


A detailed botanical illustration background featuring various plants and flowers. In the top left, there are pink and purple flowers with green leaves. In the top right, there are white flowers with green leaves. In the bottom left, there is a large yellow flower with a dark red center and green leaves. In the bottom right, there is a yellow lemon with green leaves and a small white flower. The central text is framed by a white border.

Modularização

express



*Aquilo que escuto eu esqueço,
Aquilo que vejo eu lembro,
Aquilo que faço eu aprendo.*

Confúcio

”



1. Módulos

Uma palavrinha



Módulos

- É a base da **Programação Estruturada**: divisão de um problema em **partes**.
- A cada uma dessas partes é dado o nome de **módulo**.
- **Modularização** é uma **técnica** para desenvolver programas, por meio de **refinamentos sucessivos**.
- Redução de um problema a um conjunto de tarefas bem específicas, destinadas a solucioná-lo de maneira eficiente
- Vantagens: reuso, portabilidade, facilidade para manutenção, organização
- Podem ser procedimentos ou funções
- Em Java não tem distinção. São métodos.



Procedimentos

- São módulos ou sub-rotinas que são chamados a partir de um outro programa (normalmente o programa principal).
- Depois de serem executadas, o fluxo de execução volta para o programa chamador.
- Não retorna nenhum resultado para o programa que o chamou
- Exemplos:
 - Gravar dados em arquivos
 - Enviar dados para o dispositivo de saída
 - Ordenar que o computador desligue



```
1 public class Procedimento {
2     public static void Cabecalho () {
3         System.out.println (" +-----+ " );
4         System.out.println (" |   ***   *   *   ***   | " );
5         System.out.println (" |   *   *   *   *   *   | " );
6         System.out.println (" |   *   *   *   *   *   | " );
7         System.out.println (" |   ***   *   *   *   | " );
8         System.out.println (" |   *       *   *   *   | " );
9         System.out.println (" |   *       *   *   *   | " );
10        System.out.println (" |   *       *   *   *   | " );
11        System.out.println (" |   *       ***   ***   | " );
12        System.out.println (" +-----+ " );
13    }
14    public static void main (String args[]) {
15        Cabecalho ();
16    }
17 }
```

Funções

- São módulos ou sub-rotinas que são chamados a partir de um outro programa (normalmente o programa principal)
- Depois de serem executadas, o fluxo de execução volta para o programa chamador.
- Sempre retorna um resultado para o programa que o chamou, devendo portanto ser “consumido” pelo chamador. Ou armazenando o resultado na memória, ou transferindo para o dispositivo de saída.
- Exemplos:
 - Mostrar um menu e retornar a opção
 - Solicitar uma entrada ao usuário
 - Efetuar uma operação matemática

```
1 import java.util.Scanner;
2 public class Funcao {
3     public static int Menu() {
4         Scanner dado = new Scanner(System.in);
5         System.out.println("+=====+");
6         System.out.println("|   P U C   - A D S   |");
7         System.out.println("+=====+");
8         System.out.println("|  1 - Alunos      |");
9         System.out.println("|  2 - Professores |");
10        System.out.println("|  3 - Funcionarios |");
11        System.out.println("|  4 - Terceirizados |");
12        System.out.println("|  0 - SAIR        |");
13        System.out.println("+=====+");
14        System.out.print("    opção --> ");
15        return dado.nextInt();
16    }
17    public static void main (String args[]) {
18        int opcao;
19        opcao = Menu();
20        System.out.println("\n\nA opção escolhida foi " + opcao);
21    }
22 }
```




Argumentos ou Parâmetros

- Podem ser utilizados tanto em procedimentos quanto em funções
- Servem para informar valores de dados que serão necessários para executar a tarefa solicitada
- Exemplo: `System.out.println("Olá Mundo!")`

Chamada do método	Parâmetro	Fechamento da chamada
<code>System.out.println(</code>	<code>"Olá mundo!"</code>	<code>);</code>



Passagem de parâmetros

- É o mecanismo de informar sobre quais valores o processamento definido na função deve ser realizado
- Os parâmetros são passados para um método de acordo com a sua posição
- A cada chamada de método, os parâmetros são copiados e estas cópias são utilizadas na execução do mesmo



exemplo

- ⦿ Calcular descontos percentual em valores de produtos
- ⦿ Precisa de quais dados?
 - ⦿ Valor atual do produto – double
 - ⦿ Percentual do desconto – double
- ⦿ Resulta em algum novo dado?
 - ⦿ Valor com desconto - double



```
1 import java.util.Scanner;
2 public class Parametros {
3     public static double CalcDesc(double valor, double perc){
4         double desconto, valorDesc;
5         desconto = valor * (perc / 100);
6         valorDesc = valor - desconto;
7         return valorDesc;
8     }
9     public static void main(String[] args) {
10        Scanner dado = new Scanner(System.in);
11        double valor, perc, novoValor;
12        System.out.print("Digite valor do produto: ");
13        valor = dado.nextDouble();
14        System.out.print("Digite o percentual de desconto: ");
15        perc = dado.nextDouble();
16        novoValor = CalcDesc(valor, perc);
17        System.out.println("\n\nNovo valor do produto: " + novoValor);
18    }
19 }
```





2. Métodos



orientações

Por questões de reaproveitamento do código, evite efetuar entrada/saída de dados dentro dos métodos:

- ⦿ Envie os dados necessários através de parâmetros;
- ⦿ Receba resultados pelo retorno
- ⦿ Por questões de reaproveitamento do código, evite efetuar entrada/saída de dados dentro dos métodos:
- ⦿ Lembre que ao terminar a execução de um método, o seu programa irá continuar a partir do ponto onde havia efetuado a chamada

Sintaxe geral

```
//sem parâmetros sem retorno
void NomeMetodo () {
    <corpo da função>
}

//com parâmetros sem retorno
void NomeMetodo (<tipo> arg1, ..., <tipo> argN) {
    <corpo da função>
}

//sem parâmetros com retorno
<tipo de retorno> NomeMetodo () {
    <corpo da função>
    return valorDeRetorno;
}

//com parâmetros com retorno
<tipo de retorno> NomeMetodo (<tipo> arg1, ..., <tipo> argN) {
    <corpo da função>
    return valorDeRetorno;
}
```



3. Exercícios

Exemplos



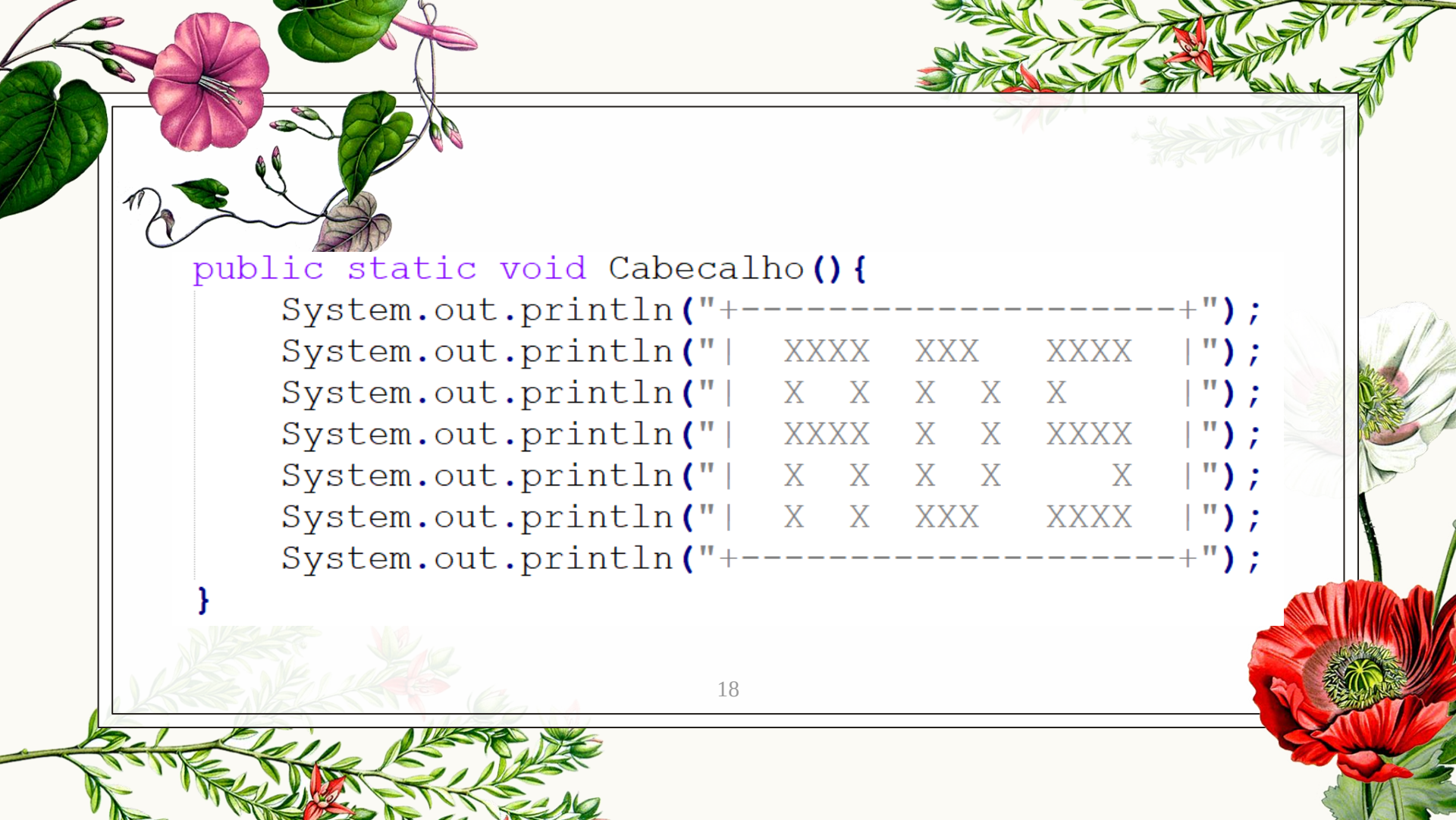


sem Parâmetros sem Retorno

1. Crie um método com o nome **Cabecalho** que mostre o seguinte layout na tela:

```
+-----+
|      |      |      |
| XXXX  XXX  XXXX  |
| X  X  X  X  X      |
| XXXX  X  X  XXXX  |
| X  X  X  X      X  |
| X  X  XXX  XXXX  |
+-----+
```





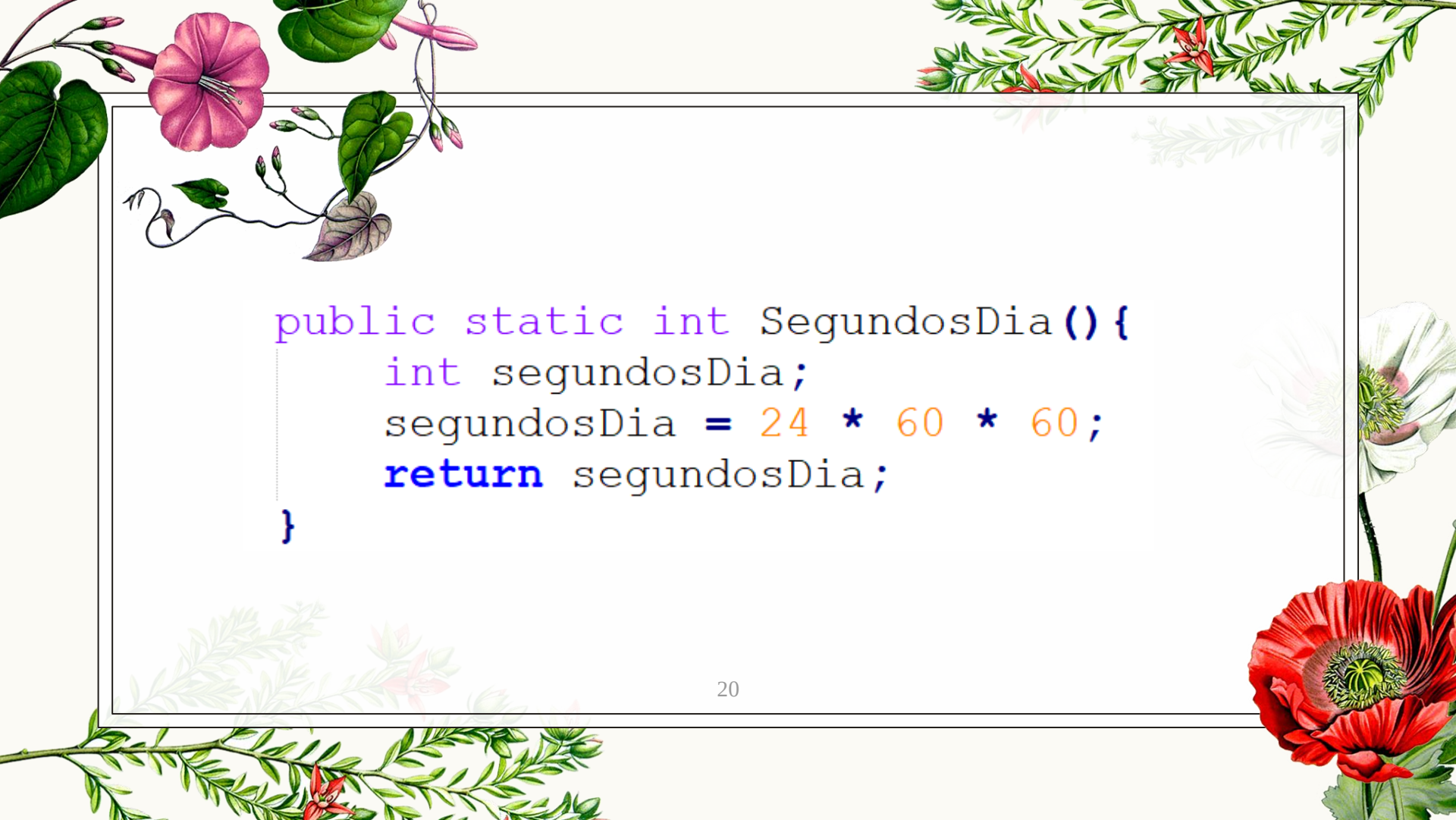
```
public static void Cabecalho () {  
    System.out.println (" +-----+ " );  
    System.out.println (" |   XXXX   XXX   XXXX   | " );  
    System.out.println (" |   X  X   X  X   X     | " );  
    System.out.println (" |   XXXX   X  X   XXXX   | " );  
    System.out.println (" |   X  X   X  X       X   | " );  
    System.out.println (" |   X  X   XXX   XXXX   | " );  
    System.out.println (" +-----+ " );  
}
```



sem Parâmetros com Retorno

2. Crie um método com o nome **SegundosDia** que retorne a quantidade de segundos que existem em um dia.



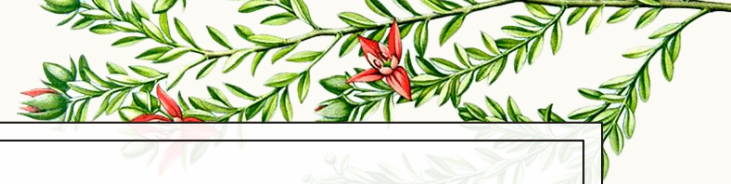


```
public static int SegundosDia () {  
    int segundosDia;  
    segundosDia = 24 * 60 * 60;  
    return segundosDia;  
}
```

com Parâmetros sem Retorno

3. Crie um método com o nome **Rodape** que receba uma mensagem motivacional e mostre o seguinte layout:

```
***** MENSAGEM MOTIVACIONAL *****
```



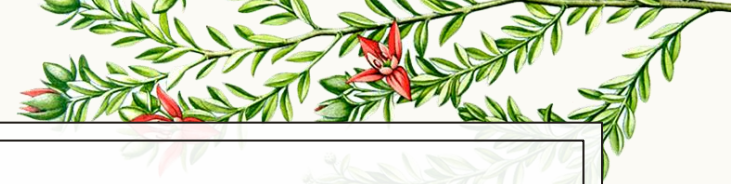
```
public static void Rodape (String mensagem) {  
    System.out.println("\n\n***** " + mensagem + " *****");  
}
```





com Parâmetros com Retorno

4. Crie um método com o nome **CalculaIdade** que receba o ano atual e o ano de nascimento e retorne a idade de uma pessoa



```
public static int CalculaIdade(int anoAtual, int anoNasc){  
    int idade;  
    idade = anoAtual - anoNasc;  
    return idade;  
}
```



Programa Principal

5. Crie um programa com o nome **Principal** que receba o nome, o ano de nascimento de uma pessoa e a mensagem motivacional e calcule a quantidade de segundos vividos por ela. Invoque todos os métodos criados nos exercícios anteriores. Esse deve ser o layout de saída:

```
+-----+
| XXXX  XXX  XXXX |
| X  X  X  X  X   |
| XXXX  X  X  XXXX |
| X  X  X  X     X |
| X  X  XXX  XXXX  |
+-----+

Lucilia Gomes Ribeiro,
Você já viveu 1.639.872.000 segundos

***** TOMÉ VACINA!!! *****
```



```
public static void main(String[] args) {
    Scanner dado = new Scanner(System.in);
    String nome, msg;
    int anoAtual = 2021, anoNasc, idade, segVividos;

    System.out.print("\n\nDigite seu nome: ");
    nome = dado.nextLine();
    System.out.print("Digite seu ano de nascimento: ");
    anoNasc = dado.nextInt();
    dado.nextLine();
    System.out.print("Mensagem motivacional: ");
    msg = dado.nextLine();
    idade = CalculaIdade(anoAtual, anoNasc);
    segVividos = idade * 365 * SegundosDia();

    Cabecalho();
    System.out.printf("\n\n%s,\nVocê já viveu %,d segundos", nome, segVividos);
    Rodape(msg);
}
```





Obrigada!

Alguma Pergunta?

professora@lucilia.com.br



Créditos

- Template: SlidesCarnival
- Ilustrações: Köhler's Medizinal-Pflanzen in naturgetreuen at BHL
- Deitel: “Java – Como programar”
- Ascencio: “Fundamentos de programação”
- Wikipedia