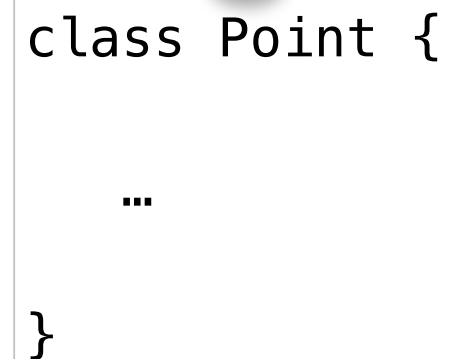


CLASSES DE OBJETOS

CLASSE DE OBJETOS

- Uma **classe de objetos** representa um **tipo de objetos**
- O nome da classe deve refletir o que os objetos são (no singular)
 - Exemplos: BinaryImage, Point, Lamp
 - Convenção: começar com letra maiúscula
- Um objeto é uma instância de uma classe
 - Por exemplo, uma imagem binária é uma instância da classe BinaryImage



```
class Point {  
    ...  
}
```

DEFINIÇÃO DE CLASSES DE OBJETOS

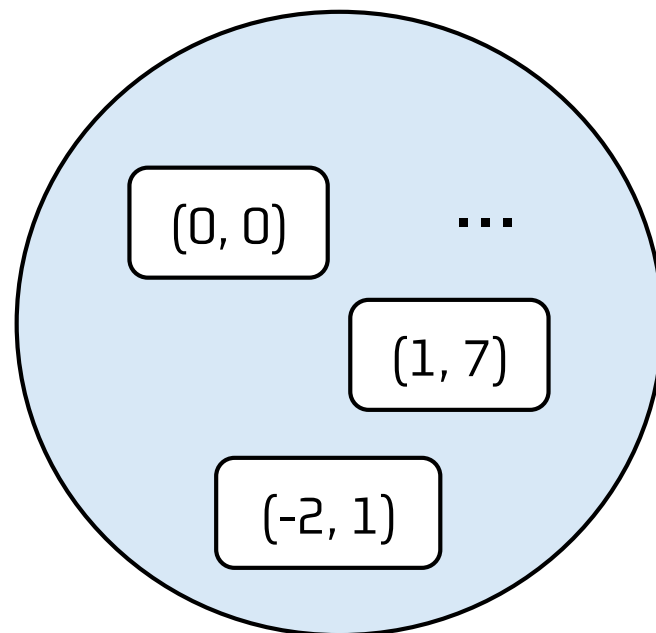
- A **definição** de uma classe de objetos é essencialmente composta por:
 - **Atributos:** variáveis que definem o estado de um objeto
 - **Métodos Construtores:** métodos particulares que têm como objetivo criar objetos da classe
 - **Métodos:** definições de operações sobre os objetos
 - Funções
 - Procedimentos

ATRIBUTOS

- Atributos são **variáveis** cujos valores **caracterizam um objeto**
 - Representam o **estado** do objeto
 - Cada objeto guarda **valores** para os seus atributos
- Exemplos:
 - Objetos `Point` (ponto cartesiano) têm `x` e `y (int)` como atributos
 - Objetos `BinaryImage` têm uma matriz de booleanos (`boolean[] []`) como atributo

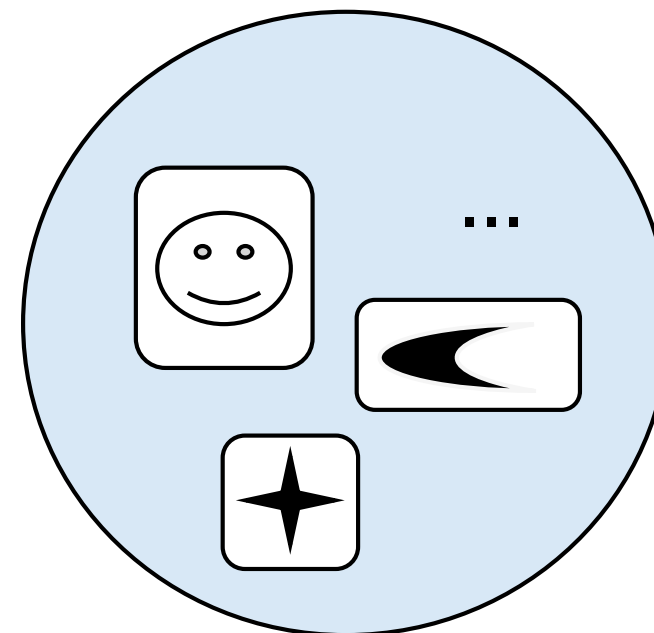
ATRIBUTOS

```
class Point {  
    int x;  
    int y;  
    ...  
}
```



Objetos do
tipo Point

```
class BinaryImage {  
    boolean[][] pixels;  
    ...  
}
```



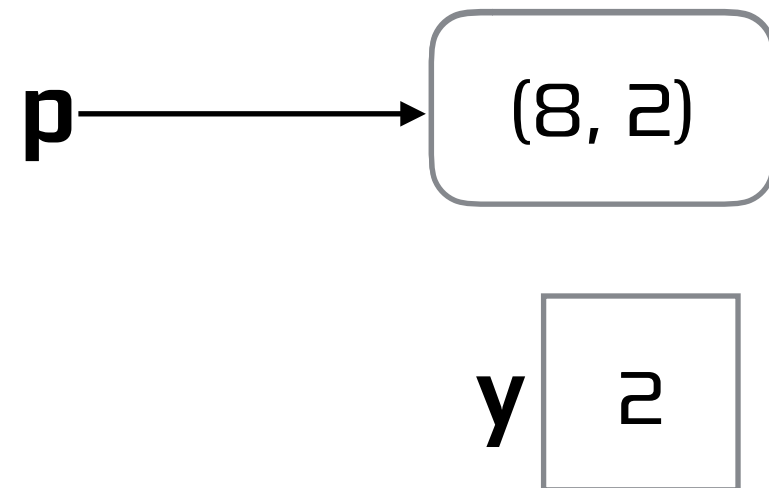
Objetos do tipo
BinaryImage

ATRIBUTOS: ACESSO

```
class Point {  
    int x;  
    int y;  
    ...  
}
```

```
Point p = new Point(1, 2);  
p.x = 8;
```

```
int y = p.y;
```



MÉTODOS CONSTRUTORES

- Um método **construtor** de uma classe é um método particular cujo propósito é **criar objetos** dessa classe
- O papel de um construtor é **inicializar os atributos do objeto criado**
- Podem haver **vários construtores** numa mesma classe, porém com parâmetros diferentes
- Caso não sejam definidos construtores numa classe, existe **por omissão um construtor sem parâmetros**

MÉTODOS CONSTRUTORES

```
class Point {  
    int x;  
    int y;  
  
    Point() {  
        x = 0;  
        y = 0;  
    }  
  
    Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```

Os construtores têm sempre o nome da classe!

```
Point p1 = new Point();
```

```
Point p2 = new Point(6, 7);
```

p1 → (0, 0)

p2 → (6, 7)


this refere-se ao próprio objeto. Nesta situação serve para desambiguar entre o nome dos parâmetros e nome dos atributos.

INVOCACÃO INTERNA DE CONSTRUTORES

```
class Point {  
    int x;  
    int y;
```

```
    Point() {  
        x = 0;  
        y = 0;  
    }
```

```
    Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }
```



```
Point() {  
    this(0, 0);  
}
```

A utilização de **this** é particularmente útil quando um construtor é um **caso especial** de outro.

A invocação de um construtor dentro de um construtor tem **obrigatoriamente** que ser a **primeira instrução**.

ATRIBUTOS CONSTANTES

No caso de classes de **objetos imutáveis**, é útil definir os atributos como constantes. Neste caso, é garantido que é feita exatamente uma atribuição de valor ao atributo (a qual tem necessariamente que ocorrer no construtor).

```
class Point {  
    final int x;  
    final int y;  
    ...  
    Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    ...  
}
```

MÉTODOS

As **operações** disponíveis num objeto são definidas em **métodos**, os quais têm acesso aos atributos.

```
class BinaryImage {  
  
    boolean[][] pixels;  
  
    ...  
  
    boolean isBlack(int x, int y) {  
        return !pixels[y][x];  
    }  
  
    void setWhite(int x, int y) {  
        pixels[y][x] = true;  
    }  
  
}
```

MÉTODO DE CLASSE *vs* MÉTODO DE INSTÂNCIA

```
class ImageUtils {  
  
    static void invert(BinaryImage img) {  
        for(int y = 0; y != img.getHeight(); y++) {  
            for(int x = 0; x != img.getWidth(); x++) {  
                if(img.isBlack(x, y))  
                    img.setWhite(x, y);  
                else  
                    img.setBlack(x, y);  
            }  
        }  
    }  
}  
  
ImageUtils.invert(img);
```

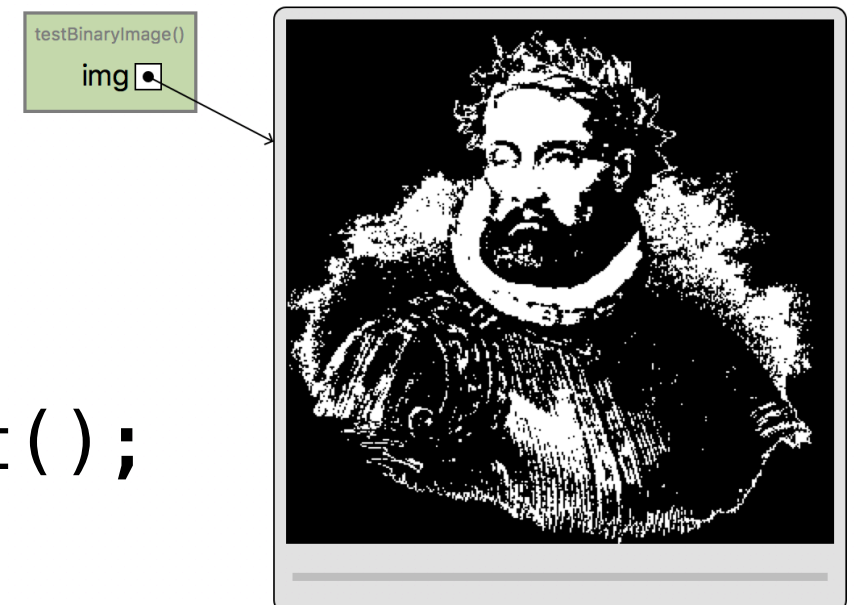
testBinaryImage()
img




MÉTODO DE CLASSE *vs* MÉTODO DE INSTÂNCIA

```
class BinaryImage {  
  
    boolean[][] pixels;  
    ...  
  
    void invert() {  
        for(int y = 0; y != pixels.length; y++)  
            for(int x = 0; x != pixels[y].length; x++) {  
                pixels[y][x] = !pixels[y][x];  
            }  
    }  
}
```

```
img.invert();
```




CLASSE ESTÁTICA *vs* CLASSE DE OBJETOS



```
class BinaryImage {  
  
    ...  
  
    boolean isBlack(...) {  
        ...  
    }  
  
    void setWhite(...) {  
        ...  
    }  
  
}
```

Classe (de objetos)



```
class ImageUtils {  
  
    static void invert(BinaryImage img) {  
        ...  
    }  
  
    static BinaryImage scale(BinaryImage img) {  
        ...  
    }  
  
    ...  
  
}
```

Classe estática

A RETER

- Classes de objectos
 - Definição
 - Atributos
 - Construtores
 - Métodos
- Métodos de classe vs métodos de instância
- Classe estática vs classe *de* objectos

