



UC Programação de Sistemas Computacionais

Professor Jean Carlo Wagner

jean.wagner@ecossistemaanima.com.br

- Modularização
- Tipos de métodos
- Programação Orientada a Objetos
- Classe
- Criando um objeto

Modularização

Modularização

- Dividir um problema em partes, denominadas módulos;
- Cada módulo resolve uma tarefa específica;
- O gerenciamento é feito pelo módulo principal;
 - que “chama” ou aciona os outros módulos.

O que são métodos

- **Em Java, chamamos as funções de métodos.**
 - São trechos de códigos que permitem modularizar um sistema;
 - dividir um sistema em pequenos blocos;
 - possuem um nome;
 - podem ser chamados várias vezes durante a execução de uma classe.

```
public static void main(String[] args){  
    double quadrado = Math.pow(2, 10);  
    System.out.println("string de impressão");  
}
```

Vantagens de criar métodos

- Redução do tamanho total de código de um sistema;
 - evita repetição de código.
- Modularização;
 - cada trecho de código realiza uma tarefa específica.
- Facilidade e agilidade na manutenção;
 - um ponto único de alteração.

Tipos de métodos

Método sem retorno (void) — Definição

- Executam uma tarefa e não retornam nenhum valor;
- Estrutura:

```
1      qualificador void nomeMetodo(<Parametros>){  
2          corpo do metodo...  
3      }
```


Método sem retorno (void) — Sintaxe

- **qualificador:**
 - static – método da classe
- **void:**
 - sem retorno — não retorna nada ao chamador!
- **nomeMetodo:**
 - é um identificador. Boas Práticas de Programação: Usar verbos de ação!
- **Parâmetros:**
 - Opcionais. Lista de argumentos que serão passados para o método, separados por vírgula.
 - Deve-se especificar o tipo de cada parâmetro

Exemplo 1 — void e sem parâmetros

```
public static void main(String[] args){  
    imprimirDadosDisciplina();  
}  
  
static void imprimirDadosDisciplina(){  
    System.out.println("=====");  
    System.out.println("\tProgramação Orientada a Objetos");  
    System.out.println("\tCarga Horária: 66 horas");  
    System.out.println("\tProfessor(a): Charles");  
    System.out.println("=====");  
}
```

Exemplo 2 — void e com parâmetros

```
public static void main(String[] args){
    imprimirDadosDisciplina("Charles");
}

static void imprimirDadosDisciplina(String nome){
    System.out.println("=====");
    System.out.println("\tProgramação Orientada a Objetos");
    System.out.println("\tCarga Horária: 66 horas");
    System.out.println("\tProfessor(a): " + nome);
    System.out.println("=====");
}
```


Exemplo 3 — void com parâmetros

```
public static void main(String[] args){  
    somar(10, 50);  
}  
  
static void somar(int n1, int n2){  
    int resultado;  
    resultado = n1 + n2;  
    System.out.println("Soma: " + resultado);  
}
```

Exercício 1

- Escreva uma método que:
 - receba um número n como parâmetro de entrada.
 - imprima a sequência de números de 1 até n .
- Exemplo:
 - $n = 10$;
 - saída do programa: 1 2 3 4 5 6 7 8 9 10

Exercício 2— Modularize o código

```
import java.util.Scanner;

public class Calculo {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double resultado = 0;

        System.out.println("Digite tres numeros");
        int n1 = entrada.nextInt();
        int n2 = entrada.nextInt();
        int n3 = entrada.nextInt();

        resultado = (n1 + n2) / n3;

        System.out.println("-----");
        System.out.println("Resultado:");
        System.out.println(resultado);
        System.out.println("-----");

        System.out.println("Digite outro numero");
        int n4 = entrada.nextInt();

        resultado = resultado + n4;

        System.out.println("-----");
        System.out.println("Resultado:");
        System.out.println(resultado);
        System.out.println("-----");
        entrada.close();
    }
}
```


Resolução

```
import java.util.Scanner;

public class Calculo2 {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double resultado = 0;

        System.out.println("Digite tres numeros");
        int n1 = entrada.nextInt();
        int n2 = entrada.nextInt();
        int n3 = entrada.nextInt();

        resultado = (n1 + n2) / n3;

        imprimir(resultado);

        System.out.println("Digite outro numero");
        int n4 = entrada.nextInt();

        resultado = resultado + n4;

        imprimir(resultado);
        entrada.close();
    }

    public static void imprimir(double resultado) {
        System.out.println("-----");
        System.out.println("Resultado:");
        System.out.println(resultado);
        System.out.println("-----");
    }
}
```

Exercício 3

- Escreva o método `exibirMes()` que:
 - recebe um número inteiro como parâmetro;
 - imprime o mês correspondente ao número.
- Exemplo:
 - 2 corresponde à “fevereiro”.
- Caso o número recebido não faça sentido;
 - o procedimento deve mostrar uma mensagem de erro.

Métodos com retorno

- Após serem chamados:
 - executam sua tarefa;
 - devolvem um valor ao chamador.
- O tipo de retorno pode ser: int, float, double ...
 - Para isso utilizar a palavra reservada **return** dentro do método.
- Estrutura:

```
1  qualificador <tipoRetorno> nomeMetodo(<parametros>){  
2      // corpo do metodo  
3      return <variavel de retorno>;  
4  }
```


Métodos com retorno — Sintaxe

- **qualificador:**
 - static – método da classe
- **tipoRetorno:**
 - indica o tipo do valor que será retornado!
- **nomeMetodo:**
 - é um identificador. Boas práticas de programação: Usar verbos de ação!
- **Parâmetros:**
 - Opcionais. Lista de argumentos que serão passados para o método, separados por vírgula.
 - Deve-se especificar o tipo de cada parâmetro

Exemplo 1 — Método com retorno e com parâmetros

```
public static void main(String[] args){  
    int valorRetornado;  
    valorRetornado = somar(10, 50);  
    System.out.println("Soma: " + valorRetornado);  
}
```

```
static int somar(int n1, int n2){  
    int resultado;  
    resultado = n1 + n2;  
    return resultado;  
}
```

Exercício 4

- Criar o método chamado encontrarMax() que:
 - recebe dois parâmetros do tipo int.
 - retorna o maior valor entre dois números.

Exercício 5

- Faça um método que:
 - recebe um número inteiro como parâmetro;
 - retorne verdadeiro (**true**) se o número for par;
 - ou retorne falso (**false**) se for ímpar.
 - nomeie o método como isPar().

Exercício 6

- Criar o método chamado `calculaSomatorio()` que:
 - recebe como parâmetro um número inteiro n ;
 - calcula a soma dos número de 1 até n .
 - retorna a soma dos números.

POO

Programação Orientada à Objetos

- A programação orientada a objetos nos ajuda a modelar o programa;
- Pensar nos componentes necessários para resolver um problema;
- Para isso ...
 - Tentaremos visualizar características em comum dos dados;
 - Criaremos estruturas que representem tais características.
 - Pensaremos na forma em que essas estruturas se comunicam



Created by Nikita Kosh
from Rice Project

Programação Orientada à Objetos

- As estruturas mencionadas anteriormente são os **objetos**!
- Ideia geral:
 - um programa orientado a objetos possui vários **objetos** que relacionam-se entre si.

Classe

Classe

- **A classe é um modelo que define a forma de um objeto**
 - Seria como um molde capaz de gerar elementos do mesmo tipo.



Classe

- **Especifica os dados e os comportamentos do objeto.**
- É uma abstração de algo:
 - Mundo real;
 - Modelos matemáticos;
 - Estruturas de dados....

Atributos e comportamentos

- **Atributos:**

- Definem as características de um objeto
- Denominados: Variáveis (atributos)
 - (quais informações temos deste objeto?)

- **Comportamentos**

- Definem as ações que um objeto pode executar
- Denominados: Métodos
 - (o que o objeto pode fazer?)

Forma geral de uma Classe

```
1 public class NomeClasse {  
2     // declara variaveis - atributos  
3     tipo var1;  
4     tipo var2;  
5     // ...  
6     tipo varN;  
7  
8     // declara metodos - comportamentos  
9     tipo metodo1(parametros){  
10         // corpo do metodo  
11     }  
12  
13     tipo metodo2(parametro){  
14         // corpo do metodo  
15     }  
16  
17     // ...  
18  
19     tipo metodoN(parametro){  
20         // corpo do metodo  
21     }  
22 }
```

Criando um objeto

Criando um objeto

- A classe apenas descreve um objeto;
- Os objetos serão as estruturas que estaremos efetivamente utilizando.
- Para utilizar um objeto precisamos **instanciá-lo**.

Criando um objeto

- Vamos criar uma classe do tipo Pessoa.
- Crie o arquivo: Pessoa.java
 - O nome do arquivo deve ser igual ao nome da classe.

```
1 public class Pessoa{  
2  
3   String nome;  
4   int idade;  
5   double altura;  
6  
7 }
```

Criando um objeto

- Agora precisamos **instanciar** um objeto dessa classe.
- Para isso, vamos criar um classe que tenha um método main.

- Crie o arquivo:
Main.java

```
1 public class Main {  
2  
3     public static void main(String[] args) {  
4  
5     }  
6  
7 }
```

Criando um objeto

- Agora devemos criar uma referência na memória para o objeto.

```
1 Pessoa objetoPessoa
```

- Depois devemos chamar o **construtor** usando o operador **new**

```
1 Pessoa objetoPessoa = new Pessoa();
```

Criando um objeto

- Instanciando um objeto do tipo Pessoa;

```
1 public class Main {  
2  
3     public static void main(String[] args) {  
4         Pessoa objetoPessoa = new Pessoa();  
5     }  
6  
7 }
```


Alterando o estado dos objetos

- Uma vez que declaramos um objeto do tipo pessoa;
- podemos acessar seus atributos e alterá-los.

```
public class Main {  
  
    public static void main(String[] args) {  
        Pessoa objetoPessoa = new Pessoa();  
  
        // insere dados  
        objetoPessoa.nome    = "Maria";  
        objetoPessoa.idade   = 24;  
        objetoPessoa.altura  = 65.0;  
  
        // acessa dados  
        System.out.println("Nome: "    + objetoPessoa.nome);  
        System.out.println("Idade: "   + objetoPessoa.idade);  
        System.out.println("Altura: "  + objetoPessoa.altura);  
    }  
}
```

Método toString()

- Convencionalmente utilizamos um método chamado **toString()** para imprimir os dados de um objeto.

```
public class Pessoa {  
  
    String nome;  
    int idade;  
    double altura;  
  
    public String toString() {  
        return "Nome: " + nome + "\nIdade: " + idade + "\nAltura: " + altura;  
    }  
}
```

- **Importante!**

- Os métodos de um objeto não possuem o qualificador **static**

Utilizando o método toString()

```
1 public class Main {  
2  
3     public static void main(String[] args) {  
4         Pessoa objetoPessoa = new Pessoa();  
5  
6         // insere dados  
7         objetoPessoa.nome    = "Maria";  
8         objetoPessoa.idade    = 24;  
9         objetoPessoa.altura   = 65.0;  
10  
11        // acessa dados  
12        System.out.println(objetoPessoa.toString());  
13    }  
14 }
```

Exercício 1

- Escreva uma classe que:
 - Represente uma lâmpada que está à venda em um supermercado.
 - Quais atributos e métodos sua classe deve ter?
- Crie uma outra classe que tem um método main;
- e instancie um objeto lâmpada para testar sua classe.

Exercício 2

- Crie uma classe smartphone.
- Pense nos atributos e comportamentos dos objetos dessa classe.
- Crie uma outra classe que tem um método main;
- e instancie alguns objetos da sua classe.

Exercício 3

- Escreva uma classe Aluno que possui as seguintes informações:
- Atributos:
 - Nome
 - RA
 - nota N1
 - nota N2,
- Comportamento:
 - calcular a média final;
 - mostrar dados do aluno;
- Teste sua classe.

ã Dúvidas!?!



