

Java

(Modularização)

Modularização

Técnica de programação que consiste em decompor a solução de um problema em blocos, chamados módulos, que interagem com um bloco principal.

[Xavier, G.F.C. Lógica de programação. São Paulo: Editora SENAC, 1998.]

Módulos:

“Conjunto de comandos que constitui uma parte de um algoritmo principal e que tem uma tarefa bem-definida e independente em relação ao resto do algoritmo.”

[Xavier, G.F.C. Lógica de programação. São Paulo: Editora SENAC, 1998.]

Estrutura do programa

```
package olaMundo;
```

```
public class OlaMundo {
```

```
    public static void main(String[] args) {  
        mensagem();  
    }
```

```
    public static void mensagem() {  
        System.out.println("Olá, mundo");  
    }
```

```
}
```

Características desejadas

- Módulos devem ter finalidade bem definida
- Independência do módulo em relação ao resto do programa
- Implementação em época diferente do restante do programa
- Manutenção mais simples, evitando efeitos colaterais
- Reuso de módulos em sistemas diferentes exigindo pouca ou nenhuma adaptação

[Xavier, G.F.C. Lógica de programação. São Paulo: Editora SENAC, 1998.]

Tipos de Módulo

- **Módulo principal** → Todo programa deve ter um módulo principal a partir do qual o programa começa a ser executado. Os demais módulos somente serão executados se forem “chamados” a partir da função principal. Em Java temos a função **main**.
- **Procedimento** → módulo que não retorna valor. No exemplo que vimos temos `mensagem()` que realiza a ação de mostrar algo na tela mas não retorna valor a ser aproveitado pelo módulo que o chamou.
- **Função** → módulo que retorna valor no nome da função. Este valor deve ser aproveitado pelo módulo que chama a função através de uma atribuição por exemplo. Em Java temos por exemplo a função `Math.sqrt(4)` que retorna o valor da raiz quadrada do valor entre parênteses.

Implementação de módulos em Java

```
public static <tipo> <identificador> (<parâmetros>){  
    <Bloco de instruções>  
}
```

- Exemplo:

```
public static void mensagem() {  
    System.out.println("Olá, mundo");  
}
```

Exemplo

Problema:

Faça um programa que receba um valor inteiro n e mostre na tela o fatorial de todos os números de 1 até n .

Exemplo – solução não modularizada

```
public static void main(String[] args) {  
    int n, fat, x, y;  
    Scanner scan = new Scanner(System.in);  
    n = scan.nextInt();  
    for(x=1; x<=n; x++) {  
        fat = 1;  
        for(y=x; y>=1; y--) {  
            fat = fat * y;  
        }  
        System.out.println(x + "! = " + fat);  
    }  
}
```


Exemplo – solução modularizada

```
public class Fatorial_modularizado {  
  
    public static void main(String[] args) {  
        int n, fat, x;  
        Scanner scan = new Scanner(System.in);  
        n = scan.nextInt();  
        for(x=1; x<=n; x++) {  
            fat = fatorial(x);  
            System.out.println(x + "! = " + fat);  
        }  
    }  
  
    public static int fatorial(int x) {  
        int y, fat = 1;  
        for(y=x; y>=1; y--) {  
            fat = fat * y;  
        }  
        return fat;  
    }  
}
```

Exemplo – Olá, mundo

```
package olaMundo;

public class OlaMundo {

    public static void main(String[] args) {
        mensagem();
    }

    public static void mensagem() {
        System.out.println("Olá, mundo");
    }

}
```

Exemplo – Olá, mundo

E se quisermos flexibilizar para direcionar o olá para alguém?

Precisamos passar o nome por parâmetro.

Exemplo – Olá, mundo

```
package olaMundo;
```

```
public class OlaMundo {
```

```
    public static void main(String[] args) {  
        mensagem("Asdrubal");  
    }
```

```
    public static void mensagem(String nome) {  
        System.out.println("Olá, " + nome + "!");  
    }
```

```
}
```

Exemplo – soma

```
public static void main(String[] args) {  
    int x = 2;  
    soma(x, 5);  
}
```

```
public static void soma(int a, int b) {  
    int c;  
    c = a + b;  
    System.out.println(c);  
}
```

Exercício 1

Faça um programa que receba a quantidade de alunos de uma turma e, em seguida, para cada aluno, receba o nome e as quatro notas do aluno, calcule e mostre a média do aluno e mostre na tela uma mensagem informando o resultado do aluno. Cada aluno receberá uma das seguintes mensagens:

- a) Parabéns, nome_do_aluno, você foi aprovado. (média mínima 7.0)
- b) Nome_do_aluno, apresente-se para a prova final. (média entre 4 e 6.9)
- c) Nome_do_aluno, infelizmente você não atingiu a média mínima. (média abaixo de 4.0)

Para implementar este programa faça uma função que receba por parâmetro o nome do aluno e a média e mostre na tela a mensagem com o resultado.

Exercício 2

Faça um algoritmo em Java que receba a idade de uma pessoa e informe em qual das categorias se enquadra em relação à obrigatoriedade do voto:

- Sem permissão para votar (0 a 15 anos)
- Voto facultativo (16 a 17 anos, acima de 65 anos)
- Voto obrigatório (18 a 64 anos)

O programa deve ter uma função que recebe a idade e mostra na tela uma das mensagens.

Exercício 3

A confederação brasileira de natação irá promover eliminatórias para o próximo mundial. Fazer um programa que receba a idade de um nadador e determine (imprima) a sua categoria segundo a tabela a seguir:

Categoria	Idade
Infantil A	5–7 anos
Infantil B	8–10 anos
Juvenil A	11–13 anos
Juvenil B	14–17 anos
Sênior	Maiores de 18 anos

O programa deve ter uma função que receba por parâmetro a idade e mostre na tela a classificação.

Exercício 4

Faça um programa em Java que receba três valores inteiros referentes ao tamanho dos três lados de um triângulo. Para formar um triângulo cada lado deve ser menor do que a soma dos outros dois lados. Uma vez verificado que os valores estão corretos (ou seja, formam um triângulo) o programa deve classifica-lo em equilátero (todos os lados iguais), isósceles (dois lados iguais), escaleno (todos os lados diferentes).

O programa deve ter uma função que receba o tamanho dos três lados e mostre na tela a classificação do triângulo.

Exemplo – soma

```
public static void main(String[] args) {  
    int x = 2;  
    soma(x, 5);  
}
```

```
public static void soma(int a, int b) {  
    int c;  
    c = a + b;  
    System.out.println(c);  
}
```

E se quisermos devolver o valor da soma para quem chamou a função?

Exemplo – soma

```
public static void main(String[] args) {  
    int x = 2, y = 3, s;  
    s = soma(x, y);  
    System.out.println(s);  
}
```

```
public static int soma(int a, int b) {  
    int c;  
    c = a + b;  
    return c;  
}
```

Exercício 5

Faça um programa em Java que receba três valores inteiros referentes ao tamanho dos três lados de um triângulo. Para formar um triângulo cada lado deve ser menor do que a soma dos outros dois lados. Uma vez verificado que os valores estão corretos (ou seja, formam um triângulo) o programa deve classifica-lo em equilátero (todos os lados iguais), isósceles (dois lados iguais), escaleno (todos os lados diferentes).

O programa deve ter duas funções:

- Uma que receba o tamanho dos três lados e devolva true se for triângulo ou false se não for triângulo.
- Outra que receba o tamanho dos três lados e devolva a classificação