UNIVERSIDADE DO ESTADO DE MINAS GERAIS
UEMG

Métodos em Java Parte 2

Kamila Rios

19/05/2015

Métodos

- Um método em Java é equivalente a uma função, subrotina ou procedimento em outras linguagens de programação;
- Não existe em Java o conceito de métodos globais. Todos os métodos devem sempre ser definidos dentro de uma classe;
- A sintaxe para construção de um método é a seguinte:

[moderadores de acesso] [*modificador*] [tipo do valor de retorno] [nome] ([*argumentos*]) {

//[Corpo do método] }

```
public static void main (String args[]) {
//Corpo do método}
static double calculaQuadrado (int x){
//Corpo do método}
```

Moderadores de Acesso ou Qualificadores de Método

- Java controla/restringe o acesso a **atributos** e **métodos** através do uso dos modificadores de acesso.
- Relacionado à visibilidade:
  - ✓ **public**: É o menos restritivo de todos. Atributos e métodos declarados como *public* em uma classe podem ser acessados pelos métodos da própria classe, por classes derivadas desta e por qualquer outra classe em qualquer outro pacote;
  - ✓ **protected**: Atributos e métodos definidos como *protected* são acessíveis pelos métodos da própria classe e pelas classes derivadas;
  - ✓ **private**: É o mais restritivo. Atributos e métodos declarados como *private* só podem ser acessados pelos métodos da própria classe.

Modificador do Método - Qualificadores de Método

- O modificador do método permite especificar algumas propriedades de um método, determinando como classes derivadas podem ou não redefinir ou alterar o método, e de que forma esse método será visível. São eles:

  - ✓ **static:** o método é compartilhado por todos os objetos instanciados a partir da mesma classe.

    - ✓ Um método *static* não pode acessar qualquer variável declarada dentro de uma classe (salvo se a variável estiver declarada também como **static**, pois não é capaz de discernir entre os diferentes objetos que compartilham esse método.

Modificador do Método

- **abstract**: Pode-se declarar um método sem especificar seu corpo, chamando-o de abstrato:
  - Funciona como uma espécie de lembrete para que alguma classe derivada complete a declaração fornecendo um corpo;
  - Assim, uma classe que contenha um método abstrato, ou que seja derivada de alguma classe que contenha um método abstrato, mas não complete sua declaração, não pode ser instanciada;
  - O uso da palavra-chave **abstract** é opcional, isto é, as duas declarações seguintes são equivalentes:

```
abstract void print();
void print();
```

Modificador do Método

- **final**: Especifica que nenhuma classe derivada pode alterar ou redefinir este método. Um método declarado como *final* deve ser obrigatoriamente seguido de um corpo;

- **synchronized**: Esta declaração é usada para desenvolver o programa de *processamento concorrente*. Seu propósito é impedir que dois métodos executando concorrentemente acessem os dados de uma classe ao mesmo tempo:
  - **synchromized** especifica que, se um método estiver acessando o dado, outros que também desejem acessá-lo têm que esperar.

Qualificadores do Método

- Quando nenhum qualificador de visibilidade é definido, os atributos e métodos podem ser acessados pelos métodos da própria classe, pelas classes derivadas e por qualquer outra classe dentro do mesmo pacote;

- Com exceção dos qualificadores de visibilidade, um método pode ter mais de um qualificador.

Tipos de Retorno de um Método

- O tipo de valor de retorno é especificado por uma palavra chave ou nome de classe na declaração do método, estabelecendo o valor que ele pode devolver;
- É obrigatório;
- Pode ser:
  - ✓ Tipo primitivo (*boolean, char, short, int, long, float e double*);
  - ✓ Referências (classes definidas pelo Java, por exemplo - classe String, e classes definidas pelo programador);
  - ✓ void.

- Um método que retorna valor, isto é, não declarado como **void**, deve conter a linha ***return*** **...;** que especifica o valor a ser retornado.
  - ✓ Por exemplo, **return x;** especifica que o valor da variável **x** será retornado.

Nome do Método

- O nome de um método deve começar com uma letra (a-z/A-Z), um *underscore* (_), ou um sinal de dólar ($). Os caracteres subsequentes podem incluir os dígitos de 0 a 9;

- **Convenção**:
  - ✓ Use verbos para nome de métodos.
  - ✓ Escreva a primeira letra do nome minúscula com cada letra inicial interna maiúscula.

  Exemplo:

  getNomeUsuario ()

  getPrecoMaximo()

Argumentos/Parâmetros do Método

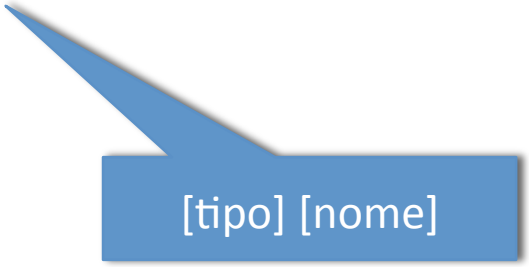
- Um método pode ter zero ou mais argumentos;
- Caso não tenha argumentos, é necessário informar os parênteses vazios tanto na definição como na chamada do método;
- O nome do método acrescido de seus parâmetros é denominado **assinatura do método**;
- Cada argumento deve ser declarado como define-se uma variável, especificando o seu tipo e nome;
- Caso haja mais de um argumento, cada declaração deve estar separada por vírgula.

Exemplo - Argumentos de Método

```
//Exemplo 1
public void setIdade (int newIdade)
{
idade= newIdade;
}
```

```
//Exemplo 2
public void displayIdade()
{
System.out.println("Idade é : " + idade);
}
```

```
//Exemplo 3
public void setUsuario (int newIdade, String newNome)
{
idade= newIdade;
nome= newNome;
}
```



Retornando valor a partir de um Método

- É possível associar um valor de retorno a um método:
  - ✓ Defina na frente do nome do método, o tipo do valor a ser retornado;
  - ✓ O tipo pode ser um primitivo ou uma classe;
  - ✓ Caso o método não retorne valor algum, é obrigatória a utilização do tipo void na assinatura do método.

```
//Exemplo 1
public int getIdade () {
return idade;
}
```

```
//Exemplo 2
public String getNome () {
return nome;
}
```

O comando *return*

- O comando return é utilizado para passar o valor requerido de volta para quem chamou o método e é obrigatório quando o tipo de retorno não é void;
- Esse comando deve receber um único valor ou uma expressão que deve ser compatível com o tipo de retorno especificado para o método. Quando o comando é encontrado, o método termina sua execução imediatamente, ignorando todas as instruções subsequentes.
- Um comando return sem valor de retorno retorna o tipo void, mas não é obrigatória sua utilização nestes casos.

Passando Parâmetros para um Método

- Sempre que um método receber dados, estes serão recebidos através dos parâmetros;

- Em Java, assim como em C ou C++, é preciso deixar claro quais e qual o tipo dos dados que serão passados;

- Em algumas linguagens, como Perl, isso não é necessário:
  - Isso tem a ver com tipagem, alocação de memória e outras coisas de mais baixo nível.

```
//Exemplo
public int metodo(int numero){
   //código
   return intNum;
}
```

Declarações que se pode fazer com um método

```
//mais simples

class Numero {

  double x = 1;
  void print() {
    System.out.println("O valor e " + x);
  } //fim do método

}//fim da classe Numero
```

```
//com valor de retorno

class Calculadora {
   int Soma(int a, int b) {
      return a + b;
   } //fim do método Soma
   double Produto(double a, double b)
{
      return a * b;
   }//fim doa método Produto
}//fim da classe Calculadora
```

Exercícios

1. Crie um aplicativo java que solicite ao usuário um número inteiro e retorne o quadrado desse valor:

   ✓ Crie o método main e um método chamado quadrado.

2. Crie um aplicativo java que solicite ao usuário seu peso e sua altura e retorne o seu IMC (Índice de Massa Corporal):

   ✓ IMC é calculado pelo peso (em kilogramas) dividido pelo quadrado da altura (em metros) da pessoa.

Resposta sugerida para exe. 1

```
import java.util.Scanner;
public class Quadrado {
   public static int quadrado(int num) {
      int quadrado;
      quadrado = num * num;
      return quadrado;
  } //fim do método quadrado
   public static void main(String[] args) {
      int numero, numero_quadrado;
      Scanner entrada = new Scanner(System.in);
      System.out.print(”Digite um inteiro: ");
      numero = entrada.nextInt();

      numero_quadrado=quadrado(numero);
      System.out.printf("%d elevado ao quadrado é %d", numero, numero_quadrado);
   }//fim do método main
}//fim da classe quadrado
```

Resposta sugerida para exe. 2

```
import java.util.Scanner;
public class IMC {

   public static float IMC(float peso, float altura){
      float imc;
      imc = peso/(altura*altura);
      return imc; }

   public static void main(String[] args) {
      float peso, altura, imc;
      Scanner entrada = new Scanner(System.in);

      System.out.print(”Digite seu peso, em kilos: ");
      peso = entrada.nextFloat();

      System.out.print(”Digite sua altura, em metros: ");
      altura = entrada.nextFloat();

      imc = IMC(peso, altura);

      System.out.printf("Seu IMC vale: %.2f",imc); }
}
```