Aula 05 – Subrotinas e Condicionais

Norton Trevisan Roman

22 de março de 2013

Visão geral do código

```
class AreaCasa {
    /*
        Calcula a área da casa

        Parâmetros:
            lateral - comprimento da lateral da cabana
            cquarto - lateral maior do quarto
    */
    static void areaCasa(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "
                           +areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "+areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);
}
```

```
/*
    Calcula a área da piscina

    Parâmetros:
        raio - O raio da piscina
*/
static double areaPiscina(double raio) {
    return Math.PI * Math.pow(raio,2);
}

public static void main(String[] args) {
    double areap; // área da piscina

    areaCasa(11,7);

    areap = areaPiscina(2);
    System.out.println("A área da piscina é "
                       +areap);
}
}
```

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não.

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal
- `(int)(valor*100)` ?

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- `(int)(valor*100)` ?
  - Multiplique valor (double) por 100

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- `(int)(valor*100)` ?
  - Multiplique valor (double) por 100
    - ★ O valor será double

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- `(int)(valor*100)` ?
  - Multiplique valor (double) por 100
    - ★ O valor será double
  - O resultado transforme em um inteiro

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- `(int)(valor*100)` ?
  - Multiplique valor (double) por 100
    - ★ O valor será double
  - O resultado transforme em um inteiro
    - ★ Guarda 314 em *novoValor*

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de apresentar com apenas 2 casas decimais?
  - Não. Contudo, podemos implementar nossa função para tal

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- `(int)(valor*100)` ?
  - Multiplique valor (double) por 100
    - ★ O valor será double
  - O resultado transforme em um inteiro
    - ★ Guarda 314 em *novoValor*

- Atenção! – Esse método não responde muito bem quando os valores estão no limite do Java

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro
- Cuidado!

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- Cuidado!
  - Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- Cuidado!
  - Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)
    - ★ `return((double)novoValor/100)` não gerou perda

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- Cuidado!
  - Mudanças de tipos menores para maiores não geram perda (ex: int → long)
    - ★ `return((double)novoValor/100)` não gerou perda
  - Já de tipos maiores para menores podem gerar perda (ex: long → int)

Type Casting

- Ao final, transformamos *novoValor* novamente em *double*, para usar
  - E se tivéssemos feito `return(novoValor/100);`?
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        int novoValor = (int)(valor*100);
        return((double)novoValor/100);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

- Cuidado!
  - Mudanças de tipos menores para maiores não geram perda (ex: int → long)
    - ★ `return((double)novoValor/100)` não gerou perda
  - Já de tipos maiores para menores podem gerar perda (ex: long → int)
    - ★ `(int)(valor*100)` gerou uma perda

Type Casting

- Muito embora seja útil ver casting, nesse exemplo específico não é a melhor solução

Type Casting

- Muito embora seja útil ver casting, nesse exemplo específico não é a melhor solução
- A melhor solução seria fazer uso do operador %:

```
/*
Programa para truncar valores.
*/
class Truncar {

    /*
        Trunca um determinado valor em 2 casas
    */
    static double trunca(double valor) {
        return(valor - valor%0.01);
    }

    public static void main(String[] args) {
        System.out.println(trunca(Math.PI));
    }
}
```

Atributos

- Suponha agora que queremos também saber o valor da construção

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?
  - Duas alternativas:

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?
  - Duas alternativas:

    - ★ Definir o valor dentro do método:

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?
  - Duas alternativas:
    - ★ Definir o valor dentro do método:

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?
  - Duas alternativas:
    - ★ Definir o valor dentro do método:

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

    - ★ Passar o valor como parâmetro

Atributos

- Suponha agora que queremos também saber o valor da construção
  - Tomando como base o valor do metro quadrado
- Como fazer?
  - Duas alternativas:

★ Definir o valor dentro do método:

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

★ Passar o valor como parâmetro

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - ★ Ou devem ser buscadas dentro do método (onde quer que ele esteja no código)

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - ★ Ou devem ser buscadas dentro do método (onde quer que ele esteja no código)
    - ★ Ou devem ser buscadas em cada chamada ao método

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - ★ Ou devem ser buscadas dentro do método (onde quer que ele esteja no código)
    - ★ Ou devem ser buscadas em cada chamada ao método
- O preço do metro quadrado parece mais um atributo do problema como um todo

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - ★ Ou devem ser buscadas dentro do método (onde quer que ele esteja no código)
    - ★ Ou devem ser buscadas em cada chamada ao método
- O preço do metro quadrado parece mais um atributo do problema como um todo
  - É único para o programa como um todo

Atributos

- Qual das duas alternativas seria a melhor?

```
/*
    Calcula o valor total da construção
*/
static double valor(double area) {
    double valorM2 = 1500;

    return(valorM2*area);
}
```

```
/*
    Calcula o valor total da construção
*/
static double valor(double area,
                    double valorM2) {
    return(valorM2*area);
}
```

- Ambas apresentam problemas semelhantes:
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - ★ Ou devem ser buscadas dentro do método (onde quer que ele esteja no código)
    - ★ Ou devem ser buscadas em cada chamada ao método
- O preço do metro quadrado parece mais um atributo do problema como um todo
  - É único para o programa como um todo
  - Algo que, em softwares gerais, estaria em algum menu "Opções", "Setup" etc.

Atributos

- E como declarar atributos assim?

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças
  - E o static? ...

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças
  - E o static? ... depois...

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças
  - E o static? ... depois...

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

- E como podemos acessar o valor?

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças
  - E o static? ... depois...

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

- E como podemos acessar o valor?
  - De dentro de qualquer método (ou corpo) do programa

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    /*
        Calcula o valor total da
        construção
    */
    static double valor(double area){
        return(valorM2*area);
    }
}
```

Atributos

- E como declarar atributos assim?
  - Fora de qualquer método no programa
  - Deixamos variável para permitir mudanças
  - E o static? ... depois...

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
}
```

- E como podemos acessar o valor?
  - De dentro de qualquer método (ou corpo) do programa
  - Como faríamos com uma constante

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    /*
        Calcula o valor total da
        construção
    */
    static double valor(double area){
        return(valorM2*area);
    }
}
```

Escopo

- Consideremos agora outro método do mesmo programa:

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    static void areaCasa(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "+areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "+areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);
}
...
    static double valor(double area) {
        areat = 3;
        valorM2 = 5;
        return(valorM2*area);
    }
    ...
}
```

Escopo

- Consideremos agora outro método do mesmo programa:
- Conseguiremos fazer essas atribuições?

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    static void areaCasa(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "+areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "+areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);
}
...
    static double valor(double area) {
        areat = 3;
        valorM2 = 5;
        return(valorM2*area);
    }
    ...
}
```

Escopo

- Consideremos agora outro método do mesmo programa:
- Conseguiremos fazer essas atribuições?

```
AreaCasa.java:51: cannot find symbol
symbol  : variable areat
location: class AreaCasa
        areat = 3;
        ^
1 error
```

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    static void areaCasa(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "+areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "+areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);
}
...
    static double valor(double area) {
        areat = 3;
        valorM2 = 5;
        return(valorM2*area);
    }
    ...
}
```

Escopo

- Consideremos agora outro método do mesmo programa:
- Conseguiremos fazer essas atribuições?

```
AreaCasa.java:51: cannot find symbol
symbol  : variable areat
location: class AreaCasa
        areat = 3;
        ^
1 error
```

  - *areat* não foi encontrada, mas *valorM2* foi

```
class AreaCasa {
    /* valor do metro quadrado */
    static double valorM2 = 1500;
    ...
    static void areaCasa(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "+areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "+areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);
}
...
    static double valor(double area) {
        areat = 3;
        valorM2 = 5;
        return(valorM2*area);
    }
    ...
}
```

Escopo

- Variáveis declaradas dentro de um método:

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade:

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade:

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa...

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa... Na verdade, não é bem isso, mas veremos mais adiante.

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa... Na verdade, não é bem isso, mas veremos mais adiante.
- Cuidado!

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa... Na verdade, não é bem isso, mas veremos mais adiante.
- Cuidado!
  - Da mesma forma que qualquer método pode acessar um atributo...

Escopo

- Variáveis declaradas dentro de um método:
  - Visibilidade: dentro do próprio método
  - Diz-se que seu escopo é o método
- Variáveis declaradas fora de qualquer método (inclusive o *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa... Na verdade, não é bem isso, mas veremos mais adiante.
- Cuidado!
  - Da mesma forma que qualquer método pode acessar um atributo... se ele não for *final*, qualquer método poderá também modificá-lo

```
static double valor(double area) {
    valorM2 = 5;
    return(valorM2*area);
}
```

Testando os Parâmetros

- Considere o código ao lado:

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos
- E como podemos testar?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos
- E como podemos testar?
  - SE o parâmetro for positivo ENTÃO calcule a área

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos
- E como podemos testar?
  - SE o parâmetro for positivo ENTÃO calcule a área
  - SENÃO...

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

- E como podemos testar?
  - SE o parâmetro for positivo ENTÃO calcule a área
  - SENÃO... Retorne um valor indicando erro

Testando os Parâmetros

- Considere o código ao lado:
- Qual será a saída?

```
O valor da construção é -30000.0
```

- O que aconteceu?
  - Não testamos o valor passado ao parâmetro
  - Por se tratar de uma área, não poderia aceitar valores negativos

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        return(valorM2*area);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

- E como podemos testar?
  - SE o parâmetro for positivo ENTÃO calcule a área
  - SENÃO... Retorne um valor indicando erro
    - ★ Nesse caso, pode ser -1

Testando os Parâmetros

- E como codificar isso?

Testando os Parâmetros

- E como codificar isso?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- E como codificar isso?
- >=?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- E como codificar isso?
- >=?
  - Operador Relacional:

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- E como codificar isso?
- $>=$?
  - Operador Relacional:

Matemática	*Computação*
$>$	$>$
$<$	$<$
$=$	$==$
$\neq$	$!=$
$\leq$	$<=$
$\geq$	$>=$

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?
  - Senão retorne -1

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?
  - Senão retorne -1
- O código dentro do *if* é executado somente se a condição entre parênteses for verdadeira

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?
  - Senão retorne -1
- O código dentro do *if* é executado somente se a condição entre parênteses for verdadeira
  - Se a condição for falsa, o código no *if* é ignorado

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?
  - Senão retorne -1
- O código dentro do *if* é executado somente se a condição entre parênteses for verdadeira
  - Se a condição for falsa, o código no *if* é ignorado
- O código dentro do *else* é executado somente se a condição for falsa

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- O que o código no *if* diz?
  - Se area ≥ 0, então faça o cálculo e retorne
- E o *else*?
  - Senão retorne -1
- O código dentro do *if* é executado somente se a condição entre parênteses for verdadeira
  - Se a condição for falsa, o código no *if* é ignorado
- O código dentro do *else* é executado somente se a condição for falsa
  - Se a condição for verdadeira, o código no *else* é ignorado

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        else {
            return(-1);
        }
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Será que tem como melhorar esse código?

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ...

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.
  - Nada mais será executado...

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.
  - Nada mais será executado... e o else é ignorado de qualquer forma

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.
  - Nada mais será executado... e o else é ignorado de qualquer forma

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        return(-1);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.
  - Nada mais será executado... e o else é ignorado de qualquer forma
- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o *return*

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        return(-1);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Será que tem como melhorar esse código?
- Precisa realmente do *else* nesse caso? ... Ou ele sempre é ignorado quando a condição for verdadeira?
  - Se a condição for verdadeira, há o retorno.
  - Nada mais será executado... e o else é ignorado de qualquer forma
- Note que, nesse caso, devido ao condicional, o compilador permite que haja código após o *return*
  - Não há como dizer de antemão se haverá o retorno

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) {
                return(valorM2*area);
        }
        return(-1);
    }

    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        System.out.println("O valor da construção é "
                           +preco);
        preco = valor(20);
        System.out.println("O valor da construção é "
                           +preco);
    }
}
```

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os $\{\}$ denotam bloco de comandos

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os $\{\}$ denotam bloco de comandos
  - E que basta um ; para denotar o fim de um único comando

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os $\{\}$ denotam bloco de comandos
  - E que basta um ; para denotar o fim de um único comando
- Então...

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) return(valorM2*area);
        return(-1);
    }
    ...
```

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os $\{\}$ denotam bloco de comandos
  - E que basta um ; para denotar o fim de um único comando
- Então...
- E como usamos isso no main?

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) return(valorM2*area);
        return(-1);
    }
    ...
```

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os {} denotam bloco de comandos
  - E que basta um ; para denotar o fim de um único comando
- Então...
- E como usamos isso no main?
  - O condicional evita que usemos um resultado inválido do método

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) return(valorM2*area);
        return(-1);
    }
    ...


    ...
    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        if (preco >= 0) System.out.println("O valor
                                da construção é "+preco);
        else System.out.println("Valor de área
                                negativo");
    }
```

Testando os Parâmetros

- Mas ainda dá pra deixar mais enxuto.
- Lembre que os {} denotam bloco de comandos
  - E que basta um ; para denotar o fim de um único comando
- Então...
- E como usamos isso no main?
  - O condicional evita que usemos um resultado inválido do método
  - Evita inconsistências futuras difíceis de serem encontradas

```
class AreaCasa {
/* valor do metro quadrado */
    static double valorM2 = 1500;

    /*
        Calcula o valor total da construção
    */
    static double valor(double area) {
        if (area >= 0) return(valorM2*area);
        return(-1);
    }
    ...


    ...
    public static void main(String[] args) {
        double preco; // área da piscina

        preco = valor(-20);
        if (preco >= 0) System.out.println("O valor
                            da construção é "+preco);
        else System.out.println("Valor de área
                            negativo");
    }
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?

```
if (condição) {
    ...
}
else {
    ...
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?

```
if (condição) {
    ...
}
else {
    ...
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso

```
if (condição) {
    ...
}
else {
    ...
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?

```
if (condição) {
    ...
}
else {
    ...
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?

```
if (condição) {
    ...
}
else {
    ...
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?
  - Tipo de variável que armazena apenas dois valores:

```
if (condição) {
    ...
}
else {
    ...
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?
  - Tipo de variável que armazena apenas dois valores:
    - ★ Verdadeiro (true)

```
if (condição) {
    ...
}
else {
    ...
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?
  - Tipo de variável que armazena apenas dois valores:
    - ★ Verdadeiro (true)
    - ★ Falso (false) – padrão

```
if (condição) {
    ...
}
else {
    ...
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Revendo, o que colocamos dentro do *if*?
- Condição?
  - Expressão que resulta em verdadeiro ou falso
- Usando esse conceito, haveria uma maneira alternativa de escrever o main?
  - Tipo de variável que armazena apenas dois valores:
    - ★ Verdadeiro (true)
    - ★ Falso (false) – padrão
    - ★ Valores lógicos

```
if (condição) {
    ...
}
else {
    ...
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false
    - ★ Mas ela já era false...

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false
    - ★ Mas ela já era false...

- Podemos então nos livrar dela, sem problemas

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false
    - ★ Mas ela já era false...
- Podemos então nos livrar dela, sem problemas

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false
    - ★ Mas ela já era false...

- Podemos então nos livrar dela, sem problemas
  - Se *preco* ≥ 0, então *valorOK* é feita true

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Analisando o código ao lado, precisamos mesmo do else?
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* é feita false
    - ★ Mas ela já era false...

- Podemos então nos livrar dela, sem problemas
  - Se *preco* ≥ 0, então *valorOK* é feita true
  - Senão, *valorOK* continua false

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;
    else valorOK = false;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá...

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá... true, pois é verdadeiro que $12 > 10$

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá... true, pois é verdadeiro que $12 > 10$
- Em vista disso, não poderíamos reescrever o momento de atribuição de valor de *valorOK*?

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá... true, pois é verdadeiro que $12 > 10$
- Em vista disso, não poderíamos reescrever o momento de atribuição de valor de *valorOK*?

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá... true, pois é verdadeiro que $12 > 10$
- Em vista disso, não poderíamos reescrever o momento de atribuição de valor de *valorOK*?
  - Se *preco* $>= 0$, então *valorOK* conterá true

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Variáveis booleanas aceitam true ou false
- Apenas isso?
  - Como valores, sim
  - Mas também aceitam expressões
  - `boolean valorOK = 12 > 10;`
    - ★ Nesse caso, *valorOK* conterá... true, pois é verdadeiro que $12 > 10$
- Em vista disso, não poderíamos reescrever o momento de atribuição de valor de *valorOK*?
  - Se *preco* $>= 0$, então *valorOK* conterá true
  - Senão, *valorOK* conterá false

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(-20);
    if (preco >= 0) valorOK = true;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor da
                            construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                            negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado
    - ★ Se não houver *else*, o programa continua normalmente

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado
    - ★ Se não houver *else*, o programa continua normalmente

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

- Não apenas expressões, mas também variáveis booleanas podem estar no *if*

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado
    - ★ Se não houver *else*, o programa continua normalmente

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

- Não apenas expressões, mas também variáveis booleanas podem estar no *if*
  - Analisadas do mesmo modo:

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado
    - ★ Se não houver *else*, o programa continua normalmente

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

- Não apenas expressões, mas também variáveis booleanas podem estar no *if*
  - Analisadas do mesmo modo:
    - ★ Se a variavel for verdadeira, então o corpo do *if* será executado

Variáveis Booleanas

- Considere agora o segundo *if*
  - O que acontece quando se encontra um condicional?
    - ★ A expressão dentro dos parênteses é testada
    - ★ Se seu resultado for verdadeiro, o código no corpo do *if* é executado
    - ★ Se for falso, o código no corpo do *else* é executado
    - ★ Se não houver *else*, o programa continua normalmente

```
public static void main(String[] args) {
    double preco; // área da piscina
    boolean valorOK = false;

    preco = valor(20);
    valorOK = preco >= 0;

    if (valorOK) System.out.println("O valor
                    da construção é "+preco);
    else System.out.println("Valor de área
                              negativo");
}
```

- Não apenas expressões, mas também variáveis booleanas podem estar no *if*
  - Analisadas do mesmo modo:
    - ★ Se a variavel for verdadeira, então o corpo do *if* será executado
    - ★ Se for falsa, será o corpo do *else* (se existir)

Visão geral do código

```
class AreaCasa {
    static double valorM2 = 1500;

    static void areaCasa(float lateral,
                         float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        System.out.println("Cálculo da área da casa");
        // cálculo da área da sala
        areas = lateral*lateral;
        System.out.println("A área da sala é "+areas);
        // cálculo da área do banheiro
        areaq = cquarto*(lateral/2);
        System.out.println("A área do banheiro é "
                           +areaq);
        // cálculo da área do quarto
        System.out.println("A área do quarto é "
                           +areaq);
        // cálculo da área total
        areat = areas + 2*areaq;
        System.out.println("A área total é " + areat);  }
    }

    static double areaPiscina(double raio) {
        return Math.PI * Math.pow(raio,2);
    }
```

```
    static double valor(double area) {
        if (area >= 0) return(valorM2*area);
        return(-1);
    }

    public static void main(String[] args) {
        double preco; // área da piscina
        boolean valorOK = false;

        preco = valor(20);
        valorOK = preco >= 0;

        if (valorOK) System.out.println("O valor da
                                construção é "+preco);
        else System.out.println("Valor de área
                                negativo");
    }
```