Entendendo Métodos Java

**Conceituação de métodos**

**O que é um método?**

É uma porção de código (sub-rotina) que é disponibilizada por uma classe. Este é executado quando é feita uma requisição a ele. São responsáveis por definir e realizar um determinado comportamento.

**Padrão de definição**

<?visibilidade?> <? Tipo?> <?modificador?> retorno

nome (<?parâmetros?>) <?exceções?> corpo

**Onde**:

* Visibilidade: “public, “protected” ou “private”;
* Tipo: Concreto ou abstrato;
* Modificador: “static” ou “final”;
* Retorno: tipo de dado ou “void” (void é vazio, no final não retorna a nada.);
* Nome: nome que é fornecido ao método;
* Parâmetros: parâmetros que pode receber;
* Exceções: exceções que pode lançar;
* Corpo: código que possui ou vazio.

Obs: os métodos sublinhados são os mais comuns e mais utilizados.

**Exemplos**:

* public String getNome() {...);
* public Double calcularTotalNota() {...};
* public int verificarDistancia (int coordenada1, int coordenada2) {...}; //duas coordenadas
* public abstract void executar(); // corpo vazio
* public void alterarFabricante (Fabricante fabricante){...};
* public Relatorio gerarDadosAnaliticos(Cliente cliente List <Compra> compras) {...};
* public static R N (P){...} // usaremos mais esse padrão

**Utilização**:

Passa-se uma mensagem através de uma classe ou objeto.

nome\_da\_classe.nome\_do\_metodo(); ou nome\_da\_classe.nome\_do\_metodo(...);

nome\_do\_objeto.nome\_do\_metodo(); ou nome\_do\_objeto.nome\_do\_metodo(...);

Math.random(); ou Math.sqrt(4); // random gera número aleatório e sqrt é raiz quadrada.

usuário.getEmail(); ou usuário.alterarEndereco (endereco);

**Conceitos Inerentes aos métodos e boas práticas na sua criação**

**O que é assinatura?**

É a forma de identificar unicamente o método;

Exemplo: Ass = nome + parâmetros

* Métodos:

public double calcularTotalVenda (Double precoItem1, Double precoItem2, Double precoItem3) {...}

* Assinatura:

calcularTotalVenda (double precoItem1, double precoItem2, double precoItem3)

* Construtor e Destrutor:

São métodos especiais usados na OO, construtor cria objetos e destrutor auxilia na destruição desses objetos.

* Mensagem:

É o ato de solicitar ao método que o mesmo execute. Esta pode ser direcionada a um objeto ou a uma classe.

* Passagem de parâmetro:

- Por valor (cópia);

Exemplo: int i = 10;

Public void fazerAlgo (int i){

i = i +10

System.out.println (“Valor de i dentro: ” + i);

}

System.out.println(“Valor de i fora: ” + i);

-Por referência (endereço); // será visto em OO.

**Boas práticas para a criação de métodos:**

* Nomes devem ser descritivos, mas curtos;
* Notação camelo;

Ex. verificarSaldo(); executarTransferencia(); existeDebito();

* Deve possuir entre 80 e 120 linhas;
* Evite lista de parâmetros longa;
* Visibilidades adequadas

**Sobrecarga de métodos:**

É a capacidade de definir métodos para diferentes contextos, mas preservando seu nome.

Ex.

* converterParaInteiro (float f);
* converterParaInteiro (double d);
* converterParaInteiro (String s);
* converterParaInteiro (float f, RoundType rd);
* converterParaInteiro (double d, RoundType rd);
* converterParaInteiro (String s, RoundType rd);
* converterParaInteiro (RoundType rd, String s); //outro exemplo de parâmetros
* converterParaInteiro (); //outro exemplo de parâmetros

Mais exemplos em:

<http://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

<http://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html>

**Sobrecarga X Sobrescrita**

Ambos atuam sobre os métodos, mas com diferença em seus modos:

* Sobrecarga: mantêm o nome e muda de parâmetro;
* Sobrescrita: é aplicado herança de Orientação Objeto.

**Funcionamento, considerações e uso de retornos**

Retorno/Return – o método executa seu retorno quando:

* Completa todas suas instruções internas;
* Chega a uma declaração explicita de retorno;
* Lança uma exceção.

**Considerações**

* O tipo de retorno do método é definido na sua criação e pode ser um tipo primitivo ou objeto;
* O tipo de dado do return deve ser compatível com o do método;
* Se o método for sem retorno (void), pode ou não ter um “return” para encerrar sua execução.