

# **Programação OO em Java**

Profa Andréa Schwertner Charão  
DELC/CT/UFSM

# Exemplo: classe Ponto

- 2D, representado por uma coordenada (x,y)
- Construtores
  - Ponto(): default (0.0, 0.0)
  - Ponto(x,y): recebe valores de x e y
- Métodos
  - mover (x, y) para (x+dx, y+dy)
  - calcular distância de um ponto a outro

```
class Ponto
{
    private double x;
    private double y;

    public Ponto(){}
    public Ponto(double _x, double _y){}
    void move(double dx, double dy){}
    double distancia(Ponto p2){}
}

class PontoApp
{
    public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        System.out.println(p1.distancia(p2));
    }
}
```

# Exemplo: classe Ponto

## ■ Construtores

- Ponto():  
default (0.0, 0.0)
- Ponto(x,y): recebe  
valores de x e y

```
class Ponto
{
    private double x;
    private double y;

    public Ponto()
    {
        x = 0.0;
        y = 0.0;
    }

    public Ponto(double _x, double _y)
    {
        x = _x;
        y = _y;
    }

    + void move(double dx, double dy)[]
    + double distancia(Ponto p2)[]
    }

class PontoApp
{
    - public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        System.out.println(p1.distancia(p2));
    }
}
```

# Exemplo: classe Ponto

- Usando os construtores no método main

```
class PontoApp
{
    public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
    }
}
```

# Exemplo: classe Ponto

## ■ Métodos

- mover (x, y) para (x+dx, y+dy)
- calcular distância de um ponto a outro

```
class Ponto
{
    private double x;
    private double y;

    + public Ponto()
    + public Ponto(double _x, double _y)
    - void move(double dx, double dy)
    {
        x += dx;
        y += dy;
    }

    - double distancia(Ponto p2)
    {
        double dx = p2.x - x;
        double dy = p2.y - y;
        return Math.sqrt(dx*dx + dy*dy);
    }
}

class PontoApp
{
    - public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        p1.move(0.5,0.5);
        System.out.println(p1.distancia(p2));
    }
}
```

# Exemplo: classe Ponto

## ■ Método distancia

- se refere a um objeto da classe Ponto
- é um método de instância
- recebe referência para outro objeto Ponto
- para usá-lo precisamos instanciar um Ponto

```
double distancia(Ponto p2)
{
    double dx = p2.x - x;
    double dy = p2.y - y;
    return Math.sqrt(dx*dx + dy*dy);
}

class PontoApp
{
    public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        p1.move(0.5,0.5);
        System.out.println(p1.distancia(p2));
    }
}
```

# Exemplo: classe Ponto

- Por que não declarar método distancia assim?

```
double distancia(Ponto p1, Ponto p2) {...}
```

- o método já se refere a um objeto ponto
- seria **anti orientação a objetos**

- Outra alternativa para o método distancia?

- implementá-lo usando **static**

# Uso de static em Java

## ■ Exemplo com static

```
class Ponto
{
    private double x;
    private double y;

    + public Ponto(){}
    + public Ponto(double _x, double _y){}
    - public static double distancia(Ponto p1, Ponto p2)
    {
        double dx = p2.x - p1.x;
        double dy = p2.y - p1.y;
        return Math.sqrt(dx*dx + dy*dy);
    }

    + public void move(double dx, double dy){}
}
```

Chamada ao método de classe:  
NomeDaClasse.NomeDoMetodo(...)

```
class PontoApp
{
    - public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        System.out.println(Ponto.distancia(p1, p2));
    }
}
```



# Resumindo: SEM uso de static

## ■ métodos e atributos **de instância**

- atributos e métodos declarados dentro de class pertencem a objetos desta classe
- cada objeto possui seus próprios valores para os atributos
- fora da classe, é necessário instanciar objeto para acessar atributos e métodos declarados na classe

```
double distancia(Ponto p2)
{
    double dx = p2.x - x;
    double dy = p2.y - y;
    return Math.sqrt(dx*dx + dy*dy);
}

class PontoApp
{
    public static void main(String[] args)
    {
        Ponto p1 = new Ponto();
        Ponto p2 = new Ponto(0.0,10.0);
        p1.move(0.5,0.5);
        System.out.println(p1.distancia(p2));
    }
}
```

# Resumindo: COM uso de static

- atributos e métodos **static** declarados dentro de class pertencem à classe, não aos objetos
- **não é necessário** instanciar objeto para acessar atributos e métodos declarados na classe
- métodos static são chamados de **métodos de classe**
- métodos static não podem acessar atributos de instância dentro da mesma classe
- atributos static são chamados de **atributos ou variáveis de classe**
- atributos static são **compartilhados** pelos objetos da classe (valor único)

# Atributos static

- Atributos static são geralmente usados para representar constantes ou variáveis "globais" dentro de uma classe

total é "global" na classe

```
class Relogio
{
    private int hora;
    private int minuto;
    private static int total = 0;
    public static final int MinutosHora = 60;

    public Relogio(){}

    public Relogio(int h, int m)
    {
        hora = h;
        minuto = m;
        total++;
    }

    public static int numInstancias()
    {
        return total;
    }

    public int totalMinutos(){}

    public static void main(String[] args){}
```

# Atributos static

- Qual será o resultado do **primeiro** println no método main?

total é "global" na classe

```
class Relogio
{
    private int hora;
    private int minuto;
    private static int total = 0;
    public static final int MinutosHora = 60;

    public Relogio()
    {
        hora = 0;
        minuto = 0;
        total++;
    }

    public static int numInstancias()
    {
        return total;
    }

    public static void main(String[] args)
    {
        Relogio r1 = new Relogio(10,30);
        Relogio r2 = new Relogio(10,20);

        System.out.println(Relogio.numInstancias());
        System.out.println(r1.totalMinutos());
    }
}
```

# Atributos static

- Qual será o resultado do **segundo** println no método main?

variável de classe constante (final)

```
class Relogio
{
    private int hora;
    private int minuto;
    private static int total = 0;
    public static final int MinutosHora = 60;

    public Relogio(){}
    public Relogio(int h, int m){}
    public static int numInstancias(){
    }
    public int totalMinutos()
    {
        return hora * MinutosHora + minuto;
    }
    public static void main(String[] args)
    {
        Relogio r1 = new Relogio(10,30);
        Relogio r2 = new Relogio(10,20);

        System.out.println(Relogio.numInstancias());
        System.out.println(r1.totalMinutos());
    }
}
```

# Mais sobre static em Java

Mais sobre isso em:

The Java Tutorials. Understanding Instance and classe Members.

<http://download.oracle.com/javase/tutorial/java/javaOO/classvars.html>