia da semana.

   e) Mostrar o n.º médio de peças defeituosas produzidas por linha de montagem.

   f) Indicar qual a linha de montagem que produz mais peças defeituosas.

9. Uma empresa vai realizar um inquérito telefónico para tentar perceber o perfil dos consumidores de um de terminado tipo de produtos. O inquérito tem 8 questões com 4 hipóteses de resposta. Cada uma destas respostas tem um peso específico (valor entre 0 e 1), tal como exemplificado na tabela abaixo.

Questão	**a**	**b**	**c**	**d**
1	0,5	0,7	0,2	0,6
2	0,1	0,8	0,1	0,9
…	…	…	…	…

   O programa deve permitir introduzir para cada um dos inquiridos a resposta dada a cada questão e imprimir no final o somatório dos pesos correspondentes às respostas dadas. Se o somatório for inferior a 3 o inquirido terá preferência por produtos do tipo A; para valores entre 3 e 5 inclusive, os inquiridos não mostram apetência especial por nenhum dos produtos; para valores superiores a 5 os inquiridos terão apetência por produtos do tipo B.

   Depois de introduzir todas as respostas de um inquirido, o utilizador do programa deve poder optar por continuar ou terminar a execução. No final, o programa deve imprimir um resumo das respostas dos utilizadores: nº de utilizadores que têm preferência por produtos do tipo A, Tipo B e o nº de casos indecisos.

10. Pretende-se escrever um programa para apoiar os seguranças responsáveis pelo controlo dos lugares de estacionamento para camiões disponíveis na área de cargas e descargas de uma fábrica. A área de estacionamento é uma área rectangular com 80 lugares, distribuídos em 8 filas com 10 lugares cada As filas estão identificadas de 1 a 8 e os lugares de 1 a 10. Inicialmente todos os lugares devem ser assinalados como disponíveis usando uma tabela com duas dimensões. O programa deve permitir:

    a) Visualizar o mapa de ocupação dos lugares, identificando os lugares livres e os lugares ocupados. Usar um subprograma que recebe como parâmetro a tabela definida inicialmente.

	Lug 1	Lug 2	Lug 3	Lug 4	Lug 5	Lug 6	Lug 7	Lug 8	Lug 9	Lug10
fila 1	O	O	L	L	L	L	L	L	L	L
fila 2	O	L	L	L	L	L	L	L	L	L
…	…	…	…	…	…	…	…	…	…	…

    b) Assinalar a chegada de um camião. O segurança deve escolher um lugar livre, indicando o nº da fila e o nº do lugar atribuído, e esse lugar deve passar a ocupado (O). Usar um subprograma para a marcação do lugar como ocupado que recebe como parâmetros a tabela, o nº da fila e o nº do lugar a marcar. O subprograma deve validar se o lugar está realmente livre antes de fazer a marcação. Se sim, deve devolver o valor lógico `true`, indicando que a marcação foi realizada com sucesso. Se a marcação não tiver sucesso, deve devolver `false`, de forma a indicar que deve ser escolhido outro lugar.

    c) Assinalar a saída de um camião. O segurança deve indicar o nº da fila e o nº do lugar onde o camião se encontrava. Usar um subprograma equivalente ao da alínea anterior para assinalar o lugar como livre (L).

    d) Indicar o nº total de lugares livres. Usar um subprograma para o efeito.