Técnico em Desenvolvimento de Sistemas

Lógica de Programação II

Métodos

Alex Helder Cordeiro do Rosário de Oliveira

Instituto Federal de Brasília - *Campus* Brasília

Objetivo da Aula

- Implementar corretamente métodos em Java.

Métodos

- São as ações que o objeto pode realizar;
- Todos os objetos de uma mesma classe possuem as mesmas definições de método;
- Os métodos são declarados e implementados dentro de classes;
- Cada método consiste em um conjunto de instruções sequenciais que podem retornar um resultado;
- A lógica interna dos métodos segue paradigma estruturado;
- A execução do método utiliza os atributos do objeto cujo método está sendo executado.

Carro.java

```
public class Carro {
    String cor;
    String placa;
    void mudaCor(String novaCor) {
        cor = novaCor;
    }
    String corAtual() {
        return cor;
    }
    void mudaPlaca(String novaPlaca) {
        placa = novaPlaca;
    }
    String placaAtual() {
        return placa;
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        Carro fusca = new Carro();
        Carro brasilia = new Carro();
        fusca.mudaCor("preto");
        brasilia.mudaCor("amarela");
        fusca.mudaPlaca("ABC-1234");
        brasilia.mudaPlaca("XYZ-5555");
        JOptionPane.showMessageDialog(null, brasilia.corAtual());
        JOptionPane.showMessageDialog(null, brasilia.placaAtual());
        JOptionPane.showMessageDialog(null, fusca.corAtual());
        JOptionPane.showMessageDialog(null, fusca.placaAtual());
    }
}
```

Métodos

- Retorno;
- Nome;
- Argumentos de entrada;
- Corpo.

```
retorno nomeDoMetodo(int arg1, Tipo arg2, OutroTipo arg3) {
    corpo;
}
```

Métodos

```
String metodoQualquer (boolean[] arg1, int arg2) {
    String texto;
    if(arg1.length < 10) {
        texto = Integer.toString(arg2);
    } else {
        texto = Boolean.toString(arg1[10]);
    }
    return texto;
}
```

Argumentos

- Podem haver quantos argumentos quanto forem necessários;
- Pode ser que não seja necessário nenhum argumento;
  - Neste caso, deixa-se os parenteses vazios.
- Cada argumento deve ser declarado com o tipo e nome;
- Os argumentos podem ser de qualquer tipo (primitivo, objeto ou arrays).

Argumentos

- Quando é chamado um método, os valores presentes no argumento da chamada são copiados para as variáveis declaradas no argumento do método na mesma ordem.
- Não é necessário que o nome da variável na chamada do método seja a mesma declarada no argumento.

```
void solicitaSoma() {
    int x = 4;
    int y = 3;
    int z = soma(x,y);
}
// a receberá o valor 4 (presente em x)
// b receberá o valor 3 (presente em y)
int soma(int a, int b) {
    int c = a + b;
    return c;
}
```

Retorno

- Declara-se apenas o tipo do retorno (mas não seu nome);
- O retorno pode ser de qualquer tipo;
- Só pode ser retornado apenas o valor de uma variável;
- Caso não tenha nada para retornar, utiliza-se a palavra-chave `void`.
- Quando é declarado um retorno, deve-se incluir o comando `return` com o valor ou variável do mesmo tipo declarado no retorno.

Retorno

- O valor presente no comando `return`;

```
void solicitaAlgo() {
    int k = simples(); // k recebe valor 4.
}
int simples() {
    int v = 4;
    return v;
}
```

- Se não houver tipo de retorno, o comando `return`, se houver não deve trazer nenhum argumento;
- Quando o comando `return` é executado, a função se encerra, desconsiderando todos os outros comandos após ele dentro da função, e o processamento retorna de volta ao ponto onde a função foi chamada.

Nome

- Pode ser formado por letras, algarismos, cifrão (‘$’) ou traço baixo (‘_’);
- O nome não pode iniciar por algarismos;
- Formado a partir de verbos (ações)*.
- A primeira letra deve ser minúscula*.
- Cada palavra, exceto a primeira, deve ter a primeira letra maiúscula e as outras minúsculas*.
- As palavras devem ser colocadas em sequência direta sem o uso de traço baixo para separá-las (elas não devem ser separadas)*.
- Podem existir mais de um método com o mesmo nome, desde que os argumentos sejam diferentes (*Sobrecarga*).

*Convenção de código: não é obrigatório, mas é fortemente recomendado.

Sobrecarga

- É o polimorfismo ad hoc;
- Chamado Overloading em inglês;
- Pode-se definir métodos com mesmo nome, diferenciados pelos seus argumentos;
- A diferença na assinatura permite diferentes métodos apesar de terem mesmo nome;
- Pode-se mudar tanto quantidade quanto tipos dos argumentos;
- Manter o tipo de argumento e alterar apenas o nome do argumento não é uma diferença válida;
- Não podemos mudar o tipo de retorno sem mudar o argumento.

Sobrecarga - Exemplo

```
public class ExemploDeSobrecarga {
    double calculo(double a, double b, double c){
        return a+b+c;
    }
    double calculo(double a, double b){
        return a+b;
    }
    String calculo(String s, String t) {
        double d = Double.parseDouble(s);
        double o = Double.parseDouble(t);
        return Double.toString(d+o);
    }
    void exec() {
        double a = calculo(3, 5, 11);
        double b = calculo(5.3, 4.75);
        String s = calculo("13.5", "46.2");
    }
}
```

Sobrecarga - Exemplo

- No exemplo anterior, temos três métodos chamados `calculo`.
- O que diferencia um do outro é o seu argumento.
- Quando temos uma chamada com 3 argumentos do tipo `double`, ele executará o método cuja assinatura tem 3 variáveis do tipo `double`.
- Quando temos uma chamada com 2 argumentos do tipo `double`, ele executará o método cuja assinatura tem apenas 2 variáreis do tipo `double`.
- Quando temos uma chamada com 2 argumentos do tipo `String`, ele executará o método cuja assinatura tem 2 variáveis do tipo `String`.

Sobrecarga - Exemplo

- Se tivéssemos uma chamada com 4 argumentos do tipo `double`, ele não compilaria, pois não temos nenhum método com 4 variáveis do tipo `double`.
- Se tivéssemos uma chamada com 3 argumentos do tipo `String`, ele não compilaria, pois não tempos nenhum método com 3 variáveis do tipo `String`.
- Se tivéssemos uma chamada sem argumentos, ele também não compilaria, pois não temos nenhum método sem variáveis em seu argumento.

Métodos

- Para acessar o método de um objeto, deve-se usar o nome do objeto, seguido de ponto, seguido do nome do método.

```
class Objeto {
    void metodo() {
        System.out.println("Oi mundo!");
    }
}
class Programa {
    void exec() {
        Objeto obj = new Objeto();
        obj.metodo(); // Chamada do método
    }
}
```

Método

- Os métodos tem acesso direto aos outros membros do objeto*.

```
class ContaCorrente {
    float saldo;
    void realizaSaque(float valor) {
        procedimentoDeSeguranca();
        if (valor <= saldo) {
            saldo -= valor;
            System.out.println("Saque realizado.");
        } else {
            System.out.println("Saque insuficiente.");
        }
    }
    void procedimentoDeSeguranca() {
    }
}
```

*Com exceção dos métodos estáticos, que são independentes da existência do objeto e só tem acesso direto a outros membros estáticos.

Sua vez:

- Escreva um método que mostre seu nome na tela. Escreva um programa, cujo método main() chame o método criado para escrever seu nome.

Sua vez:

- Escreva um método que receba um número inteiro e retorne o dobro deste número. Escreva um programa, cujo método main() chame o método criado para dobrar o valor do número passado.