



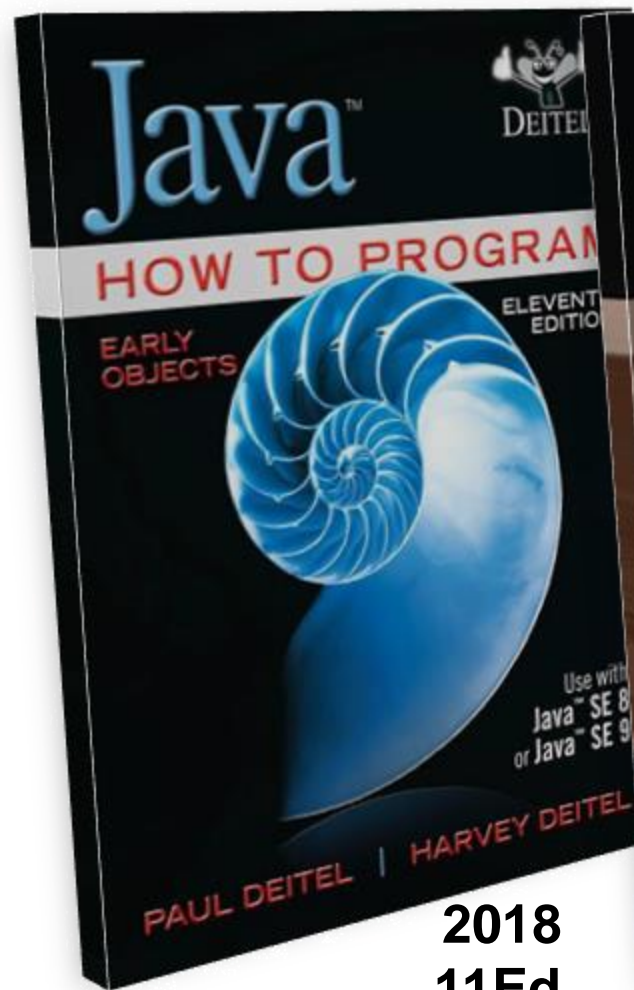
Classes e Objetos

Linguagem JAVA

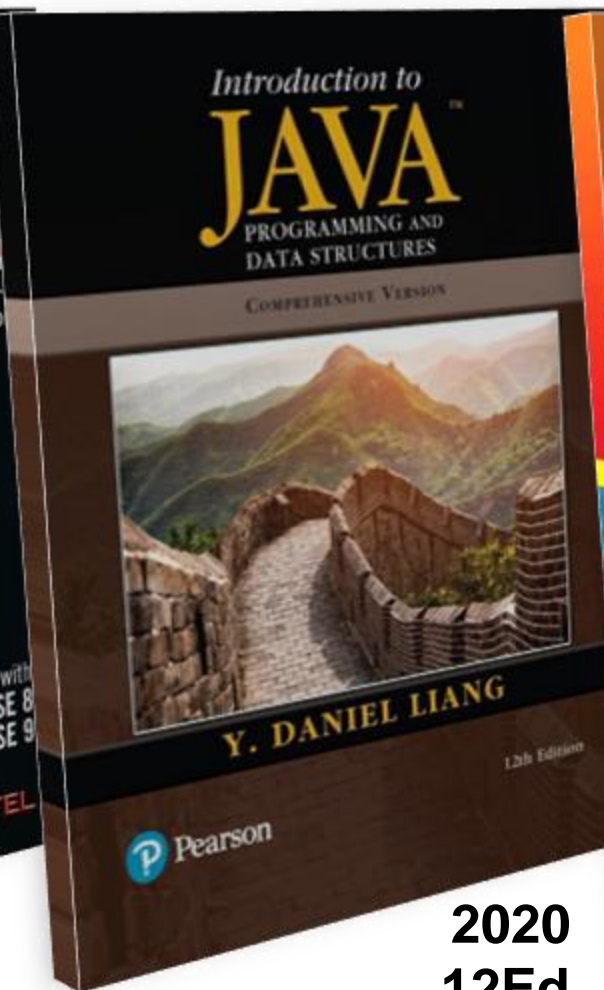


Prof. Ausberto S. Castro V.
ascv@uenf.br

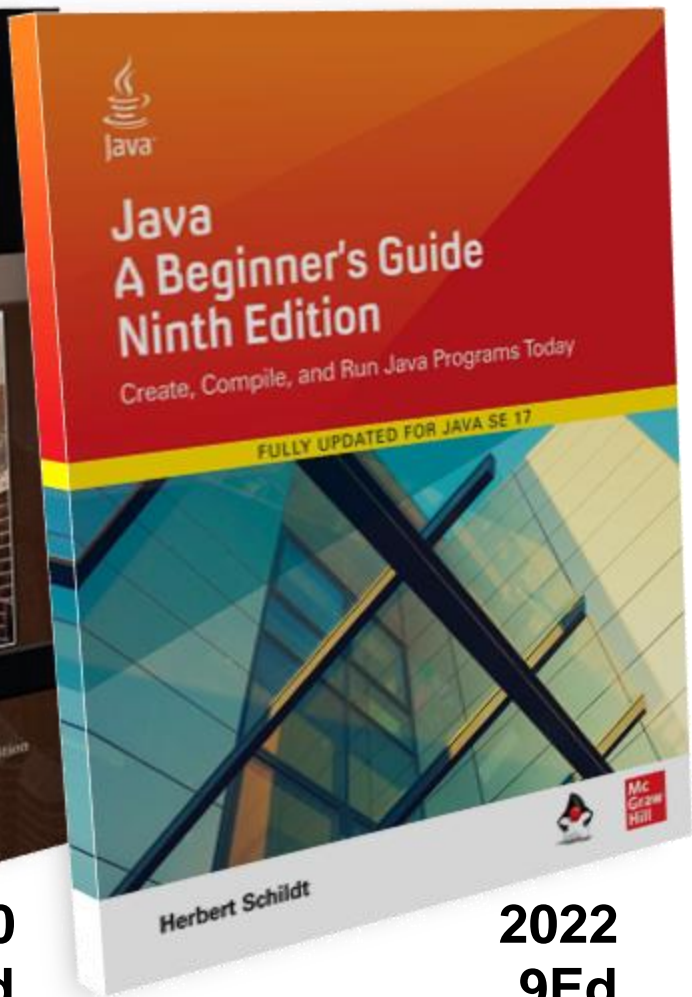
Bibliografia Básica



2018
11Ed



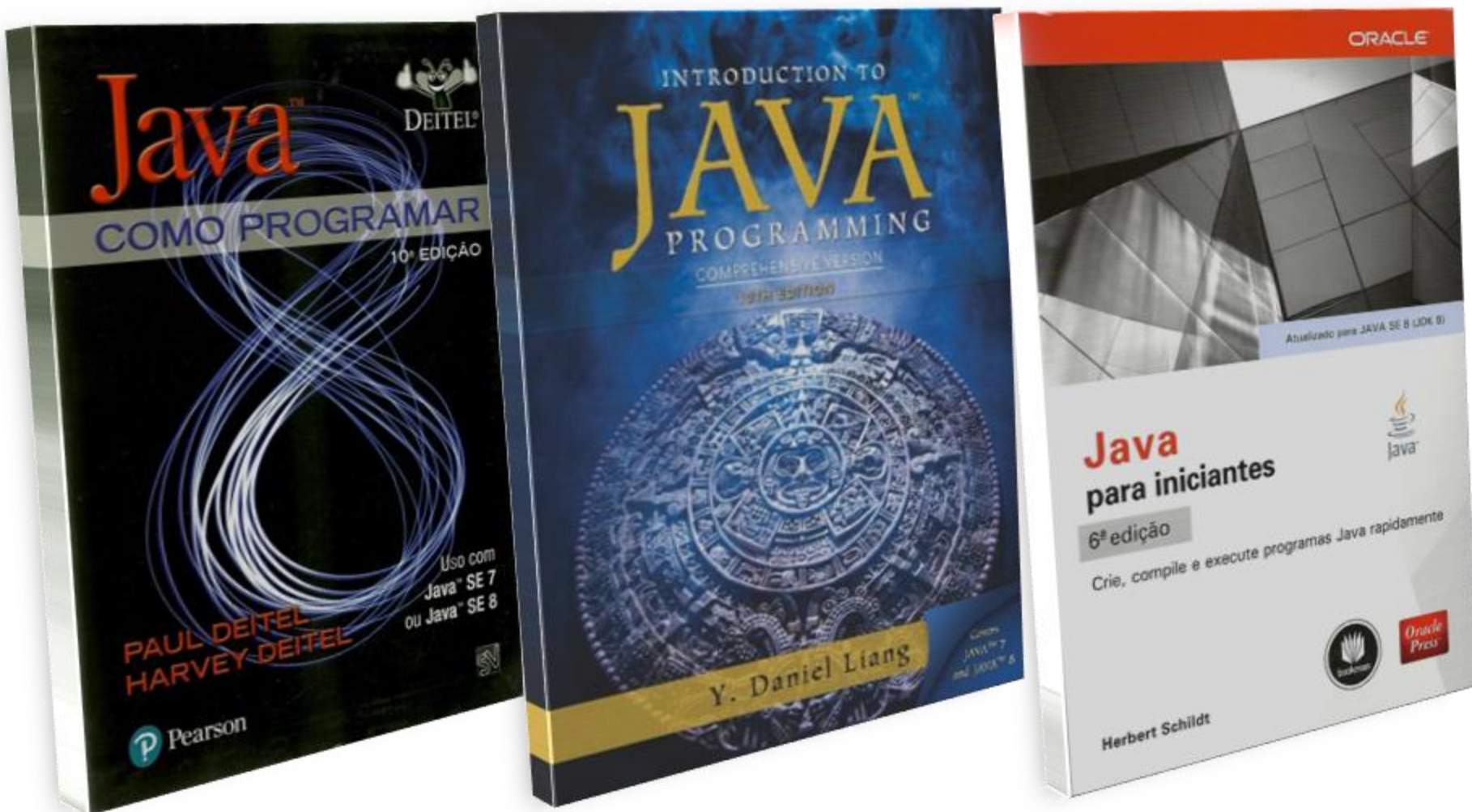
2020
12Ed



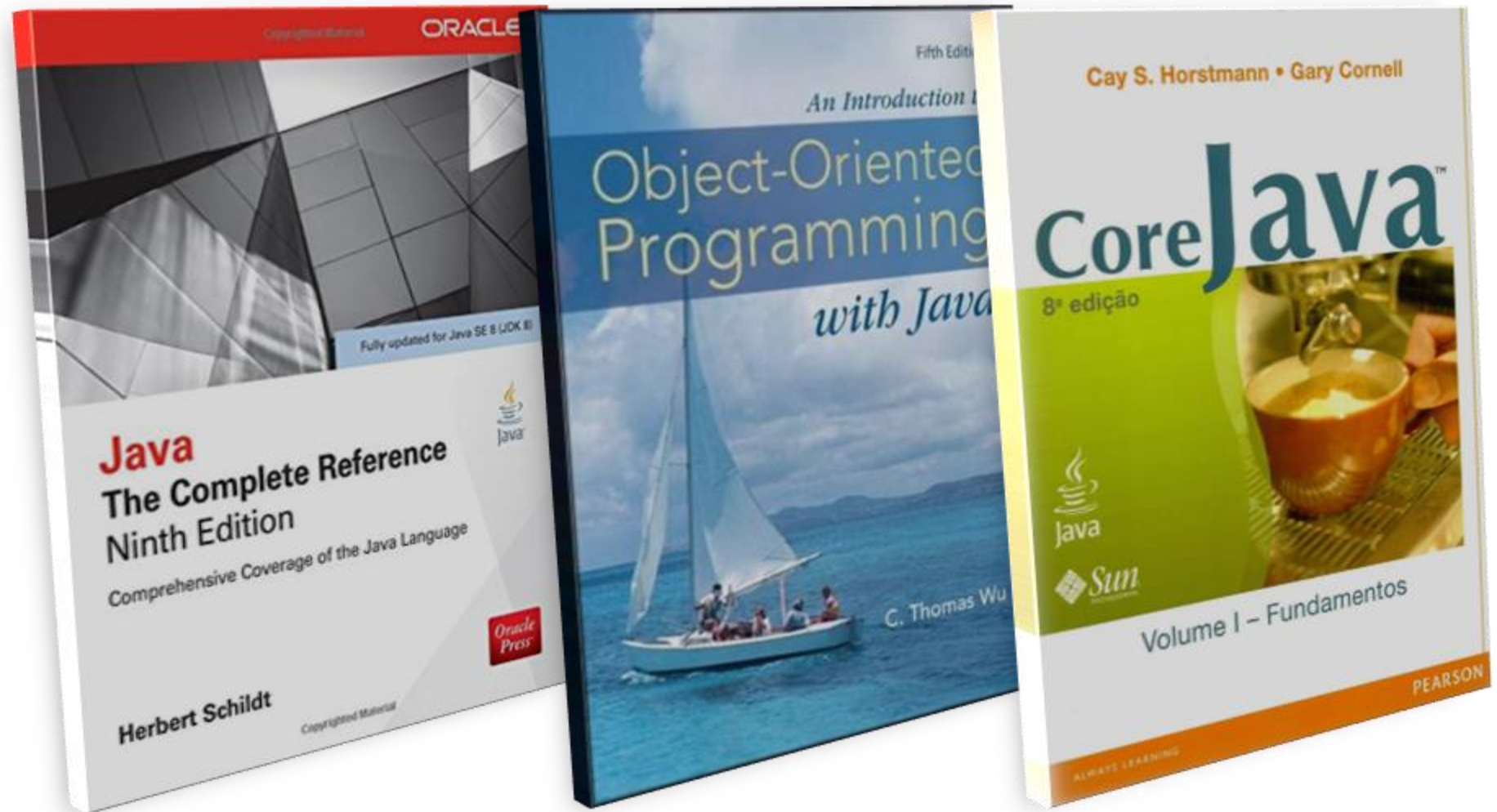
2022
9Ed

Versões anteriores em Português

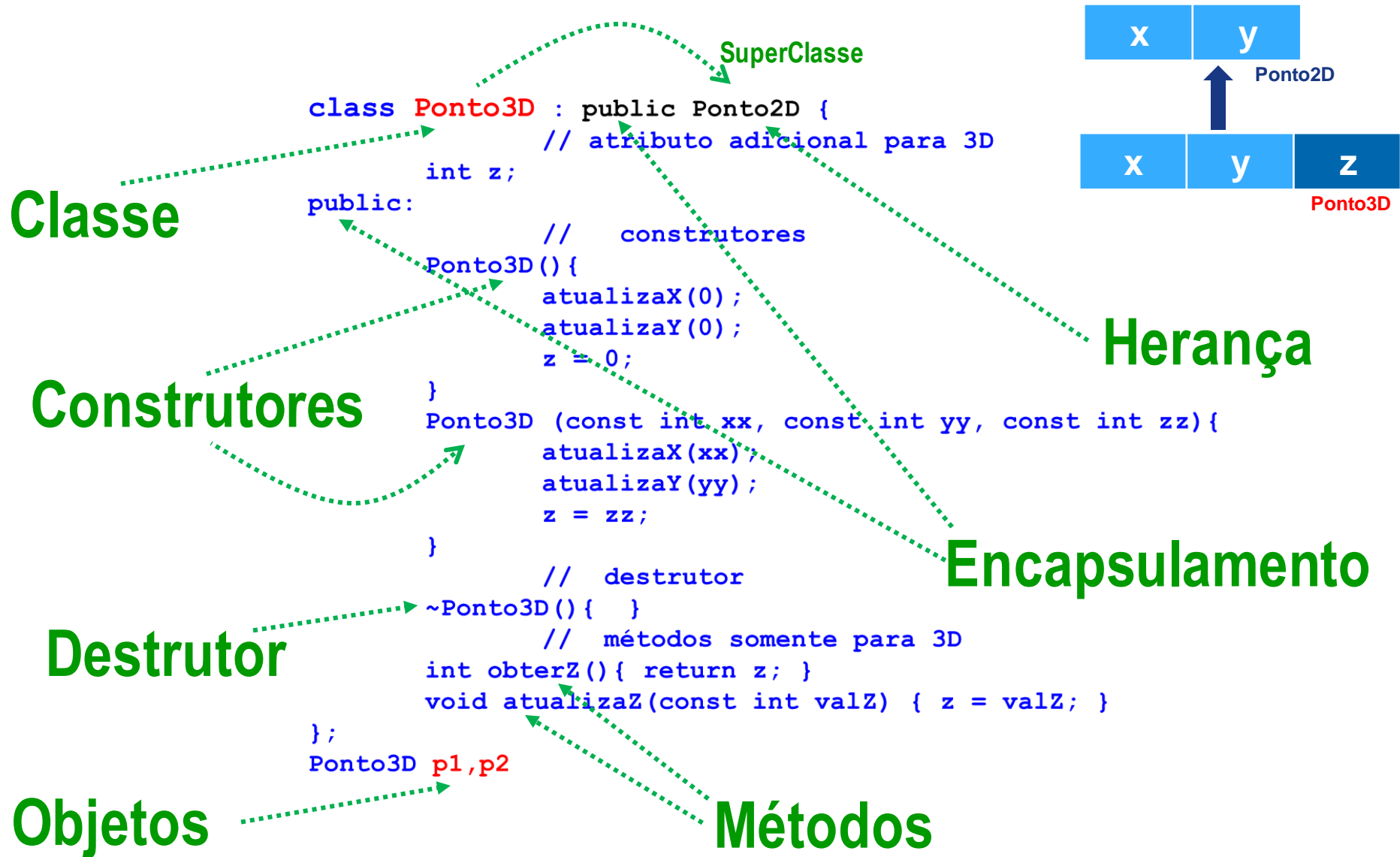
Bibliografia Básica Português



Bibliografia complementar



Paradigma Orientado a Objetos



Classes e Objetos

```
class nome da classe {  
    //declara variáveis da instancia  
    Tipo var1;  
    Tipo var2;  
    // ...  
    Tipo varN;  
  
    // declara métodos  
    Tipo metodo1 (parametros){  
        // corpo do método  
    }  
  
    Tipo metodo2 (parametros){  
        // corpo do método  
    }  
  
    // ...  
  
    Tipo metodoN (parametros){  
        // corpo do método  
    }  
  
}
```

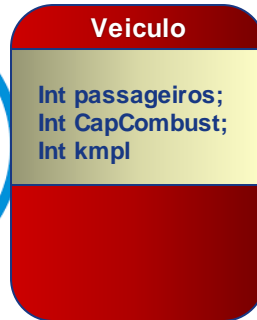
Forma Geral Simplificada
de uma classe **Java**



Classes: veiculo

CLASSE

```
10  L  */
11      class Veiculo {
12          int passageiros; // numero de passageiros
13          int CapCombust;  // capacidade de combustivel (litros)
14          int kmpl;        // consumo de combustivel Km por litro
15      }
16
```



```
Veiculo minivan = new Veiculo();
```

Classe

Objeto

Construtor

OBJETO

```
21
22      // atribui valores para campos de minivan
23      minivan.passageiros = 7;
24      minivan.CapCombust = 48;
25      minivan.kmpl = 12;
26
```



Classes: veiculo

The screenshot displays the Apache NetBeans IDE 11.0 interface. The main editor window shows the `Veiculo.java` file with the following code:

```
11  * @author Ausberto S. Castro Vera, 2019
12  * Um programa que usa a classe Veiculo.
13  */
14  class Veiculo {
15      int passageiros; // numero de passageiros
16      int CapCombust;  // capacidade de combustivel
17      int kmpl;        // consumo combustivel km/litro
18  }
19  // Esta classe declara UM objeto de tipo Veiculo.
20  class VeiculoDemo {
21      public static void main(String args[]) {
22          Veiculo minivan = new Veiculo();
23          int autonomia;
24
25          System.out.println("\nArquivo: Veiculo.java\n");
26
27          // Instanciando
28          minivan.passageiros = 7;
29          minivan.CapCombust = 16;
30          minivan.kmpl = 21;
31
32          // Calculos
33          autonomia = minivan.CapCombust * minivan.kmpl;
34
35          System.out.println("A minivan pode transportar " + minivan.passageiros +
36                          " passageiros com uma autonomia de " + autonomia + " kms");
37      }
38  }
```

Annotations and diagrams:

- A red dashed arrow points from the `VeiculoDemo` class name in the code to a callout box on the left. This box, titled **VeiculoDemo**, shows a yellow header and a red footer containing `void main()`.
- A red dashed arrow points from the `passageiros` attribute in the `Veiculo` class to a callout box on the right. This box, titled **Veiculo**, lists the attributes: `Int passageiros;`, `Int CapCombust;`, and `Int kmpl`.
- A blue circle with a car icon is positioned to the right of the `Veiculo` callout box.
- A red dashed arrow points from the `CapCombust` attribute in the `Veiculo` class to a callout box at the bottom right. This box, titled **Veiculo.java**, shows a side view of a silver minivan.

The bottom output window shows the execution results:

```
Run (ClasseAbstrata2) x Veiculo (run) x
run:
Arquivo: Veiculo.java

A minivan pode transportar 7 passageiros com uma autonomia de 336 kms

BUILD SUCCESSFUL (total time: 0 seconds)
```


Classes: veiculo

1 CLASSE

```
10  L  */
11      class Veiculo {
12          int passageiros; // numero de passageiros
13          int CapCombust;  // capacidade de combustivel (litros)
14          int kmpl;        // consumo de combustivel Km por litro
15      }
16
```



```
20      Veiculo minivan = new Veiculo();
21      Veiculo sportscar = new Veiculo();
22
```

2 OBJETOS

```
24
25      // atribui valores para campos de minivan
26      minivan.passageiros = 7;
27      minivan.CapCombust = 48;
28      minivan.kmpl = 12;
29
30      // atribui valores para campos de sportscar
31      sportscar.passageiros = 2;
32      sportscar.CapCombust = 45;
33      sportscar.kmpl = 16;
34
```



Classes: veiculo

The screenshot displays the Apache NetBeans IDE 11.0 environment. The main window is titled "DoisVeiculos - Apache NetBeans IDE 11.0". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar shows various icons for file operations, running, and debugging. The Projects window on the left shows a project named "DoisVeiculos" with a package "Source Packages" containing a class "VeiculoMetodo". The Source editor shows the code for "DoisVeiculos.java". The code defines a "Veiculo" class with attributes "passageiros", "CapCombust", and "kmpl". It also defines a "DoisVeiculos" class with a "main" method that creates "minivan" and "sportscar" objects and prints their details. The Output window at the bottom shows the execution results: "Minivan pode transportar 7 passageiros, em 576 kms" and "Carro esportivo pode transportar 2 passageiros, em 720 kms".

```
11
12  * @author Prof. Ausberto S. Castro Vera, 2019
13  */
14  class Veiculo {
15      int passageiros; // numero de passageiros
16      int CapCombust; // capacidade de combustivel (litros)
17      int kmpl;       // consumo de combustivel Km por litro
18  }
19
20  // Esta classe declara um objeto de tipo Veiculo.
21  class DoisVeiculos {
22      public static void main(String args[]) {
23          Veiculo minivan = new Veiculo();
24          Veiculo sportscar = new Veiculo();
25
26          int autonomial, autonomia2;
27
28          System.out.println("\nArquivo: DoisVeiculos.JAVA\n"); // Nome do arquivo JAVA
29
30          // atribui valores para campos de minivan
31          minivan.passageiros = 7;
32          minivan.CapCombust = 48;
33          minivan.kmpl = 12;
34
35          // atribui valores para campos de sportscar
36          sportscar.passageiros = 2;
37          sportscar.CapCombust = 45;
38          sportscar.kmpl = 16;
```

Output:

```
Minivan pode transportar 7 passageiros, em 576 kms
Carro esportivo pode transportar 2 passageiros, em 720 kms
```

DoisVeiculos.java

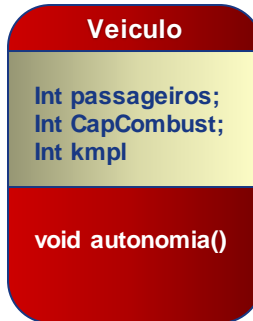
Classes e método: veículo

CLASSE

3 atributos

1 método

```
10  L  */
11      class Veiculo {
12          int passageiros; // numero de passageiros
13          int CapCombust;  // capacidade de combustivel (litros)
14          int kmpl;        // consumo de combustivel Km por litro
15
16          // metodo que exhibe autonomia
17          void autonomia() {
18              System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
19          }
20      }
```



```
24      Veiculo minivan = new Veiculo();
25      Veiculo sportscar = new Veiculo();
26
```

```
29      // atribui valores para campos de minivan
30      minivan.passageiros = 7;
31      minivan.CapCombust = 48;
32      minivan.kmpl = 12;
33
34      // atribui valores para campos de sportscar
35      sportscar.passageiros = 2;
36      sportscar.CapCombust = 45;
37      sportscar.kmpl = 16;
38
```

2

OBJETOS

Classes e método: veículo

The screenshot displays the Apache NetBeans IDE 11.0 interface. The main editor window shows the source code of `VeiculoMetodo.java`. The code defines a `Veiculo` class with three integer attributes: `passageiros` (number of passengers), `CapCombust` (fuel capacity in liters), and `kmpl` (fuel consumption in km per liter). A `autonomia()` method is also defined, which prints the total autonomy. The `VeiculoMetodo` class contains a `main` method that creates two `Veiculo` objects, `minivan` and `sportscar`, and initializes their attributes. The `minivan` object is initialized with 7 passengers, 48 liters of fuel capacity, and 12 kmpl. The `sportscar` object is initialized with default values (0 passengers, 0 liters, 0 kmpl). The `autonomia` method is called for both objects, and the results are printed to the console.

```
10  *
11  * @author Ausberto Castro Vera, 2019
12  */
13
14  class Veiculo {
15      int passageiros; // numero de passageiros
16      int CapCombust; // capacidade de combustivel (litros)
17      int kmpl;       // consumo de combustivel Km por litro
18
19      // metodo que exibe autonomia
20      void autonomia() {
21          System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
22      }
23  }
24  // Esta classe declara um objeto de tipo Veiculo.
25  class VeiculoMetodo {
26      public static void main(String args[]) {
27          Veiculo minivan = new Veiculo();
28          Veiculo sportscar = new Veiculo();
29
30          int autonomia, autonomia2;
31
32          System.out.println("\nArquivo: VeiculoMetodo.java\n");
33
34          // atribui valores para campos de minivan
35          minivan.passageiros = 7;
36          minivan.CapCombust = 48;
37          minivan.kmpl = 12;
```

The left sidebar shows the project structure, including the `VeiculoMetodo` project and its source packages. The bottom sidebar shows the `Veiculo` class and its methods, including `autonomia()` and `main(String[] args)`.

Método parametrizado: veiculo

CLASSE

3 atributos

2 métodos

```
10  */
11  class Veiculo {
12      int passageiros; // numero de passageiros
13      int CapCombust; // capacidade de combustivel (litros)
14      int kmpl;        // consumo de combustivel Km por litro
15
16      // metodo que exhibe autonomia
17      void autonomia() {
18          System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
19      }
20      // metodo que calcula combustivel necessario para certa distancia
21      double GasolinaNecessaria(int kilometros) {
22          return (double) kilometros/kmpl;
23      }
24  }
```

Veiculo

Int passageiros;
Int CapCombust;
Int kmpl

void autonomia()
double gasolinaNecessaria
(int km)

2

OBJETOS

```
35
36 // atribui valores para campos de minivan
37 minivan.passageiros = 7;
38 minivan.CapCombust = 48;
39 minivan.kmpl = 12;
40
41 // atribui valores para campos de sportscar
42 sportscar.passageiros = 2;
43 sportscar.CapCombust = 45;
44 sportscar.kmpl = 16;
45
46 // Calcula a a autonomia com tanque cheio: Minivan
47 litros = minivan.GasolinaNecessaria(distancia);
48 System.out.println("Para ir " + distancia + " kms, a minivan precisa " + litros +
49 " litros de gasolina. \n");
50
51 // Calcula a a autonomia com tanque cheio: Carro Esportivo
52 litros = sportscar.GasolinaNecessaria(distancia);
53 System.out.println("Para ir " + distancia + " kms, o carro esportivo precisa " +
54 litros + " litros de gasolina. \n");
55
```



Método parametrizado: veículo

The screenshot displays the Apache NetBeans IDE 11.0 interface. The main editor window shows the source code of `MetodoParamet.java`. The code defines a `Veiculo` class with attributes `passageiros`, `CapCombust`, and `kmpl`. It includes two methods: `autonomia()` which prints the total distance and fuel consumption, and `GasolinaNecessaria(int kilometros)` which calculates the required fuel. A red dashed arrow points from the `autonomia()` method call in the `main` method to its definition in the `Veiculo` class.

```
11  *
12  * @author Ausberto Castro Vera, 2019
13  */
14  class Veiculo {
15      int passageiros; // numero de passageiros
16      int CapCombust; // capacidade de combustivel (litros)
17      int kmpl;       // consumo de combustivel Km por litro
18
19      // metodo que exibe autonomia
20      void autonomia() {
21          System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
22      }
23
24      // metodo que calcula combustivel necessario para certa distancia
25      double GasolinaNecessaria(int kilometros){
26          return (double) kilometros/kmpl;
27      }
28  }
```

The output window, titled "Saída - MetodoParamet (run)", shows the execution results:

```
Arquivo: MetodoParamet.java
Prof. Ausberto Castro Vera - UENF 2018
Arquivo: MetodoParamet.JAVA
Para ir 252 kms, a minivan precisa 21.0 litros de gasolina.
Para ir 252 kms, o carro esportivo precisa 15.75 litros de gasolina.
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

A red box highlights the output window and the `main` method in the source code, which calls `veiculo.autonomia()` and `veiculo.GasolinaNecessaria(252)`.

Método construtor

```
9  * @author Prof. Ausberto S. Castro Vera
10 */
11 //----- CLASSE -----
12 class Veiculo {
13     int passageiros; // numero de passageiros
14     int CapCombust;  // capacidade de combustivel (litros)
15     int kmpl;        // consumo de combustivel Km por litro
16
17     //Método CONSTRUCTOR para Veiculo (com o mesmo nome da classe)
18     Veiculo(int p, int cc, int k){
19         passageiros = p;
20         CapCombust = cc;
21         kmpl = k;
22     }
23
24     // metodo que exibe autonomia
25     void autonomia(){
26         System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
27     }
28
29     // metodo que calcula combustivel necessario para certa distancia
30     double GasolinaNecessaria(int kilometros){
31         return (double) kilometros/kmpl;
32     }
33 }
34 //-----
```

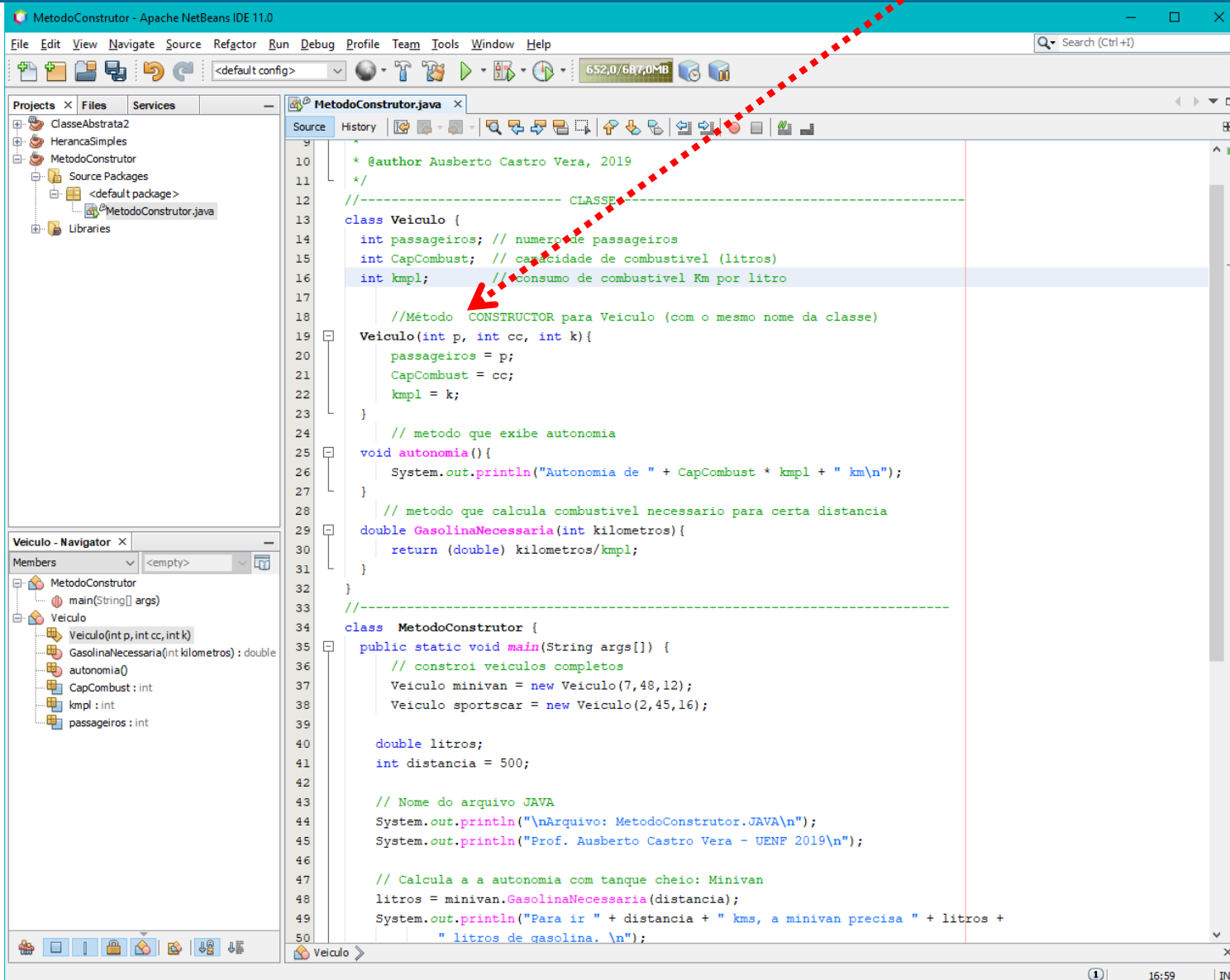
**Método
construtor**

Um *construtor* inicializa um objeto quando este é criado. Tem o *mesmo nome* da sua classe. Não tem *tipo de retorno* explícito. Se utiliza para fornecer valores iniciais para as *variáveis de instância* definidas pela classe

**Objetos construídos
utilizando o
método construtor**

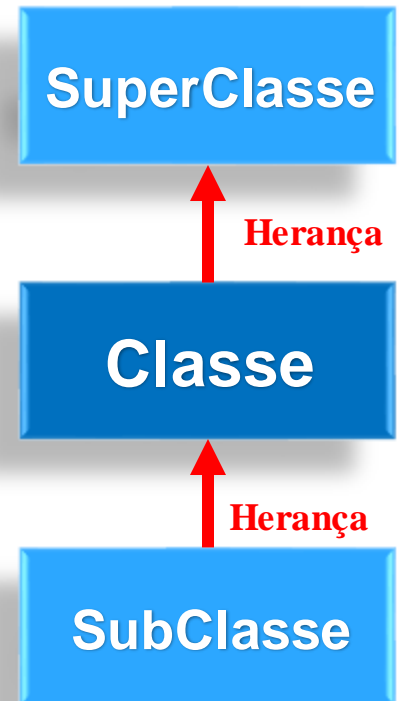
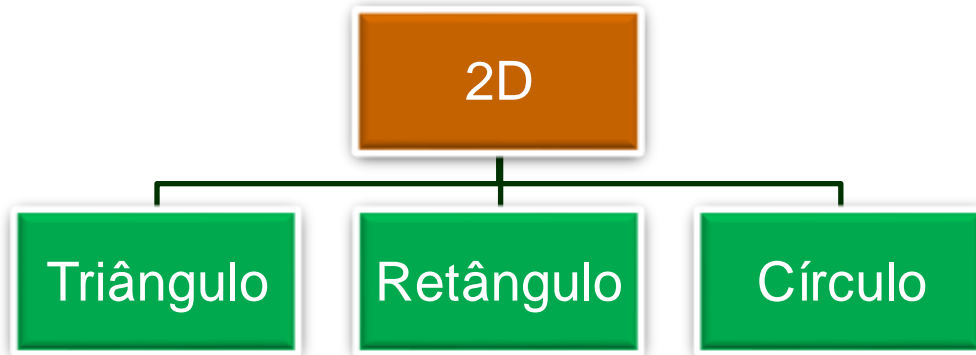
```
37 // constroi veiculos completos
38 Veiculo minivan = new Veiculo(7,48,12);
39 Veiculo sportscar = new Veiculo(2,45,16);
40
```

Método construtor



Herança

- É um dos *princípios básicos* da Programação Orientada a Objetos
- Permite a criação de *classificações hierárquicas*
- Permite criar uma *classe geral* que defina características comuns a um conjunto de itens relacionados
- Permite que uma *classe*, possa ser *herdada* por outras classes mais específicas, cada uma adicionando suas características exclusivas
- A classe que é herdada se chama *superclasse*. A classe que herda se chama *subclasse*



Herança Simples

HerancaSimples - Apache NetBeans IDE 11.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+F)

Projects Files Services

HerancaSimples.java

```
11  * @author Prof. Ausberto S. Castro Vera
12  */
13  // Uma hierarquia simples.
14  // ----- Classe de Objetos 2D -----
15  @
16  class Forma2D {
17      double largura;
18      double altura;
19
20      void MostrarDim() {
21          System.out.println("Largura = " +
22                          largura + " e a altura = " + altura);
23      }
24
25  // ----- Uma subclasse 2D para Triangulos -----
26  class Triangulo extends Forma2D {
27      String estilo;
28
29      double area() {
30          return largura * altura / 2;
31      }
32
33      void MostraEstilo() {
34          System.out.println("O triângulo é " + estilo);
35      }
36  }
37
38  // -----
39  public class HerancaSimples {
40      public static void main(String args[]) {
41          Triangulo t1 = new Triangulo();
42          Triangulo t2 = new Triangulo();
43
44          System.out.println("Prof. Ausberto Castro Vera - UENF 2019\nHERANÇA Simples");
45
46          t1.largura = 4.0;
47          t1.altura = 3.0;
48          t1.estilo = "Preenchido";
49
50          t2.largura = 6.0;
51          t2.altura = 9.0;
```

HerancaSimples.java - Navigator

Members

Forma2D

- MostrarDim()
- altura : double
- largura : double

HerancaSimples

- main(String[] args)

Triangulo :: Forma2D

- MostraEstilo()
- area() : double
- estilo : String

Forma2D

- largura : double
- altura : double
- +MostrarDim()

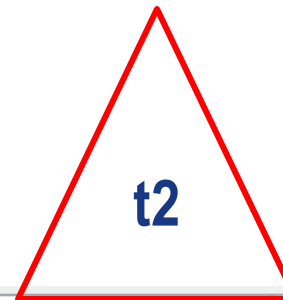
Triangulo

- estilo : string
- +area() : double
- +MostrarEstilo() : void

HerancaSimples.java

Herança Simples

```
//-----
35
36
37 class Formas {
38     public static void main(String args[]) {
39         Triangulo t1 = new Triangulo();
40         Triangulo t2 = new Triangulo();
41
42         System.out.println("Prof. Ausberto Castro Vera - UENF 2016\nHERANÇA - Formas Geometricas\n");
43
44         t1.largura = 4.0;
45         t1.altura = 3.0;
46         t1.estilo = "Preenchido";
47
48         t2.largura = 6.0;
49         t2.altura = 9.0;
50         t2.estilo = "Contorno";
51
52         System.out.println("Info para a forma t1: ");
53         t1.MostraEstilo();
54         t1.MostrarDim();
55         System.out.println("A AREA é " + t1.area());
56
57         System.out.println();
58
59         System.out.println("Info para a forma t2: ");
60         t2.MostraEstilo();
61         t2.MostrarDim();
62         System.out.println("A AREA é " + t2.area());
63     }
64 }
```



```
Saída - HerancaSimples (run) X
run:
Prof. Ausberto Castro Vera - UENF 2018
HERANÇA Simples - Formas Geometricas

Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0
A AREA é 27.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Herança Simples

HerancaSimples - Apache NetBeans IDE 11.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Projects Files Services

- ClasseAbstrata2
- HerancaSimples
 - Source Packages
 - HerancaSimples
 - HerancaSimples.java
 - Libraries

HerancaSimples.java - Navigator

Members

- Forma2D
 - MostrarDim()
 - altura : double
 - largura : double
- HerancaSimples
 - main(String[] args)
- Triangulo :: Forma2D
 - MostraEstilo()
 - area(): double
 - estilo : String

```
37 //-----
38
39 public class HerancaSimples {
40     public static void main(String args[]) {
41         Triangulo t1 = new Triangulo();
42         Triangulo t2 = new Triangulo();
43
44         System.out.println("Prof. Ausberto Castro Vera - UENF 2019\nHERANÇA Simples - Formas Geometricas\n");
45
46         t1.largura = 4.0;
47         t1.altura = 3.0;
48         t1.estilo = "Preenchido";
49
50         t2.largura = 6.0;
51         t2.altura = 9.0;
52         t2.estilo = "Contorno";
53
54         System.out.println("Info para a forma t1: ");
55         t1.MostraEstilo();
56         t1.MostrarDim();
57         System.out.println("A AREA é " + t1.area());
58
59         System.out.println();
60
61         System.out.println("Info para a forma t2: ");
62         t2.MostraEstilo();
63         t2.MostrarDim();
64         System.out.println("A AREA é " + t2.area());
65     }
66 }
```

Outp...

Run (ClasseAbstrata2) x HerancaSimples (run) x

Prof. Ausberto Castro Vera - UENF 2019
HERANÇA Simples - Formas Geometricas

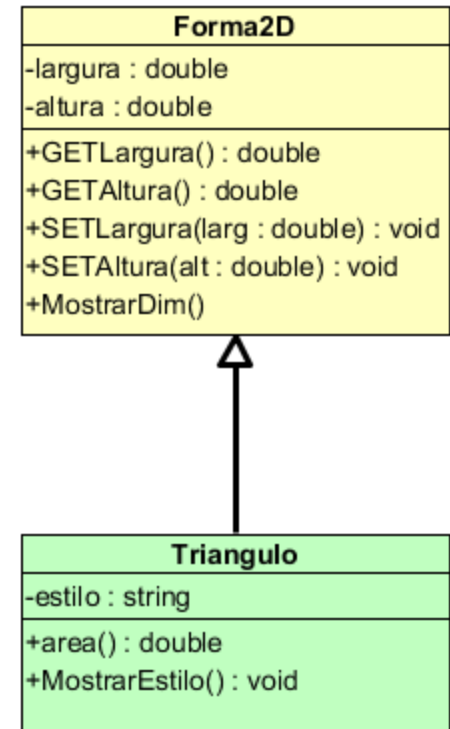
Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0

Acesso a membros e herança

```
9  * @author Prof. Ausberto S. Castro Vera
10 */
11 class Forma2D {
12     private double largura;
13     private double altura;
14
15     double GETLargura(){return largura; }
16     double GETAltura(){ return altura; }
17     void SETLargura(double larg){ largura = larg; }
18     void SETAltura(double alt){ altura = alt; }
19
20     void MostrarDim() {
21         System.out.println("Largura = " +
22             largura + " e a altura = " + altura);
23     }
24 }
25
26 // ----- Uma subclasse 2D para Traingulos -----
27 class Triangulo extends Forma2D {
28     String estilo;
29
30     double area() {
31         return GETLargura() * GETAltura() / 2;
32     }
33
34     void MostraEstilo() {
35         System.out.println("O triângulo é " + estilo);
36     }
37 }
38 //-----
```

**Métodos
acessadores**



Formas2D.java

Acesso a membros e herança

```
40 class Formas2 {
41     public static void main(String args[]) {
42         Triangulo t1 = new Triangulo();
43         Triangulo t2 = new Triangulo();
44
45         System.out.println("Prof. Ausberto Castro Vera - UENF 2016\nHERANÇA - "+
46             "Uso de Metodos para Acesso\n");
47
48         t1.SETLargura(4.0);
49         t1.SETAltura(3.0);
50         t1.estilo = "Preenchido";
51
52         t2.SETLargura(6.0);
53         t2.SETAltura(9.0);
54         t2.estilo = "Contorno";
55
56         System.out.println("Info para a forma t1: ");
57         t1.MostraEstilo();
58         t1.MostrarDim();
59         System.out.println("A AREA é " + t1.area());
60
61         System.out.println();
62
63         System.out.println("Info para a forma t2: ");
64         t2.MostraEstilo();
65         t2.MostrarDim();
66         System.out.println("A AREA é " + t2.area());
67     }
68 }
```

Saida - ASCVJavaHeranca02 (run) X

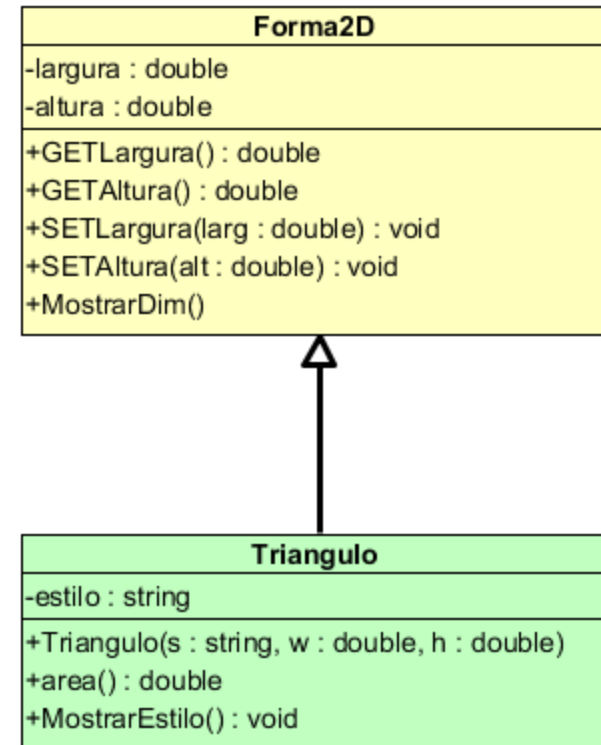
```
run:
Prof. Ausberto Castro Vera - UENF 2016
HERANÇA - Uso de Metodos para Acesso

Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0
A AREA é 27.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Construtores e Herança

```
9  * @author Prof. Ausberto S. Castro Vera
10 */
11 class Forma2D {
12     private double largura;
13     private double altura;
14
15     double GETLargura(){return largura; }
16     double GETAltura(){ return altura; }
17     void SETLargura(double larg){ largura = larg; }
18     void SETAltura(double alt){ altura = alt; }
19
20     void MostrarDim() {
21         System.out.println("Largura = " +
22             largura + " e a altura = " + altura);
23     }
24 }
25 // ----- Uma subclasse 2D para Traingulos -----
26 class Triangulo extends Forma2D {
27     private String estilo;
28
29     // Constructor
30     Triangulo(String s, double w, double h){
31         SETLargura(w);
32         SETAltura(h);
33         estilo = s;
34     }
35
36     double area() {
37         return GETLargura() * GETAltura() / 2;
38     }
39
40     void MostraEstilo() {
41         System.out.println("O triângulo é " + estilo);
42     }
43 }
44 //-----
```



Construtores e Herança

```
25 // ----- Uma subclasse 2D para Triangulos -----
26 class Triangulo extends Forma2D {
27     private String estilo;
28
29     // Constructor
30     Triangulo(String s, double w, double h){
31         SETLargura(w);
32         SETAltura(h);
33         estilo = s;
34     }
35     double area() {
36         return GETLargura() * GETAltura() / 2;
37     }
38     void MostraEstilo() {
39         System.out.println("O triângulo é " + estilo);
40     }
41 }
42 //-----
43 class Formas3 {
44     public static void main(String args[]) {
45         Triangulo t1 = new Triangulo("Preenchido",4.0,3.0);
46         Triangulo t2 = new Triangulo("Contorno",6.0,9.0);
47
48         System.out.println("Prof. Ausberto Castro Vera - UENF 2016\nHERANÇA - "+
49             "Construtores e Herança\n");
50
51         System.out.println("Info para a forma t1: ");
52         t1.MostraEstilo();
53         t1.MostrarDim();
54         System.out.println("A AREA é " + t1.area());
55
56         System.out.println();
57
58         System.out.println("Info para a forma t2: ");
59         t2.MostraEstilo();
60         t2.MostrarDim();
61         System.out.println("A AREA é " + t2.area()+"\n");
62     }
63 }
```

Saida - ASCVJavaHeranca03 (run) X

```
run:
Prof. Ausberto Castro Vera - UENF 2016
HERANÇA - Construtores e Herança

Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

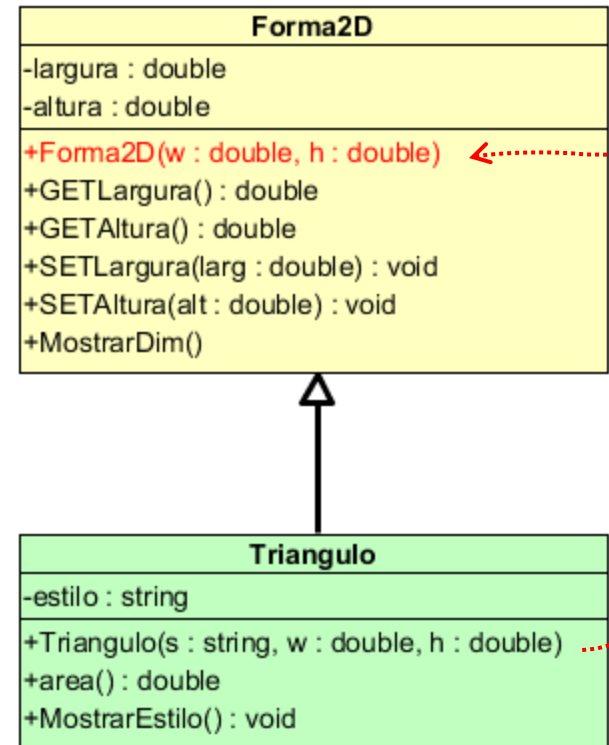
Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0
A AREA é 27.0

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Chamando construtores de Superclasse

Utilizando a instrução **super**

```
9  * @author Prof. Ausberto S. Castro Vera
10 */
11
12 class Forma2D {
13     private double largura;
14     private double altura;
15
16     // constructor parametrizado
17     Forma2D(double w, double h){
18         largura = w;
19         altura = h;
20     }
21
22     double GETLargura(){return largura; }
23     double GETAltura(){ return altura; }
24     void SETLargura(double larg){ largura = larg; }
25     void SETAltura(double alt){ altura = alt; }
26
27     void MostrarDim() {
28         System.out.println("Largura = " +
29             largura + " e a altura = " + altura);
30     }
31 }
32
33 // ----- Uma subclasse 2D para Traingulos -----
34 class Triangulo extends Forma2D {
35     private String estilo;
36
37     // Constructor
38     Triangulo(String s, double w, double h){
39         super(w,h); // chama construtor Forma2D parametrizado de SUPERclasse
40         estilo = s;
41     }
42
43     double area() {
44         return GETLargura() * GETAltura() / 2;
45     }
46
47     void MostraEstilo() {
48         System.out.println("O triângulo é " + estilo);
49     }
50 }
51
52 //-----
```



Uma *subclasse* pode chamar um construtor definido por sua *superclasse* usando a forma de **super**:
super (lista-de-parametros)

Chamando construtores de Superclasse

The screenshot displays the NetBeans IDE environment. The main editor shows the source code for `Formas4.java`, which includes a `Triangulo` class and a `Formas4` class. The `Triangulo` class has a constructor that calls `super(w, h)` to invoke the constructor of the superclass `Forma2D`. The `Formas4` class contains a `main` method that creates two `Triangulo` objects, `t1` and `t2`, and calls their `MostraEstilo()` methods. The `MostraEstilo()` method in `Triangulo` prints the triangle's style, width, height, and area.

```
34 // Construtor
35 Triangulo(String s, double w, double h){
36     super(w,h); // chama construtor Forma2D parametrizado de SUPERclasse
37     estilo = s;
38 }
39 double area() {
40     return GETLargura() * GETAltura() / 2;
41 }
42 void MostraEstilo() {
43     System.out.println("O triângulo é " + estilo);
44 }
45 //
46 class Formas4 {
47     public static void main(String args[]) {
48         Triangulo t1 = new Triangulo("Preenchido",4.0,3.0);
49         Triangulo t2 = new Triangulo("Contorno",6.0,9.0);
50
51         System.out.println("Prof. Ausberto Castro Vera - UENF 2016\nHERANÇA - Chamando construtores de SUPERclasse\n");
52
53         System.out.println("Info para a forma t1: ");
54         t1.MostraEstilo();
55         t1.MostrarDim();
56         System.out.println("A AREA é " + t1.area());
57
58         System.out.println();
59
60         System.out.println("Info para a forma t2: ");
61         t2.MostraEstilo();
62     }
63 }
```

The output window shows the results of running the program. It displays the professor's name and the course, followed by information for two triangles: `t1` (Preenchido) and `t2` (Contorno). The output also shows the area for each triangle and a success message.

Saída - ASCVJavaheranca04 (run) X

```
run:
Prof. Ausberto Castro Vera - UENF 2016
HERANÇA - Chamando construtores de SUPERclasse

Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0
A AREA é 27.0

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Saída - ASCVJavaheranca04 (run) X

```
Prof. Ausberto Castro Vera - UENF 2016
HERANÇA - Chamando construtores de SUPERclasse

Info para a forma t1:
O triângulo é Preenchido
Largura = 4.0 e a altura = 3.0
A AREA é 6.0

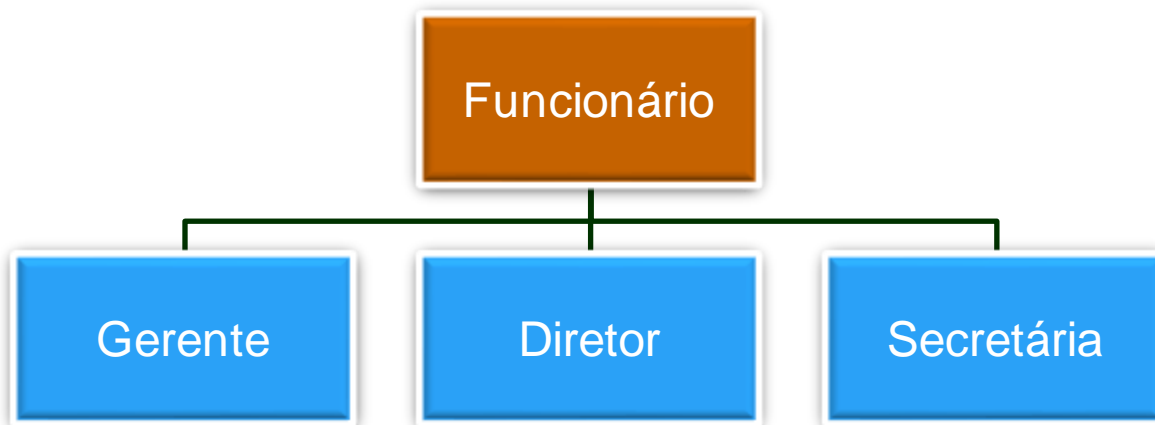
Info para a forma t2:
O triângulo é Contorno
Largura = 6.0 e a altura = 9.0
A AREA é 27.0

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

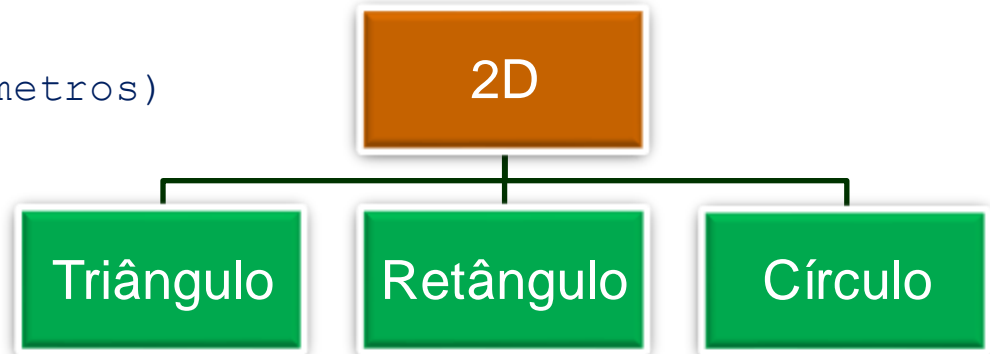
ConstrutSuperClasse.java

Classe Abstrata

- É uma classe que apenas idealiza um tipo, define apenas um rascunho.
- Uma forma (classe) generalizada para ser compartilhada por todas suas subclasses, deixando que cada subclasse insira os detalhes

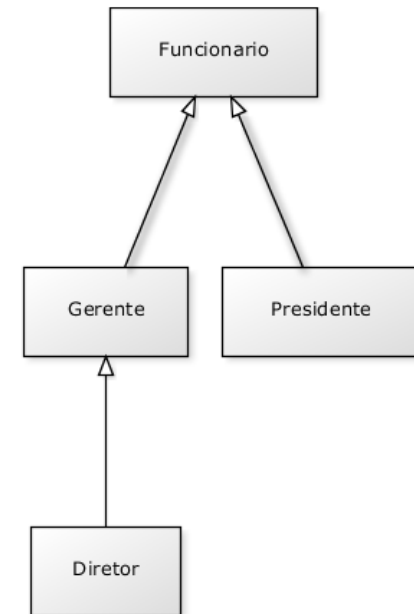


`abstract tipo nome(lista-parametros)`



Classe Abstrata

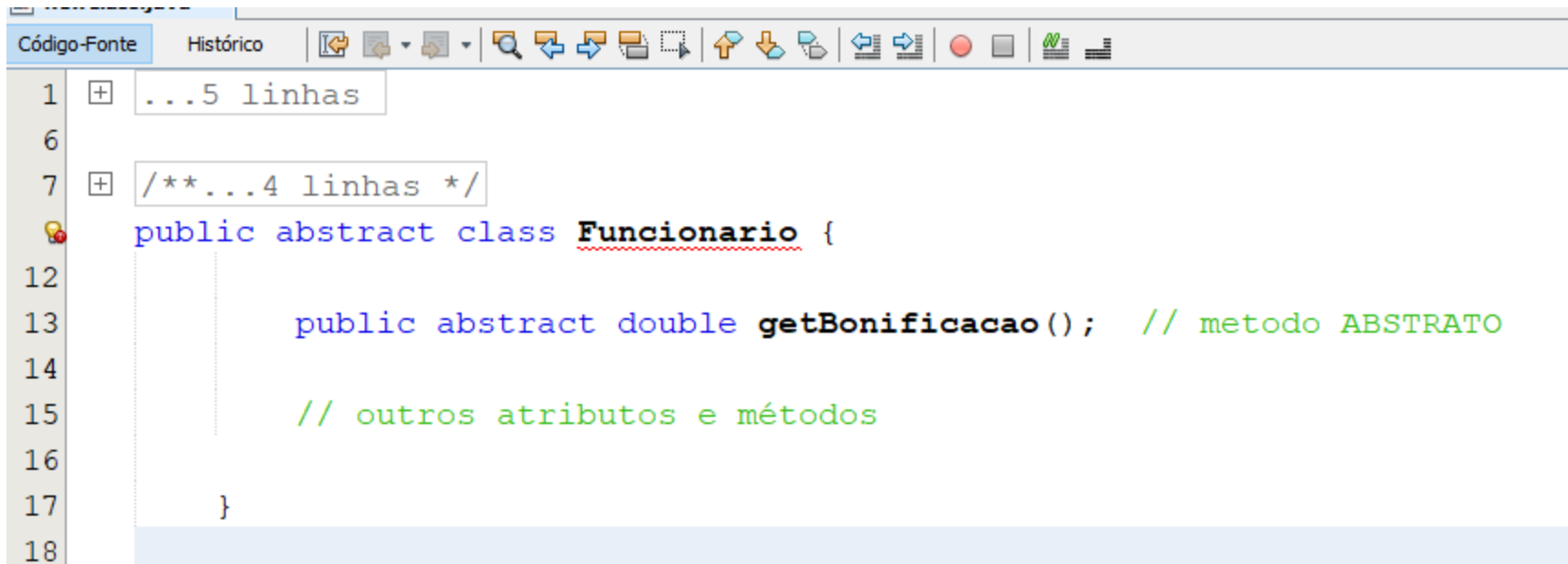
```
testarPessoa.java x Funcionario.java x
Source History
9  /*
10  * @author Prof. Ausberto S. Castro Vera (ascv@uenf.br)
11  */
12
13  import java.time.*;
14
15  public class Funcionario extends Pessoa
16  {
17      private double salario;
18      private LocalDate DiaContrato;
19      //-----
20      public Funcionario(String n, double sal, int ano, int mes, int dia)
21      {
22          super(n);
23          this.salario = sal;
24          DiaContrato = LocalDate.of(
25
```



```
Código-Fonte Histórico
1  ...5 linhas
6
7  /**...4 linhas */
8  public class Gerente extends Funcionario {
9
10
11
12
13      public double getBonificacao() {
14          return this.salario * 1.4 + 1000;
15      }
16  }
17
```


Método abstrato

- Em uma classe abstrata, podemos escrever que determinado *método* será *sempre* escrito pelas classes filhas
- Basta escrever a palavra chave **abstract** na assinatura do mesmo e colocar um ponto e vírgula em vez de abre e fecha chaves!



```
1  ...5 linhas
6
7  /**...4 linhas */
   public abstract class Funcionario {
12
13       public abstract double getBonificacao(); // metodo ABSTRATO
14
15       // outros atributos e métodos
16
17   }
18
```

Classe Abstrata

The screenshot displays the Apache NetBeans IDE 11.0 interface. The main window shows the source code of `ClasseAbstrata.java`. The code defines an abstract class `Forma2D` with attributes `Largura`, `Altura`, and `Nome`, and methods for calculating area and displaying dimensions. It also includes two subclasses, `Triangulo` and `Retangulo`, which inherit from `Forma2D` and implement the `area()` method.

The **Output - ClasseAbstrata (run)** window shows the execution results:

```
run:
Prof. Ausberto Castro - Arquivo: ClasseAbstrata.java

O objeto e' um Triangulo
A AREA e' 48.0

O objeto e' um Retangulo
A AREA e' 100.0

O objeto e' um Retangulo
A AREA e' 40.0

O objeto e' um Triangulo
A AREA e' 24.5

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```



Prof. Dr. Ausberto S. Castro Vera
Ciência da Computação
UENF-CCT-LCMAT
Campos, RJ

ausberto.castro@gmail.com
ascv@uenf.br

Facebook

facebook

LinkedIn

Linked in

WordPress



Lattes

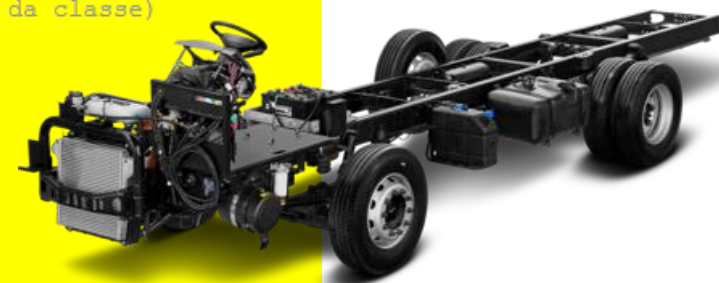
Plataforma Lattes

ti
especialistas
EM SISTEMAS DE INFORMAÇÃO

 **iMasters**

Classe Veiculo

```
9      * @author Prof. Ausberto S. Castro Vera
10     */
11     //----- CLASSE -----
12     © class Veiculo {
13         int passageiros; // numero de passageiros
14         int CapCombust; // capacidade de combustivel (litros)
15         int kmpl;        // consumo de combustivel Km por litro
16
17         //Método CONSTRUCTOR para Veiculo (com o mesmo nome da classe)
18         Veiculo(int p, int cc, int k){
19             passageiros = p;
20             CapCombust = cc;
21             kmpl = k;
22         }
23         // metodo que exibe autonomia
24         void autonomia(){
25             System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
26         }
27         // metodo que calcula combustivel necessario para certa distancia
28         double GasolinaNecessaria(int kilometros){
29             return (double) kilometros/kmpl;
30         }
31     //-----
32     // Métodos de acesso.
33     int GETPassageiros() { return passageiros; }
34     void SETPPassageiros(int p) { passageiros = p; }
35     int GETCapCombust() { return CapCombust; }
36     void SETCapCombust(int f) { CapCombust = f; }
37     int GETkmpl() { return kmpl; }
38     void SETkmpl(int m) { kmpl = m; }
39 }
```



Classe Caminhão



```
41 // Extende Veiculo para a especialização Caminhao.
42 class Caminhao extends Veiculo {
43     private int CapCarga; // capacidade carga
44     // Construtor para Caminhao.
45     Caminhao(int p, int f, int m, int c) {
46         // Inicializa Veiculo usando variaveis do construtor Veiculo
47         super(p, f, m);
48
49         CapCarga = c;
50     }
51     // métodos de acesso para CapCarga.
52     int GETCarga() { return CapCarga; }
53     void PUTCarga(int c) { CapCarga = c; }
54 }
```

Objetos Caminhão



```
55
56 class CaminhaoDemo {
57     public static void main(String args[]) {
58
59         // construct some trucks
60         Caminhao semi = new Caminhao(2, 200, 7, 44000);
61         Caminhao pickup = new Caminhao(3, 28, 15, 2000);
62         double litros;
63         int dist = 300;
64
65         litros = semi.GasolinaNecessaria(dist);
66
67         System.out.println("O caminhao Semi pode transportar " + semi.GETCarga() +
68                             " quilos.");
69         System.out.println("Para ir " + dist + " km o caminhão semi precisa " +
70                             litros + " litros de gasolina.\n");
71
72         litros = pickup.GasolinaNecessaria(dist);
73
74         System.out.println("A Pickup pode transportar " + pickup.GETCarga() +
75                             " quilos.");
76         System.out.println("Para ir " + dist + " kms a pickup precisa " +
77                             litros + " litros de gasolina.");
78     }
79 }
80
```


Objetos Caminhão

ASCVJavaHeranca05 - NetBeans IDE 8.2

Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda

Pesquisar (Ctrl+I)

Projetos x Arquivos Serviços

Código-Fonte Histórico

```
49     CapCarga = c;
50 }
51 // métodos de acesso para CapCarga.
52 int GETCarga() { return CapCarga; }
53 void PUTCarga(int c) { CapCarga = c; }
54 }
55
56 class CaminhaoDemo {
57     public static void main(String args[]) {
58
59         // construct some trucks
60         Caminhao semi = new Caminhao(2, 200, 7, 44000);
61         Caminhao pickup = new Caminhao(3, 28, 15, 2000);
62         double litros;
63         int dist = 350;
64
65         litros = semi.GasolinaNecessaria(dist);
66
67         System.out.println("O caminhao Semi pode transportar " + semi.GETCarga() +
68                             " quilos.");
69         System.out.println("Para ir " + dist + " km o caminhão semi precisa " +
70                             litros + " litros de gasolina.\n");
71
72         litros = pickup.GasolinaNecessaria(dist);
73
74         System.out.println("A Pickup pode transportar " + pickup.GETCarga() +
75                             " quilos.");
76         System.out.println("Para ir " + dist + " kms a pickup precisa " +
77                             litros + " litros de gasolina.");
78     }
79 }
80
```

Navegador

Membros

- Caminhao :: Veiculo
 - Caminhao(int p, int f, int m, int c)
 - GETCarga() : int
 - PUTCarga(int c)
 - CapCarga : int
- CaminhaoDemo
 - main(String[] args)
- Veiculo
 - Veiculo(int p, int cc, int k)
 - GETCapCombust() : int
 - GETPassageiros() : int
 - GETKmpl() : int
 - GasolinaNecessaria(int kilometros) : double
 - SETCapCombust(int f)
 - SETPPassageiros(int p)
 - SETKmpl(int m)
 - autonomia()
 - CapCombust : int
 - kmpl : int
 - passageiros : int

Saída - ASCVJavaHeranca05 (run)

```
run:
O caminhao Semi pode transportar 44000 quilos.
Para ir 350 km o caminhão semi precisa 50.0 litros de gasolina.

A Pickup pode transportar 2000 quilos.
Para ir 350 kms a pickup precisa 23.33333333333332 litros de gasolina.
CONSTRUIDO COM SUCESSO (tempo total: 0 segundos)
```

logo



logo

