02

Chamadas de métodos nos objetos

Além de acessar atributos, também podemos invocar métodos em um objeto. Para isso usamos o operador `.` (ponto), junto a uma variável de referência para um objeto. Deve-se prestar atenção ao número e tipo de parâmetros do método, além do seu retorno. Métodos declarados como `void` não possuem retorno, logo, não podem ser atribuídos a nenhuma variável ou passado para outro método como parâmetro:

```
class Pessoa{

    String nome;

    public String getNome(){
        return nome;
    }

    public void setNome(String nome){
        this.nome = nome;
    }
}

class Teste{
    public static void main(String[] args){
        Pessoa p = new Pessoa();

        //chamando método na variável de ref.
        p.setNome("Mario");
```

```
        //Atribuindo o retorno do método a variável.
        String nome = p.getNome();

        // erro, método é void
        String a = p.setNome("X");
    }
}
```

COPIAR CÓDIGO

Quando um método está sendo invocado em um objeto, podemos chamar outro método no mesmo objeto através da invocação direta ao nome do método:

```
class A {
    void metodo1() {
        metodo2(); // chama o metodo2 no objeto onde metodo1 foi
                   // chamado
    }
    void metodo2() {
    }
}
```

COPIAR CÓDIGO

Argumentos variáveis: varargs

A partir do Java 5, **varargs** possibilitam um método que receba um número variável (não fixo) de parâmetros. É a maneira de receber um array de objetos e possibilitar uma chamada mais fácil do método.

Um caso especial é quando método recebe um argumento variável (`varargs`). Neste caso, podemos chamá-lo com qualquer número de argumentos:

```
class Calculadora{
    public int soma(int... nums){
        int total = 0;
        for (int a : nums){
            total+= a;
        }
        return total;
    }

}
```

COPIAR CÓDIGO

`nums` realmente é um array aqui, você pode fazer um `for` usando o `length`, ou mesmo usar o `enhanced for`. A invocação desse método pode ser feita de várias maneiras:

```
    public static void main (String[] args){
        Calculadora c = new Calculadora();

        //Todas as chamadas abaixo sao válidas
        System.out.println(c.soma());
        System.out.println(c.soma(1));
        System.out.println(c.soma(1,2));
        System.out.println(c.soma(1,2,3,4,5,6,7,8,9));
    }
```

COPIAR CÓDIGO

Em todos os casos, um array será criado, nunca `null` será passado. Um parâmetro `varargs` deve ser sempre o último da assinatura do método para

evitar ambiguidade. Isso implica que apenas um dos parâmetros de um método seja `varargs`. E repare que os argumentos variáveis têm que ser do mesmo tipo.

E será dada a prioridade para o método que já podia existir antes no Java 1.4:

```
void metodo(int ... x) { }
void metodo(int x) {}

metodo(5);
```

COPIAR CÓDIGO

Isso vai invocar o segundo método. Podemos também passar um array de `ints` para um método que recebe um `varargs`:

```
void metodo(int ... x) { }

metodo(new int[] {1,2,3,4});
```

COPIAR CÓDIGO

Mas nunca podemos chamar um método que recebe array como se ele fosse `varargs`:

```
void metodo(int[] x) { }

metodo(1,2,3); // não compila
```

COPIAR CÓDIGO