Aula 06 – Condicionais

Norton Trevisan Roman

9 de abril de 2018

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 usarmos apenas 2 casas decimais?

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de usarmos apenas 2 casas decimais?
  - Com `println`, não.

Type Casting

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

- `(int)(valor*100)` ?

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

- `(int)(valor*100)` ?
  - Multiplique `valor` (double) por 100 (O resultado será double)
  - Transforme esse resultado em um inteiro (Guarda 314 em `novoValor`)
- Atenção! – Esse método não responde muito bem quando os valores estão no limite do double

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

  /*
    Trunca um valor na 2a casa
  */
  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
  - E se tivéssemos feito `return(novoValor/100)`?

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

  /*
    Trunca um valor na 2a casa
  */
  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
  - E se tivéssemos feito `return(novoValor/100)`?
- Mudanças assim são chamadas de Type casting

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

  /*
    Trunca um valor na 2a casa
  */
  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
  - 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 valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

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

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

  /*
    Trunca um valor na 2a casa
  */
  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

- 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 valor na 2a casa
  */
  static double trunca(double valor) {
    return(valor - valor%0.01);
  }

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

Type Casting

- Alternativamente, verifique o uso do `printf` em java
- `https://docs.oracle.com/javase/tutorial/java/data/numberformat.html`

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:

```
static double valor(double
                    area) {
  double valorM2 = 1500;
  return(valorM2*area);
}
```

Atributos

- Ou passar o valor como parâmetro

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

Atributos

- Ou passar o valor como parâmetro

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

- Qual das duas alternativas seria a melhor?

Atributos

- Ou passar o valor como parâmetro

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

- Qual das duas alternativas seria a melhor?
- Ambas apresentam problemas semelhantes

Atributos

- Ou passar o valor como parâmetro

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

- Qual das duas alternativas seria a melhor?
- 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

- 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
- 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? ... depois...

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

Atributos

- E como podemos acessar o valor?

Atributos

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

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

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

Escopo

- Consideremos agora outro método do programa

```
class AreaCasa {
  static double valorM2 = 1500;
  ...
  static void areaCasa(float
           lateral, float cquarto) {
    float areaq;
    float areas;
    float areat;

    System.out.println("...");
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."+areat);
  }
  ...
  static double valor(double area) {
    areat = 3;
    valorM2 = 5;
    return(valorM2*area);
  }
  ...
}
```

Escopo

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

```
class AreaCasa {
  static double valorM2 = 1500;
  ...
  static void areaCasa(float
           lateral, float cquarto) {
    float areaq;
    float areas;
    float areat;

    System.out.println("...");
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."+areat);
  }
  ...
  static double valor(double area) {
    areat = 3;
    valorM2 = 5;
    return(valorM2*area);
  }
  ...
}
```

Escopo

- Compilando...

```
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 {
  static double valorM2 = 1500;
  ...
  static void areaCasa(float
           lateral, float cquarto) {
    float areaq;
    float areas;
    float areat;
    ...
  }
  ...
  static double valor(double area) {
    areat = 3;
    valorM2 = 5;
    return(valorM2*area);
  }
  ...
}
```

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

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 {
  static double valorM2 = 1500;

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

  public static void main(
                 String[] args) {
    double preco;

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

Testando os Parâmetros

- Considere o código ao lado
- Sua saída é

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

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

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

  public static void main(
                  String[] args) {
    double preco;

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

Testando os Parâmetros

- Considere o código ao lado
- Sua saída é

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

- O que aconteceu?

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

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

  public static void main(
                  String[] args) {
    double preco;

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

Testando os Parâmetros

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

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

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

  public static void main(
                  String[] args) {
    double preco;

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

Testando os Parâmetros

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

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

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

  public static void main(
                  String[] args) {
    double preco;

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

Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro

Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro
  - Esse valor é algo que definimos como, por exemplo, -1

Testando os Parâmetros

- SE o parâmetro for positivo ENTÃO calcule a área
- SENÃO, retorne um valor indicando erro
  - Esse valor é algo que definimos como, por exemplo, -1
- E como codificar isso?

Testando os Parâmetros

```
class AreaCasa {
  ...
  static double valor(double area)
  {
    if (area >= 0) {
      return(valorM2*area);
    }
    else {
      return(-1);
    }
  }
  public static void main(String[]
                              args) {
    ...
    preco = valor(-20);
    ...
  }
}
```

Testando os Parâmetros

```
class AreaCasa {
  ...
  static double valor(double area)
  {
    if (area >= 0) {
      return(valorM2*area);
    }
    else {
      return(-1);
    }
  }
  public static void main(String[]
                              args) {
    ...
    preco = valor(-20);
    ...
  }
}
```

- >=?

Testando os Parâmetros

```
class AreaCasa {
  ...
  static double valor(double area)
  {
    if (area >= 0) {
      return(valorM2*area);
    }
    else {
      return(-1);
    }
  }
  public static void main(String[]
                                args) {
    ...
    preco = valor(-20);
    ...
  }
}
```

- >=?
- Operador relacional:

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

Testando os Parâmetros

- O que o código no `if` diz?

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

Testando os Parâmetros

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

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

Testando os Parâmetros

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

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

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

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

Testando os Parâmetros

- 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

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

Testando os Parâmetros

- O código dentro do *else* é executado somente se a condição for falsa
- Se a condição for verdadeira, o código no *else* é ignorado

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

Testando os Parâmetros

- Será que tem como melhorar esse código?

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

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?

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

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

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

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

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

Testando os Parâmetros

- 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

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

Testando os Parâmetros

- 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

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

- Então vamos reduzir o código um pouco

Testando os Parâmetros

- 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

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

- Então vamos reduzir o código um pouco

Testando os Parâmetros

- 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

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

- Então vamos reduzir o código um pouco
- Mas ainda dá pra deixar mais enxuto...

Testando os Parâmetros

- Lembre que os {} denotam um bloco de comandos
- E que basta um ; para denotar o fim de um único comando

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

Testando os Parâmetros

Então, em vez de

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

Podemos fazer

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

Testando os Parâmetros

- E como usamos isso no `main`?

Testando os Parâmetros

- E como usamos isso no `main`?

```
public static void main(
                String[] args) {
  double preco;

  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

- E como usamos isso no `main`?
- O condicional evita que usemos um resultado inválido do método

```
public static void main(
                String[] args) {
  double preco;

  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

- E como usamos isso no `main`?
- O condicional evita que usemos um resultado inválido do método
- Evita inconsistências futuras

```
public static void main(
                String[] args) {
  double preco;

  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

- Vejamos novamente o que está dentro do *if*

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

Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?

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

Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?
  - Expressão que resulta em verdadeiro ou falso

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

Variáveis Booleanas

- Vejamos novamente o que está dentro do *if*
- O que significa condição?
  - Expressão que resulta em verdadeiro ou falso

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

- Usando esse conceito, haveria uma maneira alternativa (não necessariamente melhor) de escrever o main?

Variáveis Booleanas

```
public static void main(String[]
                                args) {
  double preco;
  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

```
public static void main(String[]
                                args) {
  double preco;
  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");
}
```

- Boolean $\rightarrow$ tipo de variável que armazena apenas dois valores:
  - Verdadeiro (true)
  - Falso (false) – padrão

Variáveis Booleanas

```
public static void main(String[]
                                args) {
  double preco;
  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");
}
```

- Boolean $\rightarrow$ tipo de variável que armazena apenas dois valores:
  - Verdadeiro (true)
  - Falso (false) – padrão
- Valores lógicos

Variáveis Booleanas

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

```
public static void main(String[]
                                    args) {
  double preco;
  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` recebe `true`
  - Senão, `valorOK` recebe `false`... mas ela já era `false`

```
public static void main(String[]
                                    args) {
  double preco;
  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

- Podemos então nos livrar dele sem problemas
  - Se *preco* ≥ 0, então `valorOK` recebe true
  - Senão, `valorOK` continua false

```
public static void main(String[]
                                    args) {
  double preco;
  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?

Variáveis Booleanas

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

Variáveis Booleanas

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

Variáveis Booleanas

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

Variáveis Booleanas

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`

```
public static void main(String[]
                                    args) {
  double preco;
  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

- Em vista disso, poderíamos reescrever o momento de atribuição de valor de `valorOK`

```
public static void main(String[]
                                    args) {
  double preco;
  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

- Em vista disso, 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;
  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

- Em vista disso, 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;
  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 `if`
- O que acontece no condicional?

```
public static void main(String[]
                                    args) {
  double preco;
  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 `if`
- O que acontece no condicional?
  - A expressão dentro dos parênteses é testada

```
public static void main(String[]
                                    args) {
  double preco;
  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 `if`
- O que acontece no 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;
  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 `if`
- O que acontece no 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;
  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

- Se não houver `else`, o programa continua normalmente

```
public static void main(String[]
                                    args) {
  double preco;
  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

- Não apenas expressões, mas também variáveis booleanas podem estar no `if`
- E são analisadas do mesmo modo

```
public static void main(String[]
                                    args) {
  double preco;
  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

- Se a variável for verdadeira, então o corpo do `if` será executado
- Se for falsa, será o corpo do `else` (se existir)

```
public static void main(String[]
                                    args) {
  double preco;
  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");
}
```

Visão Geral do Código

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

  static void areaCasa(float lateral,
                       float cquarto) {
    float areaq;
    float areas;
    float areat;

    System.out.println("...");
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."+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;
    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");
  }
}
```

Videoaula

https://www.youtube.com/watch?v=fOintAunVVg&t=11s

https://www.youtube.com/watch?v=oIIiNl9jknA&t=5s