BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

ASSUNTO - Descrição de Algoritmos estruturando-os em módulos e codificação em Java

OBJETIVOS GERAIS:

- Implementar programas estruturados em módulos

OBJETIVOS ESPECÍFICOS:

- Conceber e descrever em pseudocódigo algoritmos, sempre que adequado estruturados em módulos
- Implementar em Java programas estruturados em módulos

TAREFAS DA SEMANA:

Exercício 1 (**)

a) Analise os seguintes módulos (em Java designados métodos) e identifique as respetivas funcionalidades.

```
public static boolean metodo1 (String pal) {
        boolean resposta = true;
        pal = pal.toLowerCase();
        int tamanho = pal.length();
        for (int i=0;  i< tamanho /2 ;  i++) {
                if (pal.charAt(i) != pal.charAt(tamanho - 1 - i)) {
                        resposta = false; break;
                }
        }
        return resposta;
}
```

```
public static boolean metodo2 (String pal) {
        int i, j;
        pal = pal.toLowerCase();
        i = 0; j = pal.length()-1;
        while (i<j && pal.charAt(i) == pal.charAt(j)) {
                i++;
                j--;
        }
        return i>=j;
}
```

BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

b) Faça um programa que leia uma sequência de palavras até encontrar um palíndromo (palavra cuja leitura da esquerda para a direita é igual à da direita para a esquerda). O programa deve contar o número de palavras lidas que antecedem o palíndromo.

OBS: Utilize o método anterior que achar apropriado.

Exercício 2 (**)

Faça um programa que represente sob a forma de gráficos de barras, o número de positivas e negativas dos alunos de uma turma a um conjunto de disciplinas. O programa deverá começar por pedir o nº de alunos da turma e o nº de disciplinas e para cada disciplina pedirá o nome da disciplina e o nº de alunos aprovados.
Deve existir um módulo para imprimir a informação de uma disciplina.

O output produzido deverá ter o seguinte aspeto:

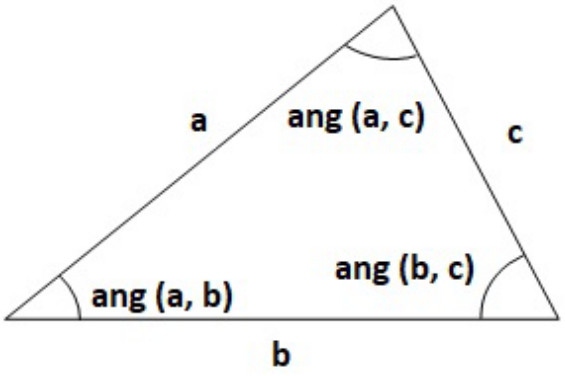
```
Disciplina: Português
- Positivas: ****************
- Negativas: ****
Disciplina: Matemática
- Positivas: ***********
- Negativas: *********
```

Exercício 3 (**)

a) Faça um método (**calcAng**) que calcule um ângulo interno de um triângulo (em graus), sendo dadas as medidas dos três lados desse triângulo.

b) Pedidas as medidas de três lados, verifique se elas são válidas e se é possível formar um triângulo. Em caso afirmativo calcule todos os ângulos internos desse triângulo. Para isso chame três vezes o método desenvolvido na alínea anterior.

Nota: A solução deve ser desenvolvida exclusivamente em modo texto. Nos resultados deverão ser apresentados **apenas** os valores dos ângulos arredondados às unidades, sem qualquer texto ou outra informação. No caso de o triângulo não ser possível, a mensagem deverá ser: **impossivel**



Ângulo	Fórmula
ang (a,b)	$arc\cos\left(\frac{a^2+b^2-c^2}{2ab}\right)$
ang (a,c)	$arc\cos\left(\frac{a^2+c^2-b^2}{2ac}\right)$
ang (b,c)	$arc\cos\left(\frac{b^2+c^2-a^2}{2bc}\right)$

BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

Exercício 4 (**)

Analise a classe CalCombinatorio definida nas aulas teóricas.
Crie um projeto onde vai incluir esta classe CalCombinatorio.
Acrescente à classe CalCombinatorio mais dois métodos de classe, o método arranjos e o método permutações.
Crie uma classe TesteCalCombinatorio para testar as funcionalidades da classe.

Exercício 5 (**)

a) Faça um módulo que dados dois números inteiros positivos retorne a quantidade de dígitos comuns nas mesmas posições.

b) Elabore um programa que leia N pares de valores inteiros positivos, sendo N introduzido pelo utilizador e validado. Após a leitura dos N pares de valores o programa deve apresentar o par que tiver mais dígitos comuns nas mesmas posições. No caso de haver mais do que um par com a mesma quantidade de dígitos em comum, deve ser apresentado o último par encontrado.

Nota: A solução deve ser desenvolvida numa única classe e exclusivamente em modo texto. O resultado deve conter apenas o par em causa no formato numero1/numero2. No caso de não haver nenhum par de números que tenha algarismos em comum deve ser apresentada a mensagem: **sem resultados**

Exercício 6 (*)

Faça um programa que permita determinar volumes de sólidos de revolução (cilindros, cones e esferas). Para cada sólido será introduzido o tipo de sólido e as respetivas dimensões. O programa termina quando o tipo de sólido for a palavra "FIM". Implemente o programa de forma modular.

OBS: $V_{esfera} = 4/3\ \pi\ R^3$
$V_{cilindro}$ = Área Base x Altura = $\pi\ R^2$ Altura
$V_{cone} = 1/3\ \pi\ R^2$ Altura

Exercício 7 (***)

a) Faça um módulo que verifique se um número é ou não um número octal.

b) Faça um módulo que converta um número octal em número decimal.

c) Faça um programa que leia uma sequência de números na base octal e os converta em números decimais. A sequência termina quando for introduzido um número que não é octal.

Exercício 8 (***)

a) Faça um módulo que converta um número decimal em um número hexadecimal.
b) Faça um programa que leia uma sequência de números do sistema decimal terminada por zero e os converta em números hexadecimais.

BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

Exercício 9 (**)

Na sucessão de Fibonacci, o primeiro termo é zero, o segundo termo é um e qualquer um dos outros termos é igual à soma dos dois anteriores. Elabore um programa em Java para mostrar os primeiros N termos desta sucessão onde N é definido pelo utilizador.

Exercícios Complementares

Exercício 1 (**)

Faça um programa modular que, dada uma sequência de números, imprima os que são números de ArmstrongP.

Um número de ArmstrongP é um número em que a soma dos cubos dos seus algarismos é igual ao próprio número. Exemplo: $407=4^3+0^3+7^3$.

Exercício 2 (**)

a) Faça um módulo que verifique se um número é ou não um número capicua.

b) Faça um programa que leia uma sequência de números inteiros e termine quando for introduzido um número capicua ou quando tiver analisado 100 números sem o encontrar. O programa deve escrever uma mensagem adequada.

Exercício 3(**)

a) Faça um módulo que calcule e retorne a soma de todos os divisores pares de um número dado como parâmetro. No entanto, não deve considerar o próprio número como divisor.

b) Elabore um programa que, dada uma sequência de números positivos, determine e apresente a percentagem de números cuja soma dos seus divisores pares é a maior.

Exercício 4 (***)

a) Faça um método que receba os seguintes parâmetros:
   - Um número inteiro, sob a forma de texto, numa determinada base;
   - Base (de 2 a 16) na qual está especificado o número;
   - Base (de 2 a 16) para a qual vai ser convertido o número.

   O método deverá verificar a validade de todos os parâmetros (incluindo o primeiro). Se estas validações forem bem sucedidas, o método deverá retornar, sob a forma de string, o número convertido para a base especificada para esse efeito. Se algum dos parâmetros for inválido, o método deverá retornar uma string vazia. Os símbolos que podem surgir na representação de um número são os algarismos (de 0 a 9) ou letras do alfabeto (de A a F).

b) Faça um programa que teste o método anterior.

BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

Exercício 5 (**)

Faça um programa que dadas as dimensões de um paralelepípedo (dimensões da base e da altura) determine: área das bases, área lateral, área total e volume.

Este programa deve ser dirigido por um menu.

Sugere-se a implementação das seguintes funcionalidades:

- menu
- validarDadosEntrada

Exercício 6 (*)

Corrija os erros do seguinte programa em Java.

```
public class Main {

    public static void main(String[] args) {

        int numero, alg, i, j;

        resp = JOptionPane.showInputDialog("Qual o número?");
        numero = Integer.parseInt(resp);

        do {
            j = j + 1;
            alg = numero % 10;
            if (alg%2 = 0)
                i = i + 1;
            numero = numero/10;
        }while (numero>0);

        perc = (float) i / j  + 0.5f;
        JOptionPane.showMessageDialog("Valor=" + (int) perc);
}
```

Exercício7 (***)

Na sequência 6788, 2688, 768, 336, 54, 20, 0, cada termo é o produto dos dígitos do número anterior:

6 * 7 * 8 * 8 = 2688
2 * 6 * 8 * 8 = 768

Para um dado número inicial, o número de passos até que se atinja um número com um único dígito (não necessariamente zero) é designado por "persistência" desse número (no exemplo acima é 6).

Escreva um programa em Java para calcular a persistência de um número dado via teclado.

BLOCO 2 – Implementação de algoritmos em Java e decomposição modular – PL6

Exercício 8 (**)

Faça um programa que determine quantos pontos (X,Y) introduzidos pelo utilizador estão dentro de um círculo. A introdução de pontos termina quando for introduzido um ponto igual ao centro. O programa deverá ler os seguintes dados do teclado:

- Coordenadas do centro do círculo;
- Raio do círculo;
- Pontos a serem testados.

O programa deverá utilizar um módulo que calcule a distância entre dois pontos. Desenvolva esse módulo atendendo a que para calcular a distância entre os dois pontos $A(x_a, y_a)$ e $B(x_b, y_b)$ usamos a expressão:

$$d_{AB}{}^2 = (x_b - x_a)^2 + (y_b - y_a)^2$$

Exercício 9 (*)

a) Faça um módulo que verifique se um número é ou não um número perfeito.

b) Faça um programa que leia uma sequência de números inteiros e termine quando for introduzido um número perfeito dizendo qual a sua posição na sequência.

Um número é perfeito quando é igual à soma de todos os seus divisores excluindo o próprio número. Por exemplo, 6 é perfeito porque 1+2+3 = 6.