Ficha de Exercícios

Exercícios de introdução

1. Escreva uma aplicação que gere um inteiro aleatório entre 0 e 100, inclusive, e que permita ao utilizador adivinhar o número gerado, através de indicação sucessiva de valores.

2. Escreva uma aplicação que peça ao utilizador para pensar num número entre 1 e 100, inclusive. Através de várias perguntas ao utilizador, o qual responderá 'ACERTOU', 'É MAIOR' ou 'É MENOR', a aplicação tentará adivinhar o número. No final deve ser apresentado o número de tentativas que foram necessárias para adivinhar.

3. Escreva uma aplicação de consola que crie um *array* de inteiros, perguntando ao utilizador a dimensão do *array* e quais os valores dos seus elementos. Com base no *array* criado desenvolva funções *static* para...
   a. Imprimir o *array*.
   b. Calcular e imprimir o maior e o menor elemento presentes no *array*.
   c. Somar todos os elementos do *array*, retornando a soma.
   d. Calcular e imprimir a média dos valores do *array*.
   e. Inverter todos os valores presentes no *array*, $[i_0,i_1,...,i_{n-1}] \Rightarrow [i_{n-1},...,i_1,i_0]$

4. Defina uma classe que contenha, como membro, um *array* de inteiros com 20 elementos. Os valores a incluir neste *array* são inteiros aleatórios entre 0 e 100, inclusive. Um inteiro gerado aleatoriamente apenas deve ser colocado no *array* caso esse valor ainda não esteja armazenado, isto é, o *array* não deve conter valores repetidos. Usar uma função membro booleana (que devolve um valor booleano) para verificar se um determinado valor existe no *array*. Para cada valor gerado, deverá ser chamada a função para verificar se o mesmo já existe no *array*. Se ele existir, o valor deve ser descartado e deve ser gerado um novo valor.
   A classe deve ter funções para:
   a. Listar os 20 valores do *array*.
   b. Mostrar a quantidade de valores que foram gerados em duplicado, ou seja, a diferença entre o total de valores gerados e o número de valores aproveitados (no caso concreto serão 20).

5. Defina uma classe que represente uma aposta no Totoloto. Uma aposta é definida por cinco números inteiros compreendidos entre 1 e 49, todos diferentes entre si e não necessariamente introduzidos de forma crescente, e um número da sorte compreendido entre 1 e 13 (pode ser igual a um dos 5 números indicados anteriormente). A classe deve permitir:
   a. Preencher um número de cada vez com vista à construção da aposta.
   b. Verificar se a aposta está completa.
   c. Preencher automaticamente uma aposta completa.
   d. Comparar a aposta corrente com uma aposta ganhadora.

6. Escreva uma aplicação que calcule e imprima a transposta de uma matriz. A matriz deverá ser representada através de classe adequada e deve possuir métodos adequados para as funcionalidades referidas.

7. Escreva uma aplicação que some matrizes retangulares. Uma matriz deverá ser representada através de uma classe adequada. A soma deverá ser realizada através de duas formas distintas:

a. Função membro que acumulará aos valores de uma matriz os valores de outra matriz.
b. Função estática que recebe duas matrizes e retornará uma nova matriz com o resultado da soma, ou seja, não se pretende que sejam alteradas as matrizes originais.

8. Defina uma classe que contenha, como um membro, uma matriz de *m x n* elementos. Esta classe deve ter funções para alterar os elementos da matriz, calcular a soma de cada linha, calcular a soma de cada coluna, calcular a soma de todos os seus elementos e imprimir toda a informação.
Exemplo de output para uma matriz definida programaticamente:

```
Matriz:
1    0    2    -1   3
4    3    2    1    0
1    -2   3    4    5
8    5    1    3    2

Soma das linhas: 5, 10, 11, 19
Soma das colunas: 14, 6, 8, 7, 10
Soma total: 45
```

9. Escreva uma aplicação que calcule e imprima o triângulo de Pascal. O triângulo de Pascal deverá ser representado através de uma classe própria, que deverá incluir funções para: gerar triângulo com uma determinada profundidade, gerar uma *String* representativa do triângulo (`toString()`) e mostrar o triângulo na consola. Quando o objeto for criado poderá ser indicado através do *construtor* a profundidade pretendida.

Estruturação de programas

10. Escreva uma aplicação que implemente o jogo do enforcado. Na implementação do jogo deverá garantir uma separação entre o código referente a interação com o utilizador e as classes necessárias para realizar a sua gestão. O código deverá ser estruturado nas seguintes classes:

- `JogoEnforcado`, que inclui (apenas) a função `main`;
- `JogoEnforcadoDicionario`, que permite disponibilizar as palavras a usar no jogo. Deverá possuir apenas membros estáticos para armazenar uma tabela com as palavras, e métodos para acesso à quantidade de palavras armazenadas e a uma palavra (através do seu índice);
- `JogoEnforcadoModelo`, que efetuará a gestão de todo o jogo (sem qualquer código de interação com o utilizador). Deverá conter métodos que permitem iniciar jogo (sorteando uma nova palavra), tentar/verificar uma letra, verificar fim de jogo, gerir informação de evolução de jogo: número de tentativas, letras já tentadas, ...;
- `JogoEnforcadoIU`, onde será implementada toda a interação com o utilizador.

Exemplo da classe `JogoEnforcado`:

```
public class JogoEnforcado {
    public static void main(String args[]) {
        JogoEnforcadoModelo jogo = new JogoEnforcadoModelo();
        JogoEnforcadoIU jogoIU = new JogoEnforcadoIU(jogo);
        jogoIU.jogar();
    }
}
```