

@_workchronicles

follow on *Twitter & Instagram* for more (and don't forget the *underscore*)

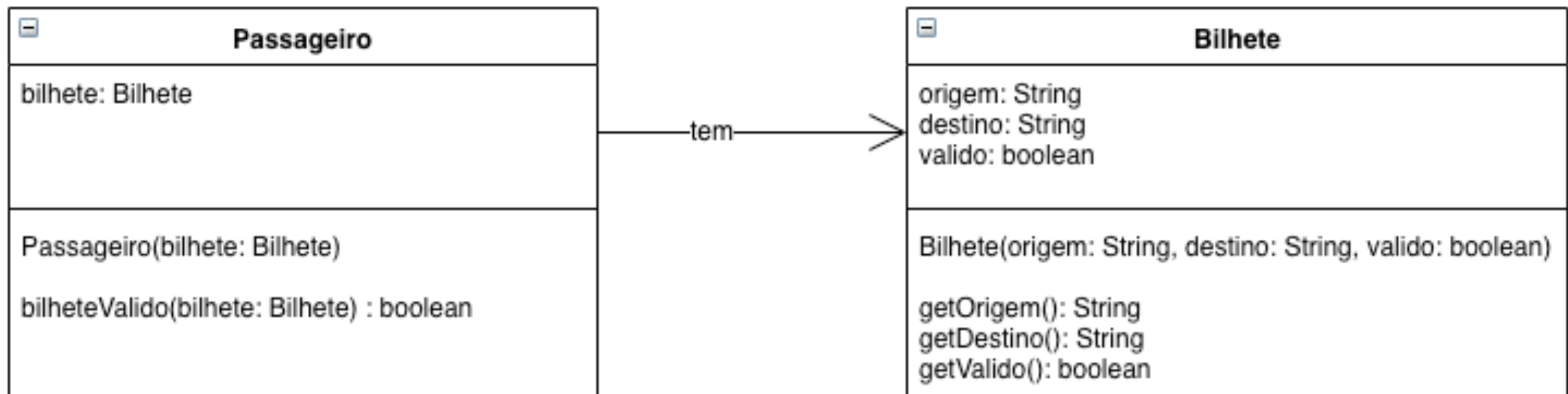
workchronicles.com

© Pedro Alves 2020

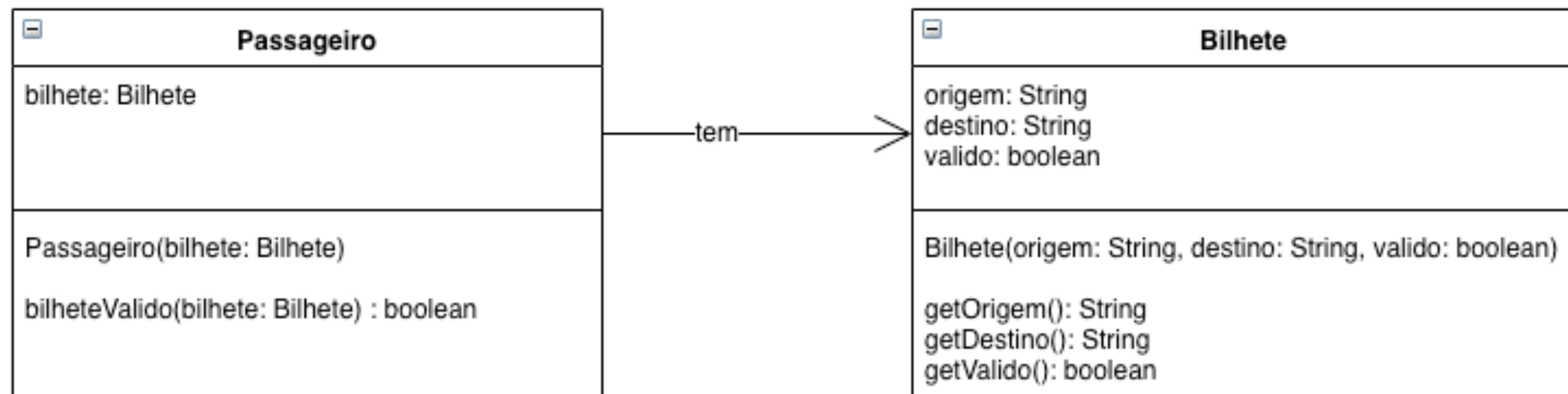
Exercício UML (breakout rooms)



O que está mal neste diagrama?



Exercício UML


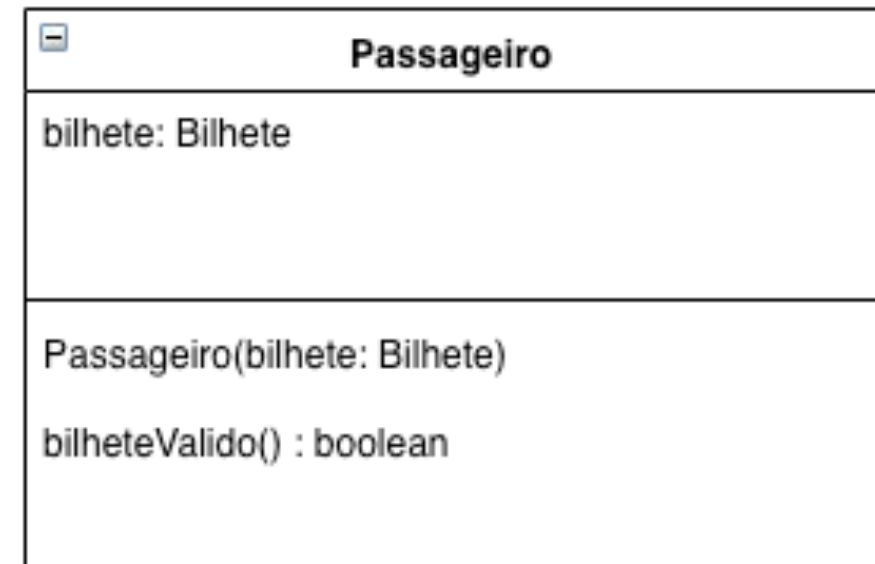
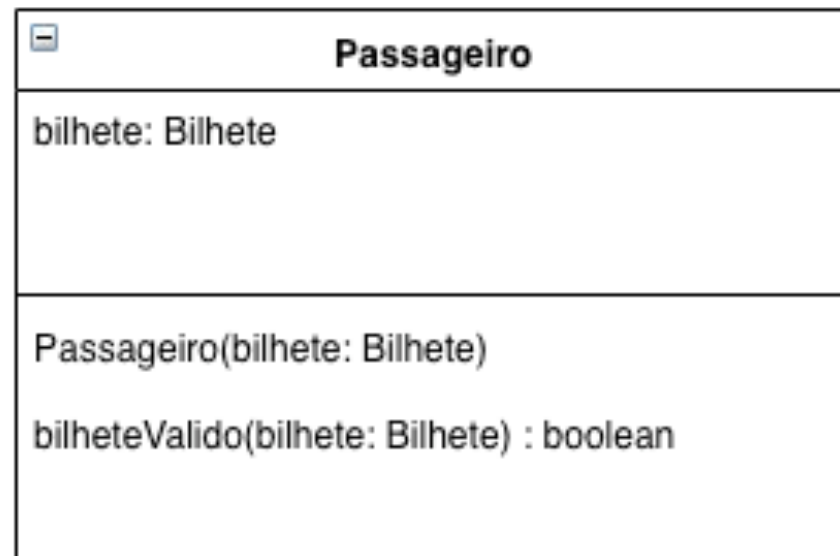


```
class Passageiro {
    Bilhete bilhete;


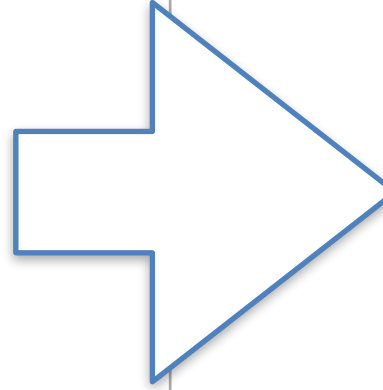
    Passageiro(Bilhete bilhete) {
        this.bilhete = bilhete;
    }

    boolean bilheteValido(Bilhete bilhete) {
        return bilhete.getValido();
    }
}
```

Exercício UML

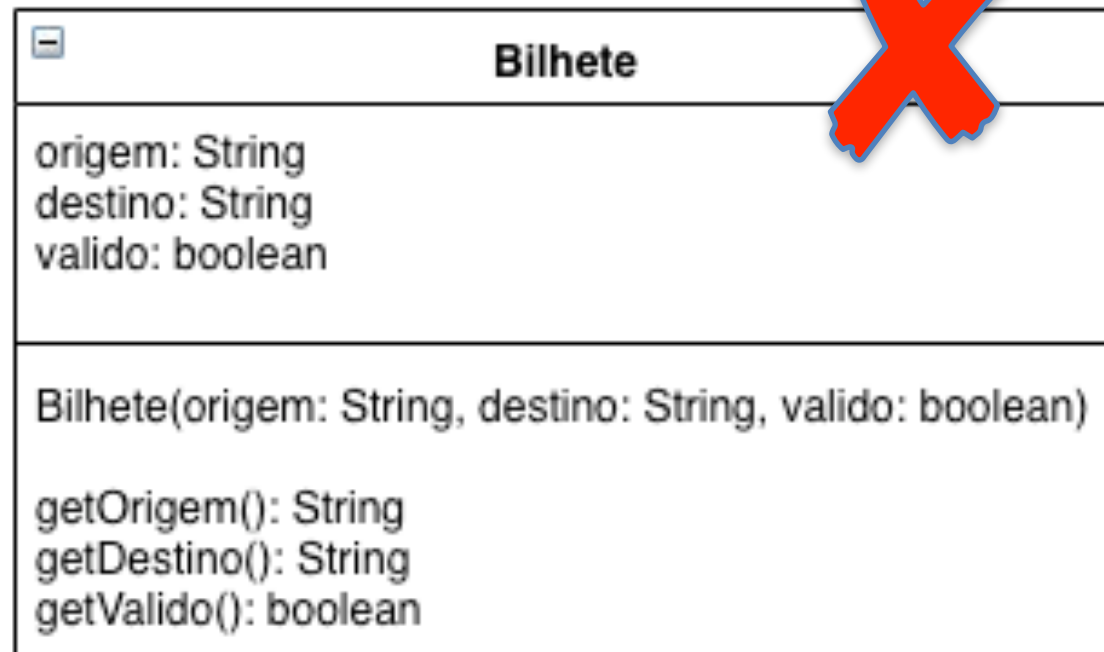


```
class Passageiro {  
    Bilhete bilhete;  
  
    Passageiro(Bilhete bilhete) {  
        this.bilhete = bilhete;  
    }  
  
    boolean bilheteValido(Bilhete bilhete) {  
        return bilhete.getValido();  
    }  
}
```

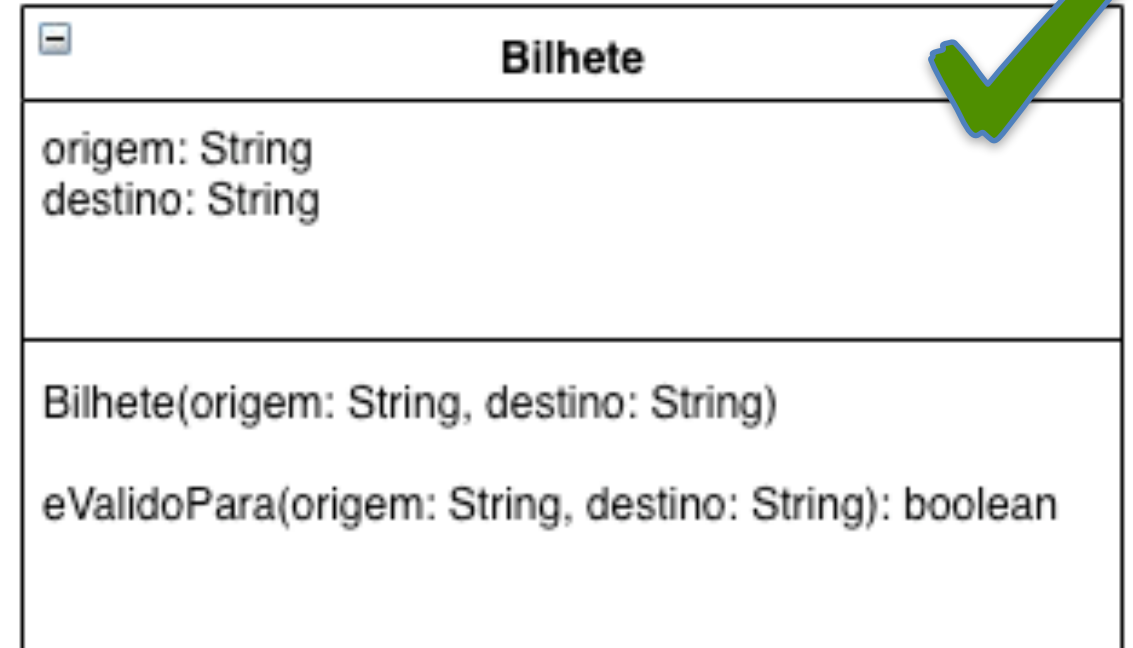


```
class Passageiro {  
    Bilhete bilhete;  
  
    Passageiro(Bilhete bilhete) {  
        this.bilhete = bilhete;  
    }  
  
    boolean bilheteValido() {  
        return bilhete.getValido();  
    }  
}
```

Exercício UML

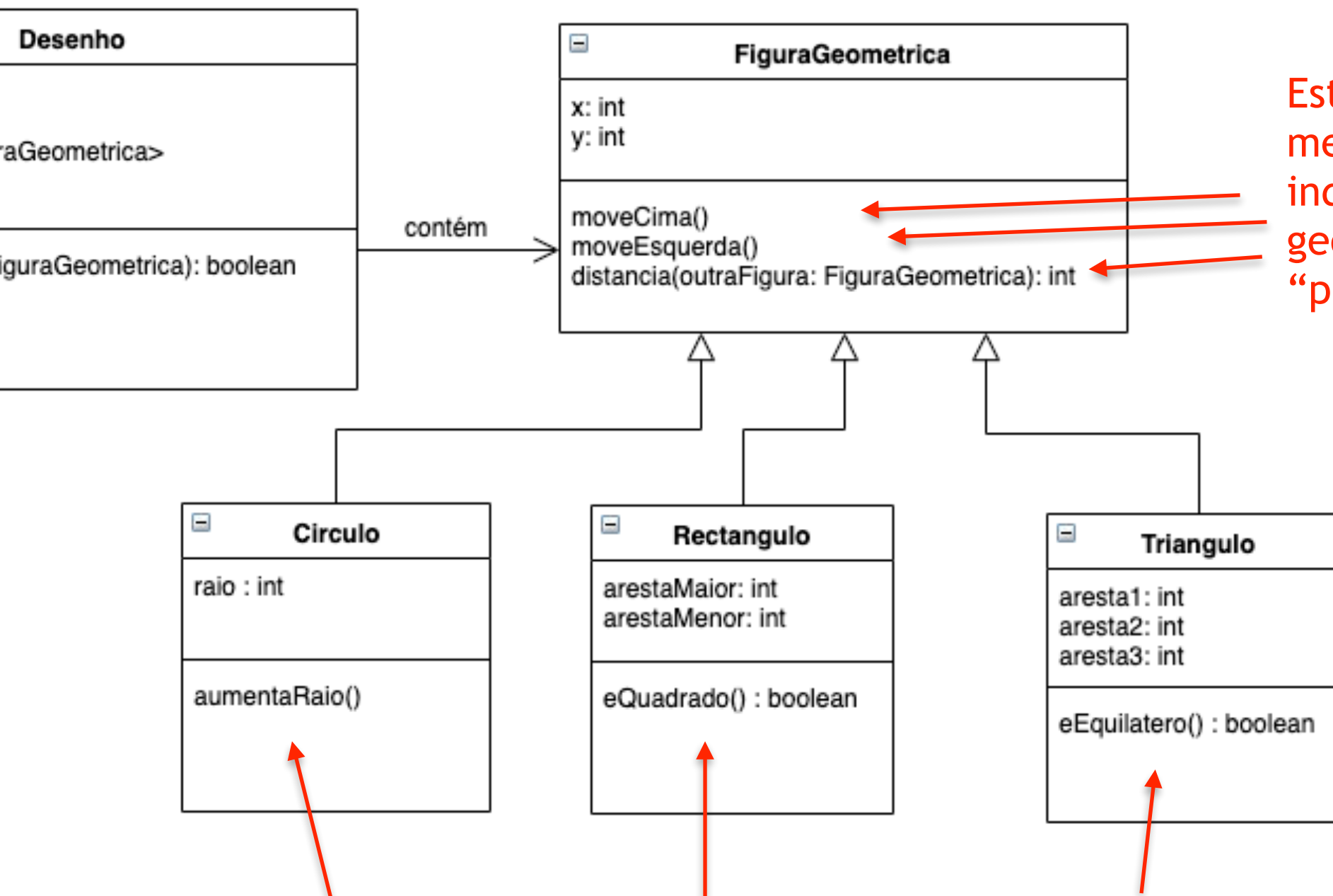


- Faz sentido um bilhete ser construído logo com o estado inválido?
- **valido** não é um atributo, é uma acção (validar) que utiliza atributos da classe para verificar se é válido



```
Bilhete bilhete = new Bilhete("Lisboa", "Barcelona");  
  
if (bilhete.eValidoPara("Lisboa", "Paris")) {  
    System.out.println("Vou entrar no avião");  
}
```

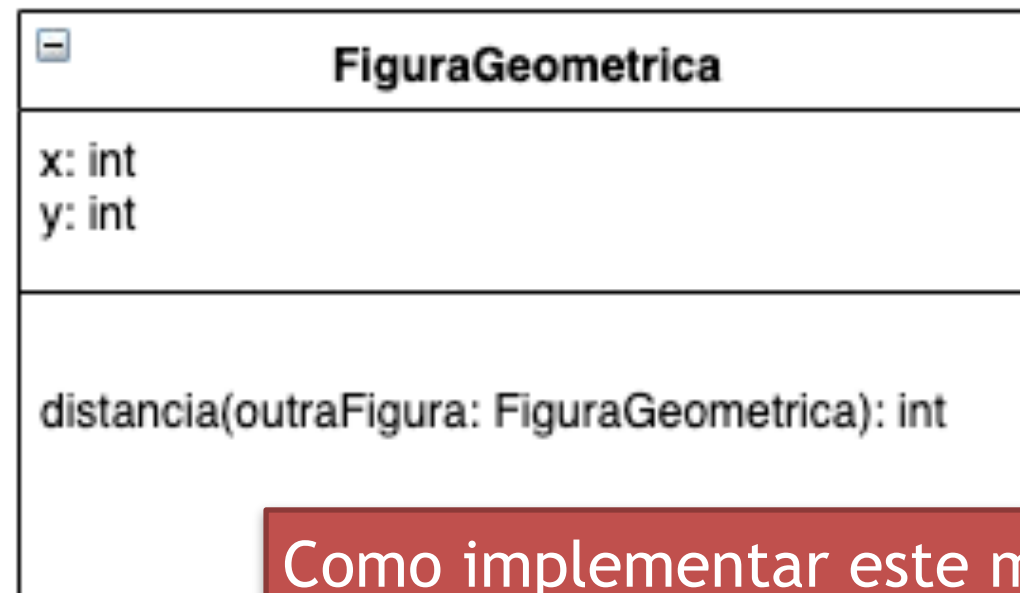
Resolução



Estes métodos funcionam da mesma forma independentemente da figura geométrica. Por isso estão no “pai”.

Estes métodos só fazem sentido na figura específica ao qual pertencem. Por isso estão no “filho”.

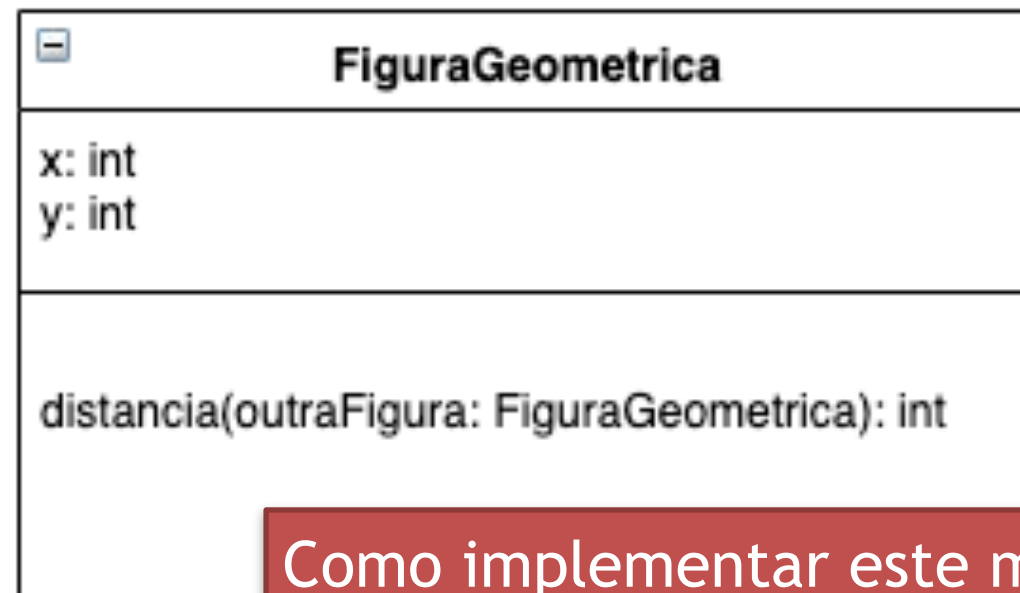
Exemplo



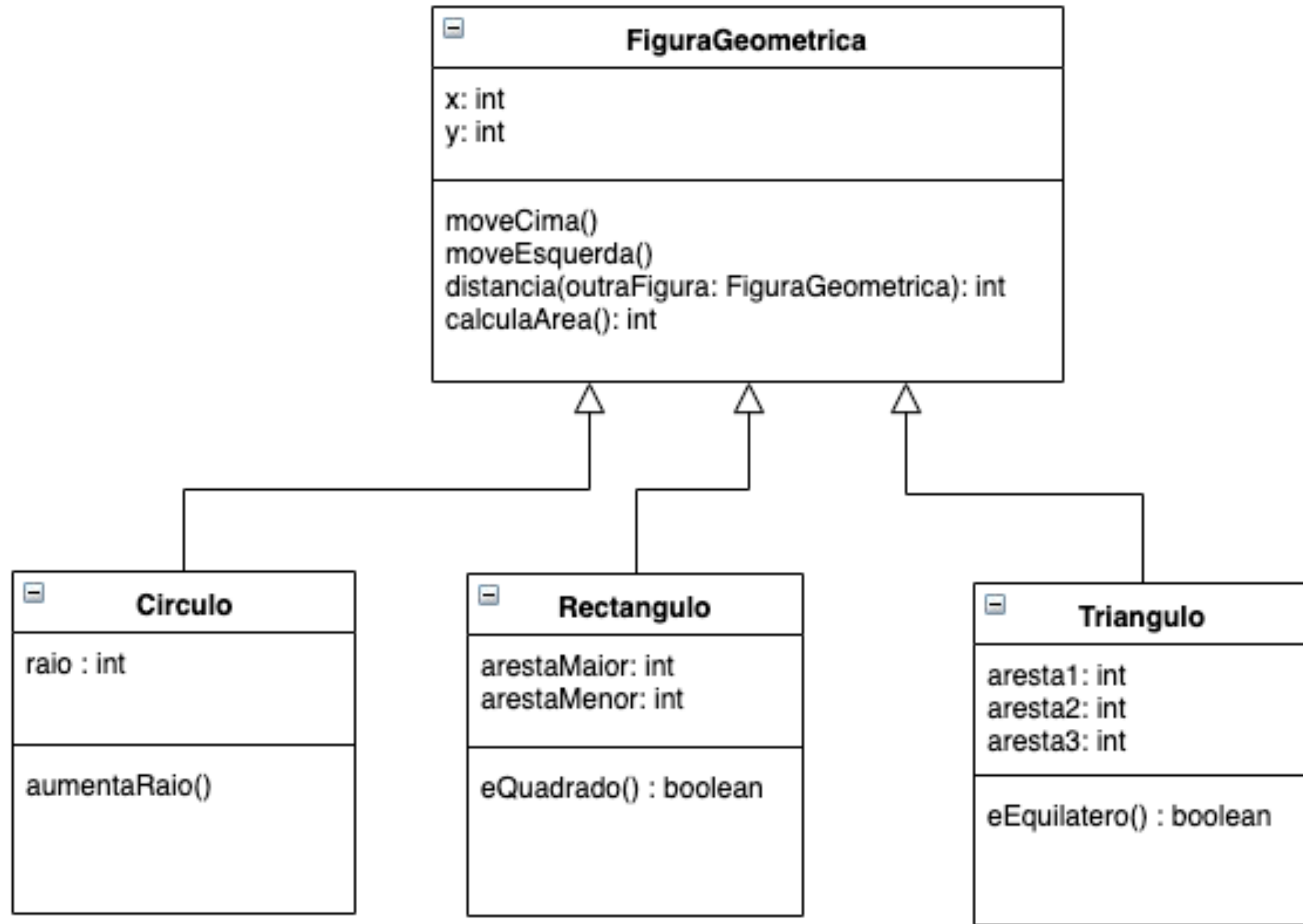
Como implementar este método?

```
Circulo c1 = new Circulo(3, 5);  
  
Rectangulo r1 = new Rectangulo(3, 8);  
  
System.out.println("Distancia = " + c1.distancia(r1));
```

Exemplo

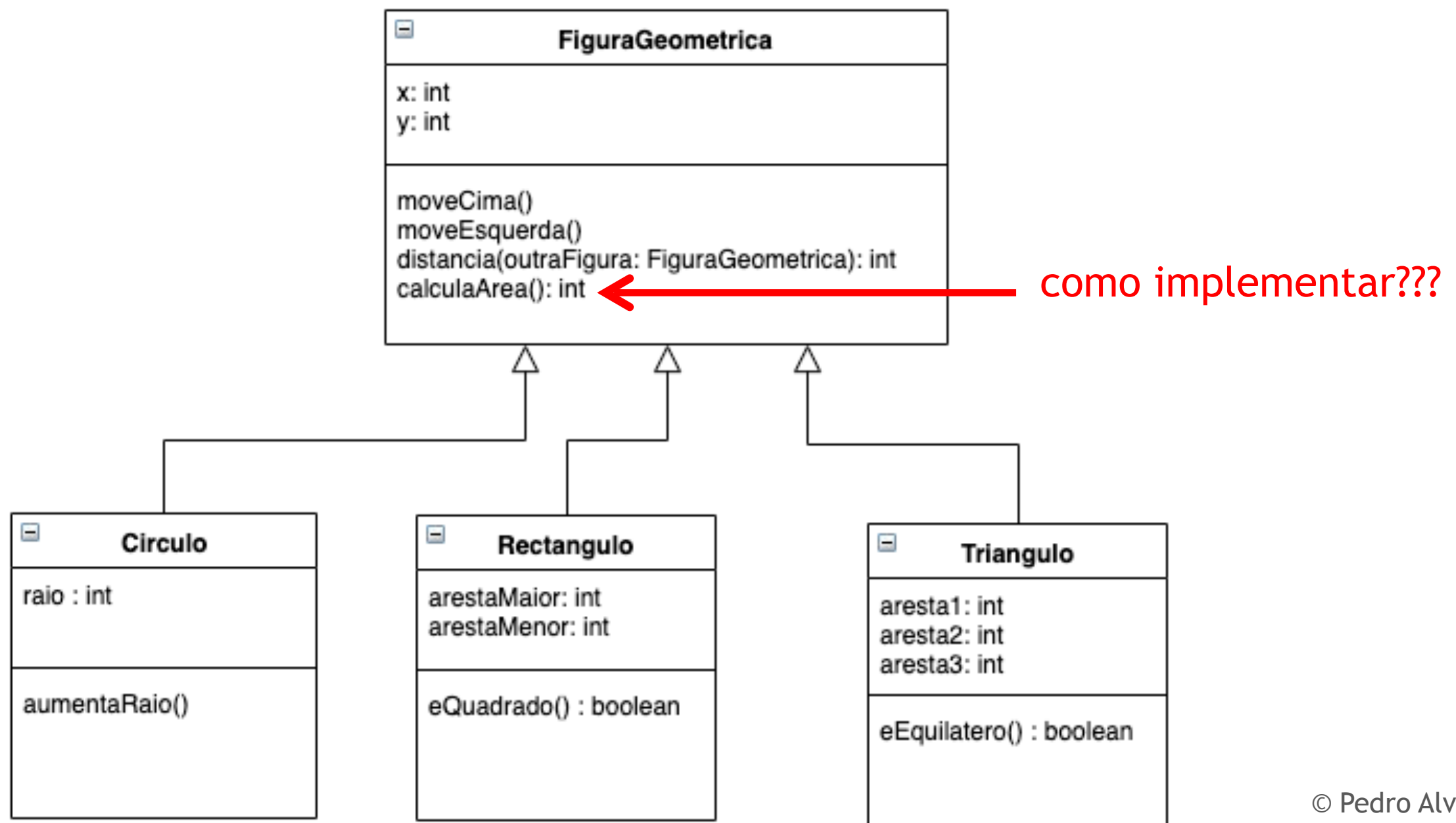


```
class FiguraGeometrica {  
    int x, y;  
  
    int distancia(FiguraGeometrica outraFigura) {  
        return (int) Point2D.distance(x, y, outraFigura.x, outraFigura.y);  
    }  
}
```

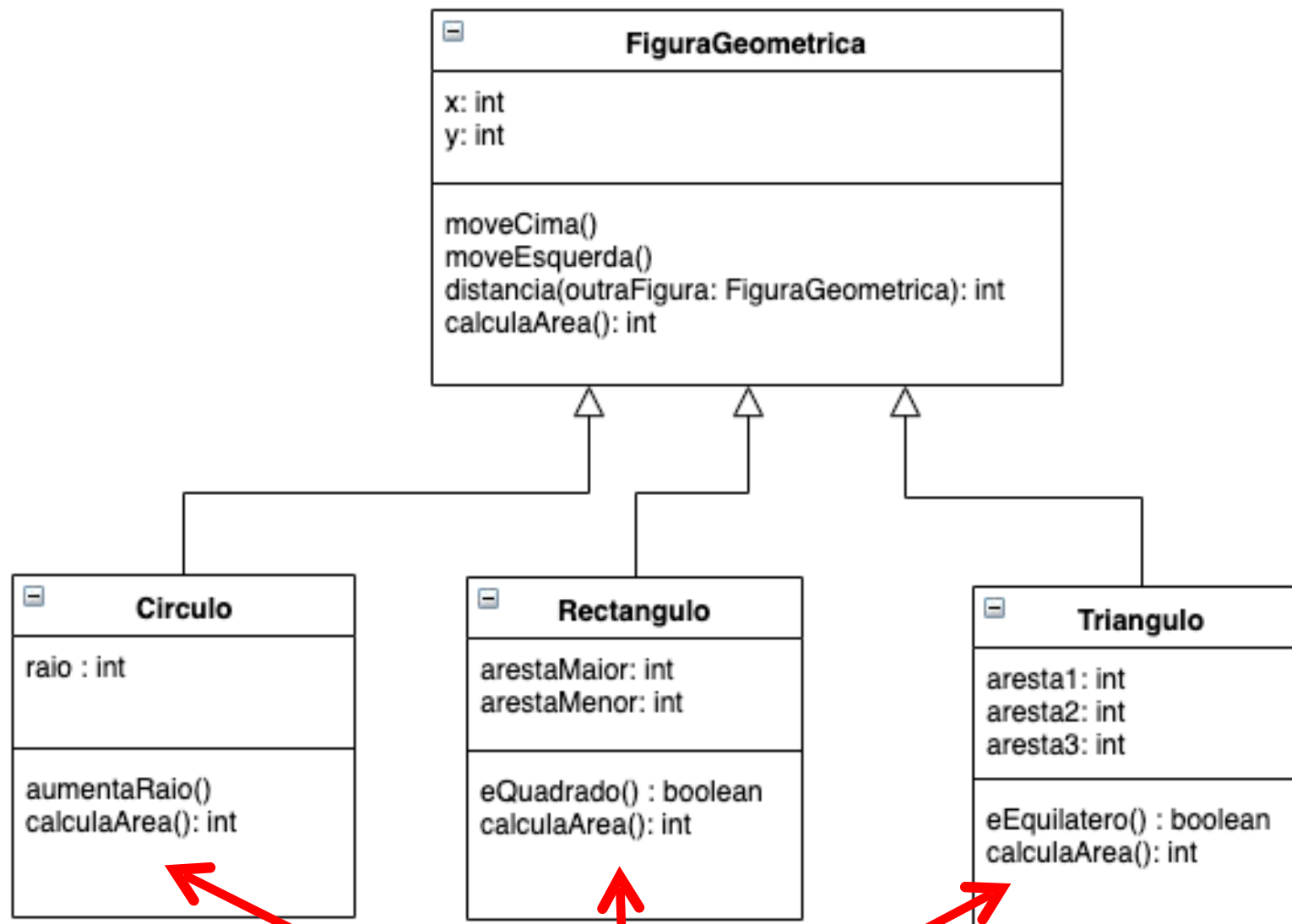
Próxima aula

Todas as figuras geométricas têm uma área mas o seu cálculo varia de figura para figura



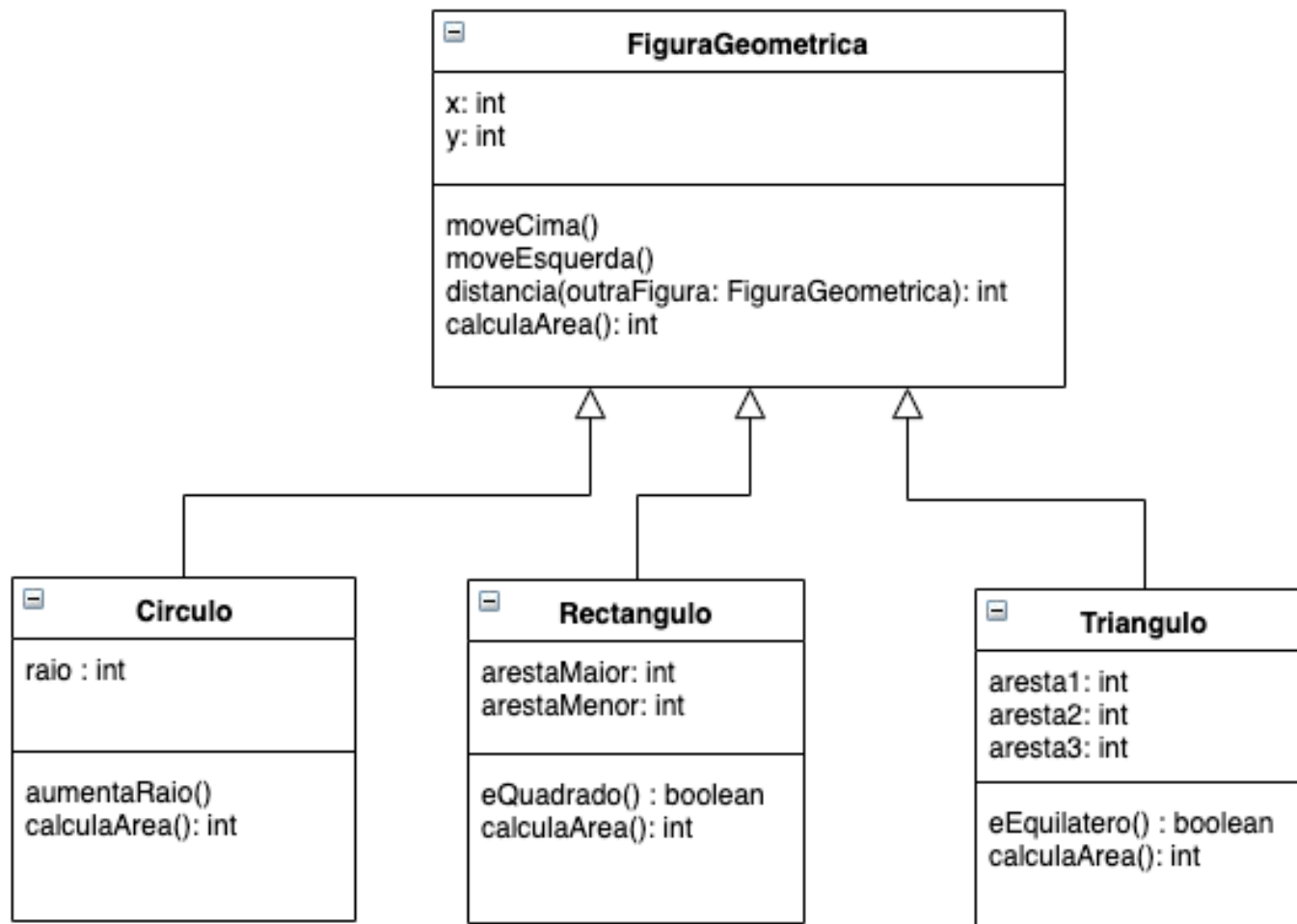
Herança

Todas as figuras geométricas têm uma área mas o seu cálculo varia de figura para figura



Implementações específicas

Herança



```
class FiguraGeometrica {
    int x, y;

    int calculaArea() {
        return 0; // ERRADO!
    }
}
```

```
class Circulo extends FiguraGeometrica {

    int raio;

    int calculaArea() {
        return (int) (raio * raio * Math.PI);
    }
}

class Rectangulo extends FiguraGeometrica {

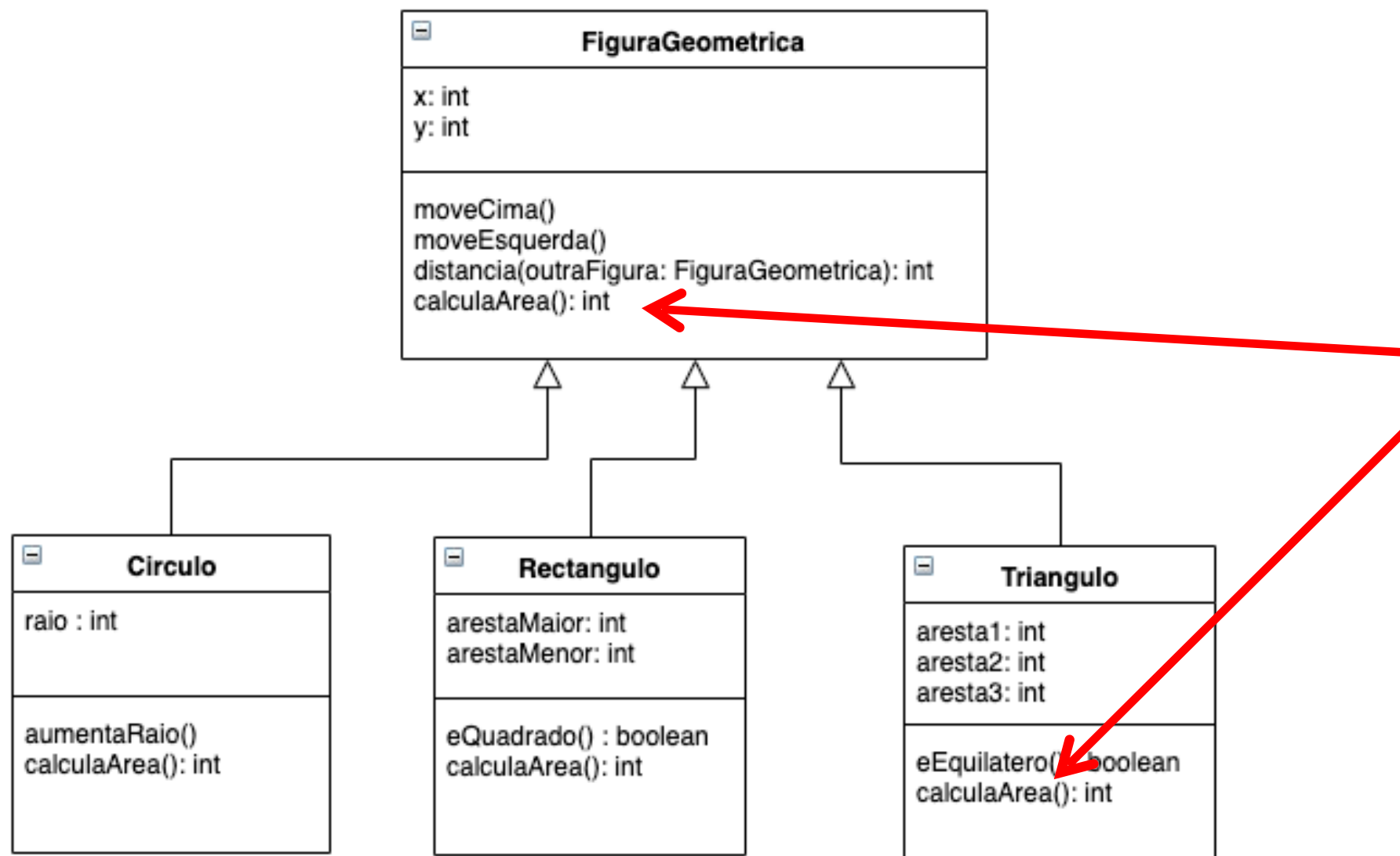
    int arestaMaior, arestaMenor;

    int calculaArea() {
        return arestaMaior * arestaMenor;
    }
}
```

Herança

Method overriding

Quando um método de uma subclasse tem a mesma assinatura que um método de uma superclasse



O método *calculaArea* foi **overriden** (redefinido) pela classe Triangulo

Override vs Overload

Method overriding

Quando um método de uma subclasse tem a mesma assinatura (nome, argumentos e retorno) que um método de uma superclasse

```
class FiguraGeometrica {  
  
    Point localizacao;  
  
    int calculaArea() {  
        |     return 0; // wrong, must be overridden  
    }  
}  
  
class Circulo extends FiguraGeometrica {  
  
    int raio;  
  
    // overrides calculaArea() from FiguraGeometrica  
    int calculaArea() {  
        |     return (int) (raio * raio * Math.PI);  
    }  
  
    // overloads calculaArea()  
    int calculaArea(boolean aDuplicar) {  
        |     return this.calculaArea() * 2;  
    }  
}
```

Method overloading

Quando um método tem o mesmo nome mas argumentos diferentes

@Override

```
class FiguraGeometria {  
    int x, y;  
  
    int calculaArea() {  
        return 0;  
    }  
}
```

```
class Circulo extends FiguraGeometria {  
    int raio;  
  
    @Override  
    int calculaArea() {  
        return (int) raio * raio * Math.PI  
    }  
}
```

```
class Rectangulo extends FiguraGeometria {  
    int arestaMaior, arestaMenor;  
  
    @Override  
    int calculaArea() {  
        return arestaMaior * arestaMenor;  
    }  
}
```

Method overriding

Para sinalizar que estamos a fazer overriding devemos “marcar” esses métodos com a anotação **@Override**

Problema

```
class FiguraGeometrica {  
    int x, y;  
  
    int calculaArea() {  
        return 0;  
    }  
}
```

Cheira-me que isto vai dar asneira

```
class Circulo extends FiguraGeometrica {  
    int raio;  
  
    @Override  
    int calculaArea() {  
        return (int) (raio * raio * Math.PI);  
    }  
}
```



```
class Triangulo extends FiguraGeometrica {  
}
```

E se me esquecer de redefinir (override) o calculaArea() ???

Métodos abstratos

```
abstract class FiguraGeometrica {  
    int x, y;  
  
    abstract int calculaArea();  
}
```

Um método **abstrato** é um método que não tem implementação no pai e que os filhos são obrigados a implementar

```
class Circulo extends FiguraGeometrica {  
    int raio;  
  
    @Override  
    int calculaArea() {  
        return (int) (raio * raio * Math.PI);  
    }  
}
```

```
class Triangulo extends FiguraGeometrica {  
  
}
```

Erro de compilação! Preciso de redefinir o calculaArea() !

Métodos abstratos

```
abstract class FiguraGeometrica {  
    int x, y;  
  
    abstract int calculaArea();  
}  
  
class Circulo extends FiguraGeometrica {  
    int raio;  
  
    @Override  
    int calculaArea() {  
        return (int) (raio * raio * Math.PI);  
    }  
}  
  
class Triangulo extends FiguraGeometrica {  
    int base, altura;  
  
    @Override  
    int calculaArea() {  
        return base * altura / 2;  
    }  
}
```

Todas as subclasses redefinem (override) o `calculaArea()`, por isso não há erros de compilação

Métodos abstratos

```
abstract class FiguraGeometrica {  
    int x, y;  
  
    abstract int calculaArea();  
}  
  
class Circulo extends FiguraGeometrica {  
    int raio;  
  
    @Override  
    int calculaArea() {  
        return (int) (raio * raio * Math.PI);  
    }  
}  
  
class Triangulo extends FiguraGeometrica {  
    int base, altura;  
  
    @Override  
    int calculaArea() {  
        return base * altura / 2;  
    }  
}
```

Métodos abstratos

Métodos sem implementação (só a assinatura) que têm obrigatoriamente que ser redefinidos por todas as subclasses

1. A classe respetiva tem que ser abstract
2. O método tem que ser abstract
3. Não pode ter implementação

Classes abstratas

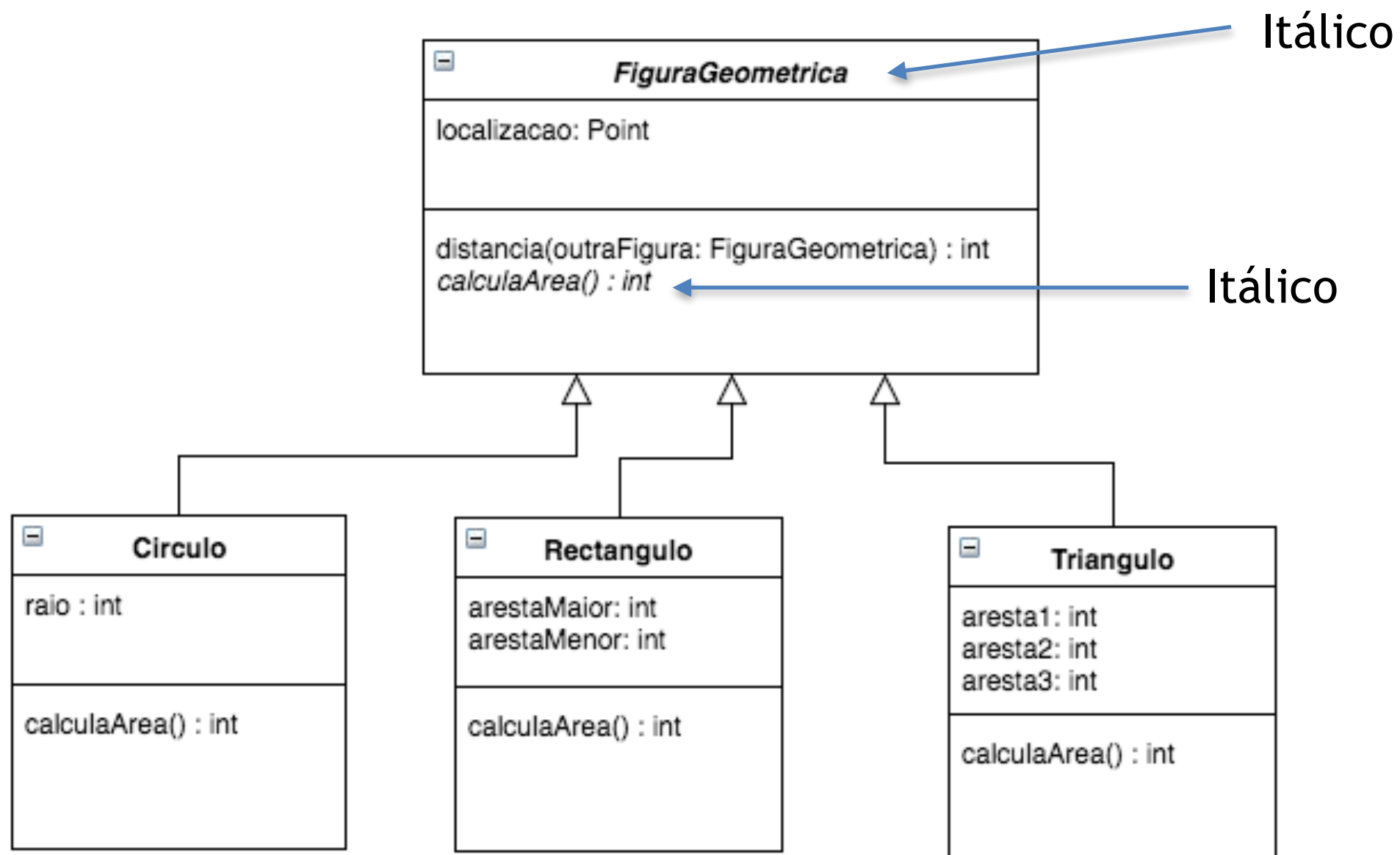
Classes abstratas são classes que têm um ou mais métodos abstractos

Não podem ser instanciadas!!

```
abstract class FiguraGeometrica {  
  
    Point localizacao;  
  
    // must be overridden in all subclasses  
    abstract int calculaArea();  
}  
  
class App {  
    public static void main(String[] args) {  
        new FiguraGeometrica();  
    }  
}
```

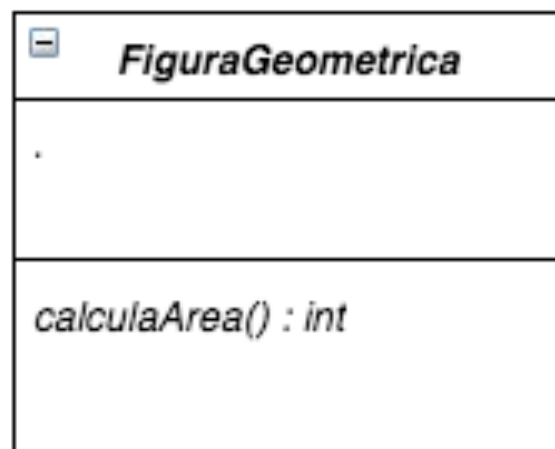
Classes abstratas

Em UML, as classes e métodos abstractos devem estar em *itálico*

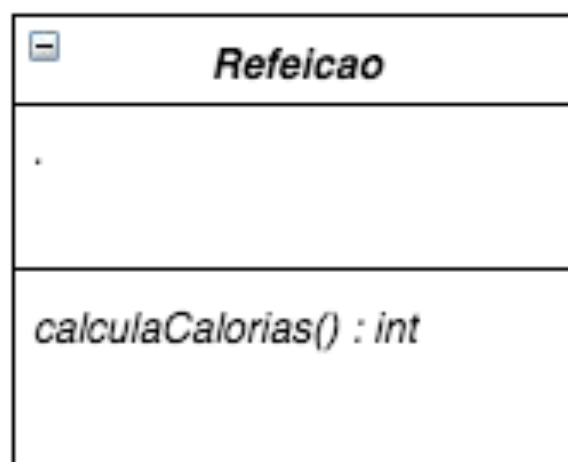


Classes abstratas

Devem ser usadas quando queremos indicar que existe um certo comportamento mas não o modo como esse comportamento acontece



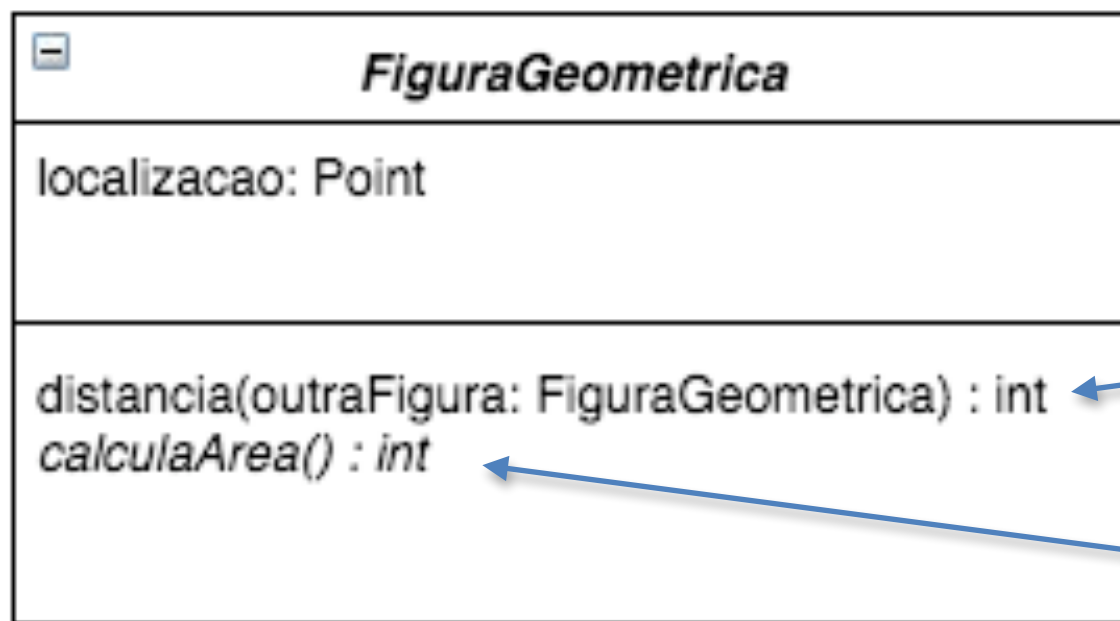
Todas as figuras geométricas têm que conseguir calcular a área, mas a forma como é calculada varia de figura para figura



Todas as refeições têm que conseguir calcular as suas calorias, mas a forma como é calculada varia de refeição para refeição

Classes abstratas

Classes abstractas podem ter métodos não-abstratos
(comportamento por omissão)



método não-abstrato

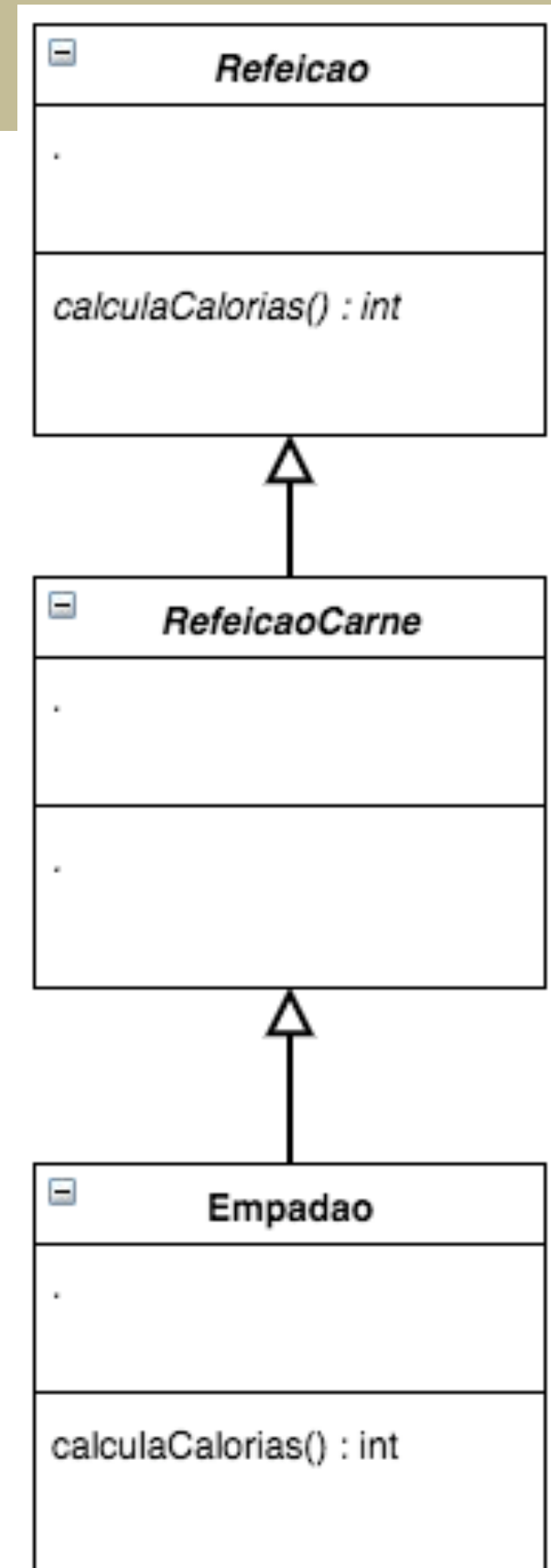
método abstrato

Classes abstratas

Se uma classe A herdar de uma classe abstracta B, então tem que implementar todos os métodos abstractos de B

Ou então, pode ela própria ser abstracta!

Neste exemplo, Refeicao e RefeicaoCarne são abstractos



Construtores + herança

Vídeo no Moodle

Exercício 1

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Qual o output deste programa?
Enviar via teams para p4997

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    4 Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    4 Pessoa(String nome, int idade) {  
        5 this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Resolução

```
class Pessoa {  
    6 Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
    4 Pessoa(String nome, int idade) {  
        5 this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```


Resolução

```
class Pessoa {  
    6 Pessoa(String nome) {  
        7 System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    4 Pessoa(String nome, int idade) {  
        5 this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Sou uma pessoa com o nome cristina

Resolução

```
class Pessoa {  
    6 Pessoa(String nome) {  
        7 System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
    4 Pessoa(String nome, int idade) {  
        5 this(nome);  
        8 System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30

Resolução

```
class Pessoa {  
    6 Pessoa(String nome) {  
        7 System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
    4 Pessoa(String nome, int idade) {  
        5 this(nome);  
        8 System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
    2 Estudante(String nome, int idade, int numeroEstudante) {  
        3 super(nome, idade);  
        9 System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567); 1  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante();  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567

```
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        2 super("pedro", 32);  
    }  
}
```

```
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante(); 1  
    }  
}
```

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        3 this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        2 super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante(); 1  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        4 System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        3 this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        2 super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante(); 1  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567
Sou uma pessoa com o nome pedro

Resolução

```
class Pessoa {  
  
    Pessoa(String nome) {  
        4 System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        2 this(nome);  
        5 System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    Estudante() {  
        2 super("pedro", 32);  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        Estudante estudante2 = new Estudante(); 1  
    }  
}
```

Sou uma pessoa com o nome cristina
Sou uma pessoa com a idade 30
Sou um estudante com o numero 234567
Sou uma pessoa com o nome pedro
Sou uma pessoa com a idade 32

Exercício 2

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
    }  
  
    void printActivity() {  
        System.out.println("Vou ser");  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    void printActivity() {  
        System.out.println("Vou estudar");  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Pessoa pessoa = new Pessoa("pedro");  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
        pessoa.printActivity();  
        estudante1.printActivity();  
    }  
}
```

Qual o output deste programa?

Exercício 3

```
class Pessoa {  
  
    Pessoa(String nome) {  
        System.out.println("Sou uma pessoa com o nome " + nome);  
    }  
  
    Pessoa(String nome, int idade) {  
        this(nome);  
        System.out.println("Sou uma pessoa com a idade " + idade);  
        printActivity();  
    }  
  
    void printActivity() {  
        System.out.println("Vou ser");  
    }  
}  
  
class Estudante extends Pessoa {  
  
    Estudante(String nome, int idade, int numeroEstudante) {  
        super(nome, idade);  
        System.out.println("Sou um estudante com o numero " + numeroEstudante);  
    }  
  
    void printActivity() {  
        System.out.println("Vou estudar");  
    }  
}  
  
class App {  
    public static void main(String[] args) {  
        Estudante estudante1 = new Estudante("cristina", 30, 234567);  
    }  
}
```

Qual o output deste programa?

Classificadores de visibilidade

Vídeo no Moodle

Exercício

Colocar os classificadores de visibilidade

```
1 class Veiculo {
2
3     String matricula;
4     int cilindrada;
5
6     Veiculo(String matricula, int cilindrada) {
7         this.matricula = matricula;
8         this.cilindrada = cilindrada;
9     }
10
11     String getMatricula() {
12         return matricula;
13     }
14
15     int getCilindrada() {
16         return cilindrada;
17     }
18 }
19
20 class Carro extends Veiculo {
21
22     int numPortas;
23
24     Carro(String matricula, int cilindrada, int numPortas) {
25         super(matricula, cilindrada);
26         this.numPortas = numPortas;
27     }
28
29     void modificaMotor(int novaCilindrada) {
30         this.cilindrada = novaCilindrada;
31     }
32 }
33
34
35 class App {
36     public static void main(String[] args) {
37         Carro carro = new Carro("34-ED-21", 1100, 5);
38         System.out.println("A matricula é " + carro.getMatricula());
39         carro.modificaMotor(1200);
40     }
41 }
```


Enviar via teams para p4997

Resolução

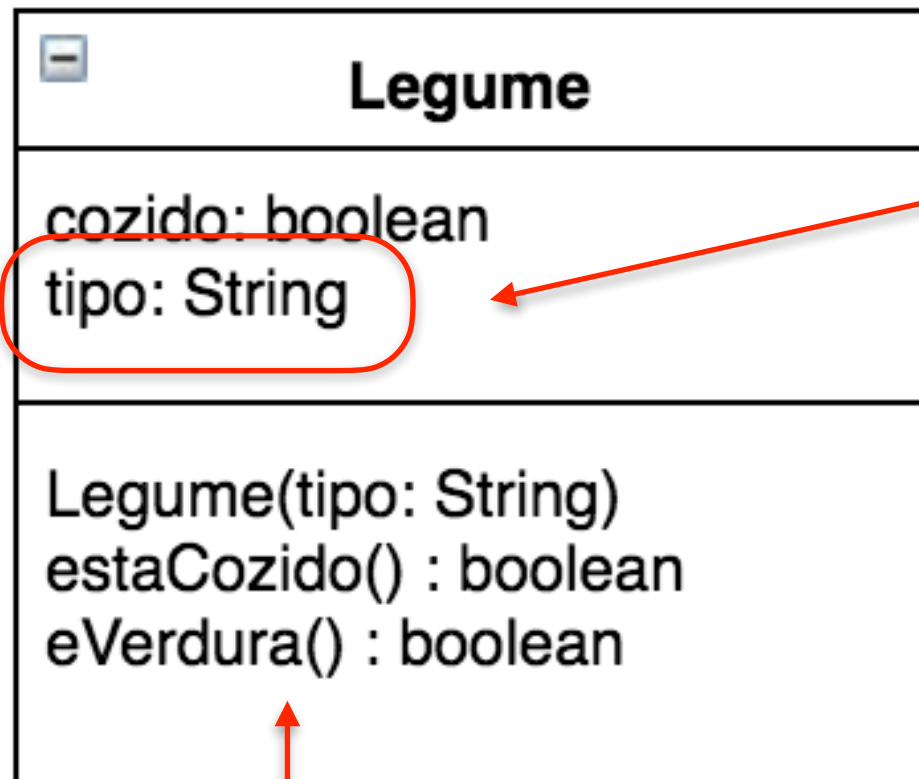
Colocar os classificadores de visibilidade

```
1  class Veiculo {
2
3      private String matricula;
4      protected int cilindrada;
5
6      public Veiculo(String matricula, int cilindrada) {
7          this.matricula = matricula;
8          this.cilindrada = cilindrada;
9      }
10
11     public String getMatricula() {
12         return matricula;
13     }
14
15     public int getCilindrada() {
16         return cilindrada;
17     }
18 }
19
20 class Carro extends Veiculo {
21
22     private int numPortas;
23
24     public Carro(String matricula, int cilindrada, int numPortas) {
25         super(matricula, cilindrada);
26         this.numPortas = numPortas;
27     }
28
29     public void modificaMotor(int novaCilindrada) {
30         this.cilindrada = novaCilindrada;
31     }
32
33 }
34
35 public class App {
36     public static void main(String[] args) {
37         Carro carro = new Carro("34-ED-21", 1100, 5);
38         System.out.println("A matricula é " + carro.getMatricula());
39         carro.modificaMotor(1200);
40     }
41 }
```

Quando usar herança

| | |
|---|---------------|
|  | Legume |
| cozido: boolean tipo: String | |
| Legume(tipo: String) estaCozido() : boolean eVerdura() : boolean | |

Quando usar herança

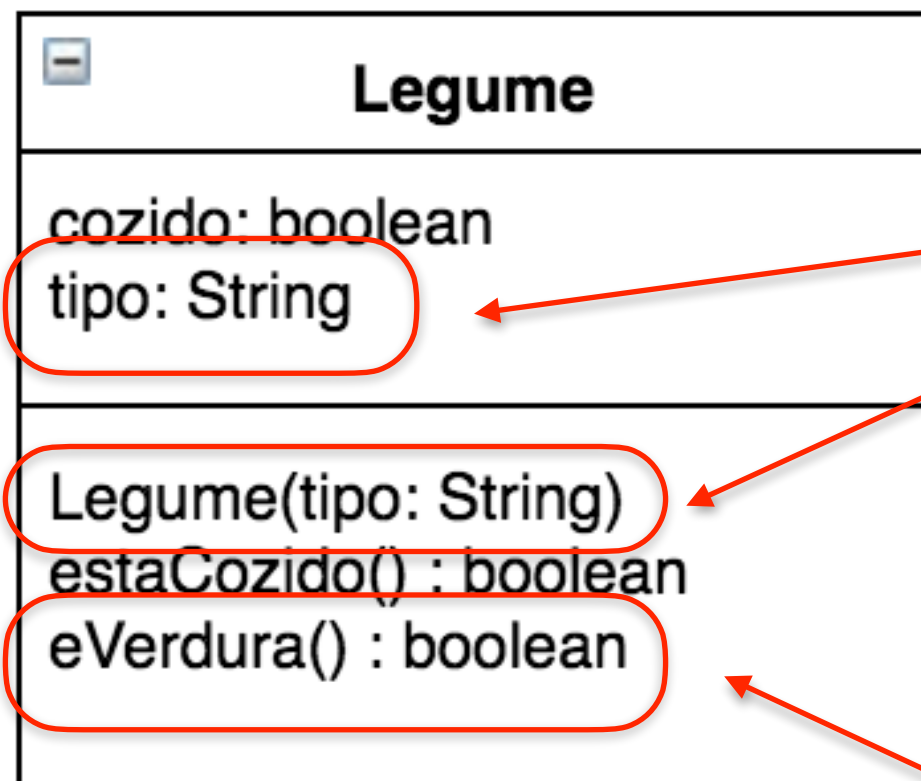


O tipo pode ser “Cenoura”,
“Couve”, “Bróculo”, “Cebola”

```
boolean eVerdura() {  
    if (tipo.equals("Cenoura") || tipo.equals("Cebola")) {  
        return false;  
    }  
    if (tipo.equals("Bróculo") || tipo.equals("Couve")) {  
        return true;  
    }  
}
```

Quando usar herança

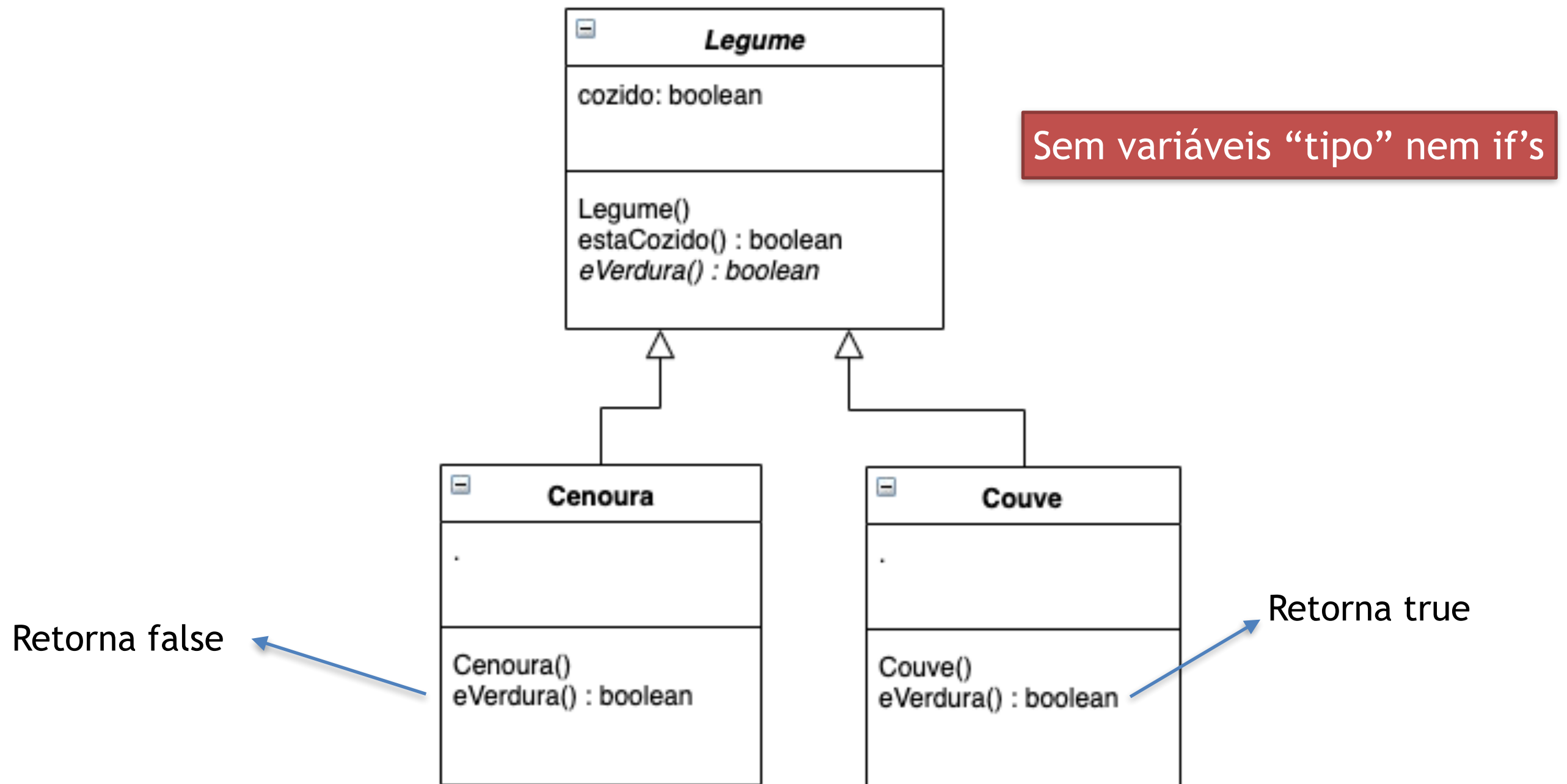
Este modelo tem vários problemas!



Uma variável “tipo” indica quase sempre um erro na modelação orientada a objectos.
Cada tipo deverá ser uma subclasse

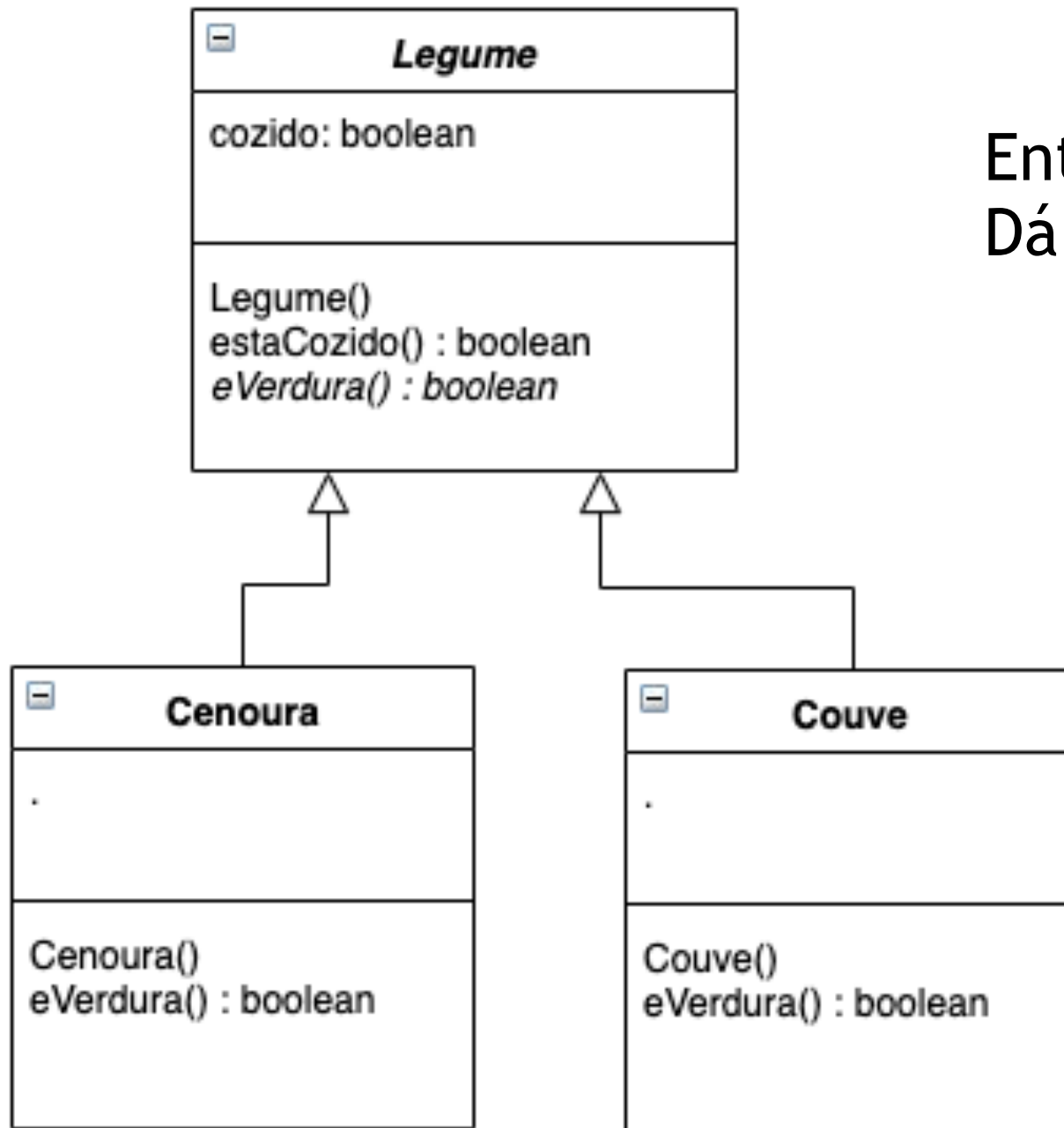
Quando um método tem que fazer uma série de if's para decidir o que fazer, possivelmente temos um erro na modelação orientada a objectos.
Cada subclasse deve fazer override a este método.

Quando usar herança



Nota: Classes Brocolo e Cebola omitidas por simplificação

Próxima aula



Então e se tivermos uma `List<Legume>`?
Dá para misturar cenouras com couves?

Paint - Episódio 4

Paint com herança!

A resolução deste episódio pode valer até 0,3 valores

Nota máxima apenas para quem tira coelhos da cartola.

Importante: No comentário git têm que descrever o vosso “coelho”

Prazo: 1/Dez 23h00



<https://classroom.github.com/a/EM124I71>