OBJECTOS

PROGRAMAÇÃO ORIENTADA POR OBJECTOS

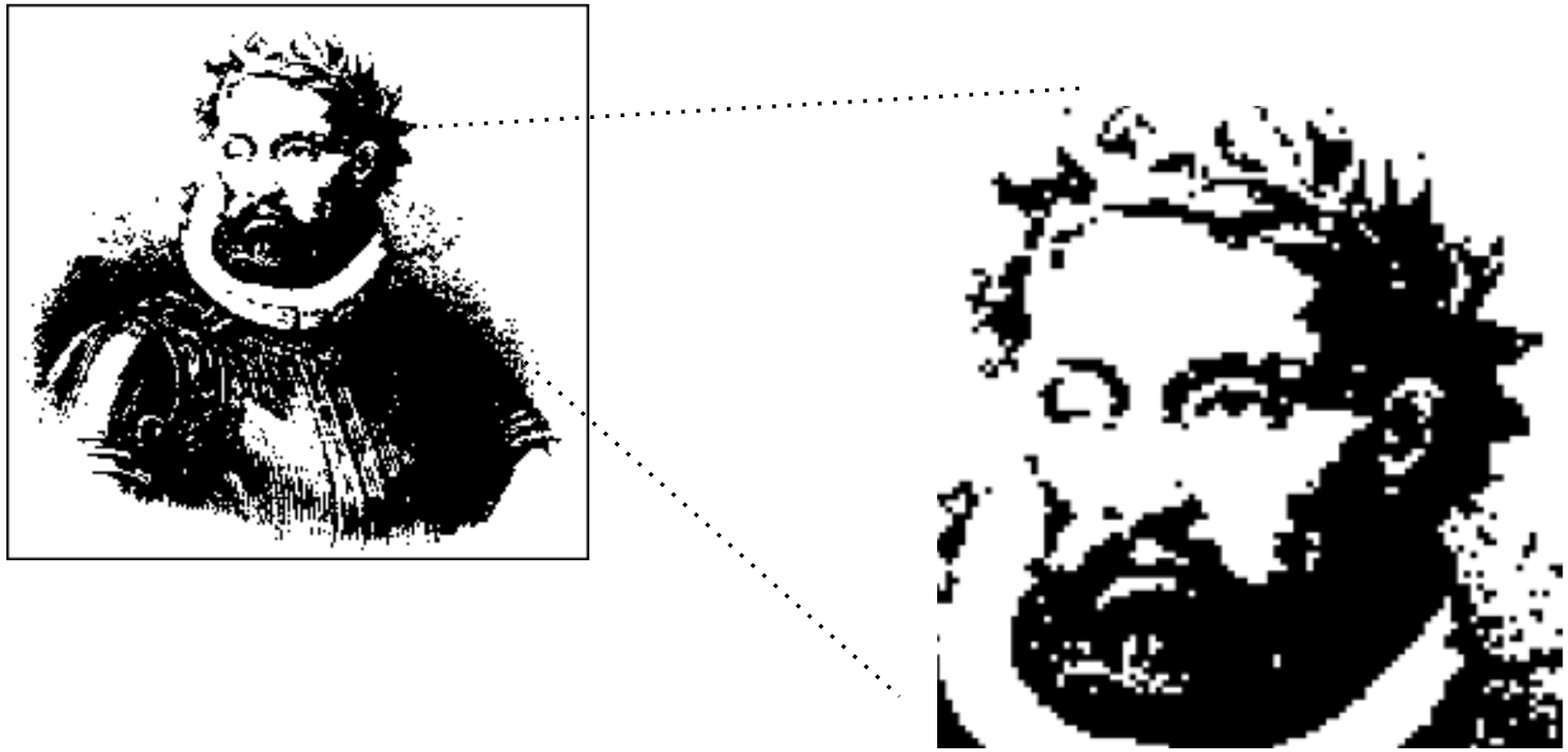
- O paradigma de programação **dominante** hoje em dia é o da **programação orientada por objetos**.

- Neste paradigma, o código manipula **objetos com estado e comportamento próprios**. Um objeto representa tipicamente uma **entidade** do mundo real (física ou não).

- O universo dos **objetos de determinado tipo** é designado por **classe de objetos** (descrição dos objectos de um dado tipo).

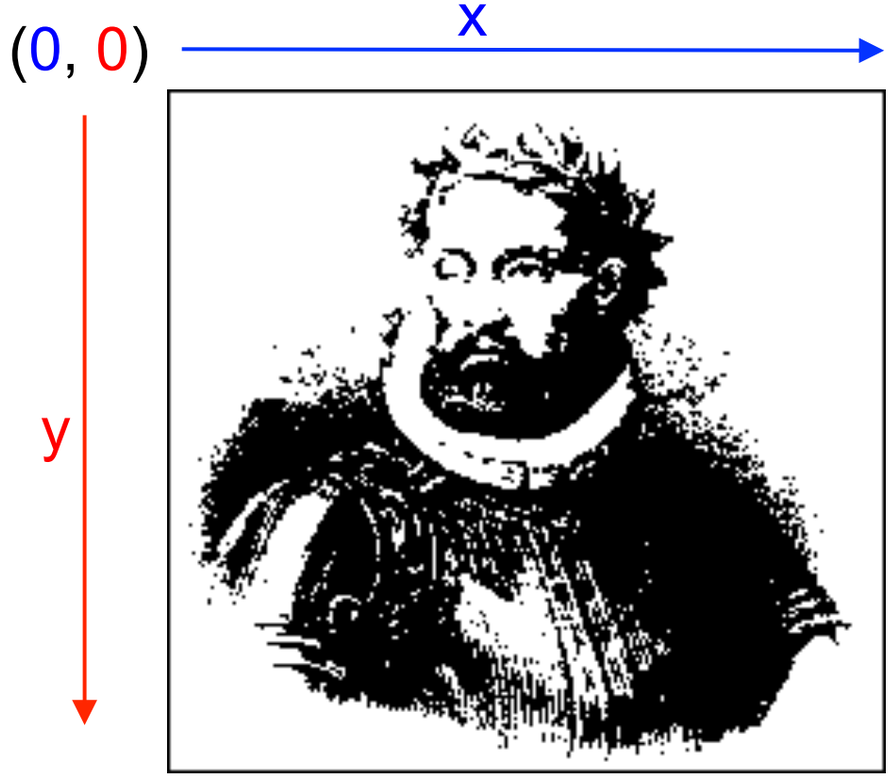
OBJECTOS: IMAGENS BINÁRIAS

Uma imagem binária é uma imagem onde cada pixel só tem dois valores possíveis (p.e. branco ou preto).



IMAGENS: COORDENADAS DE PÍXEIS

A coordenada (0, 0) corresponde ao píxel do canto superior esquerdo. À medida que o valor de ***x*** aumenta, deslocamo-nos para a direita na imagem e à medida que o valor de ***y*** aumenta deslocamo-nos para baixo na imagem.



IMPORTAÇÃO DE CLASSES

Para utilizar classes não disponíveis por omissão, torna-se necessário importá-las. Para tal, adiciona-se uma declaração de importação no cabeçalho do ficheiro `.java`, contendo o nome completo da classe:

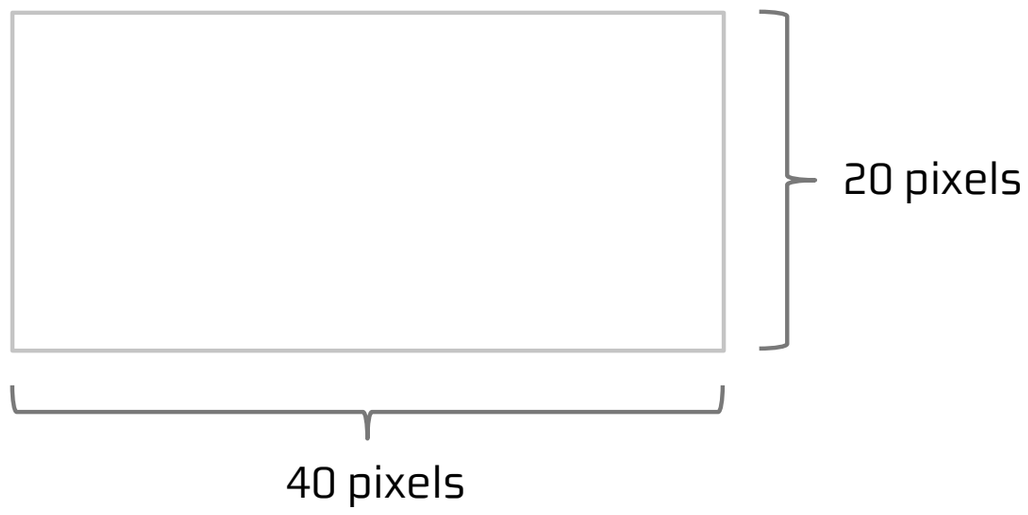
```
import aguiaj.iscte.BinaryImage;

class MyClass {

    // ... utilização da classe BinaryImage ...

}
```

CRIAÇÃO DE OBJECTOS

Na sua forma essencial, a criação de objetos é feita mediante um **método construtor**.

Nome da classe (tipo de objeto)

```
BinaryImage img = new BinaryImage(40, 20);
```



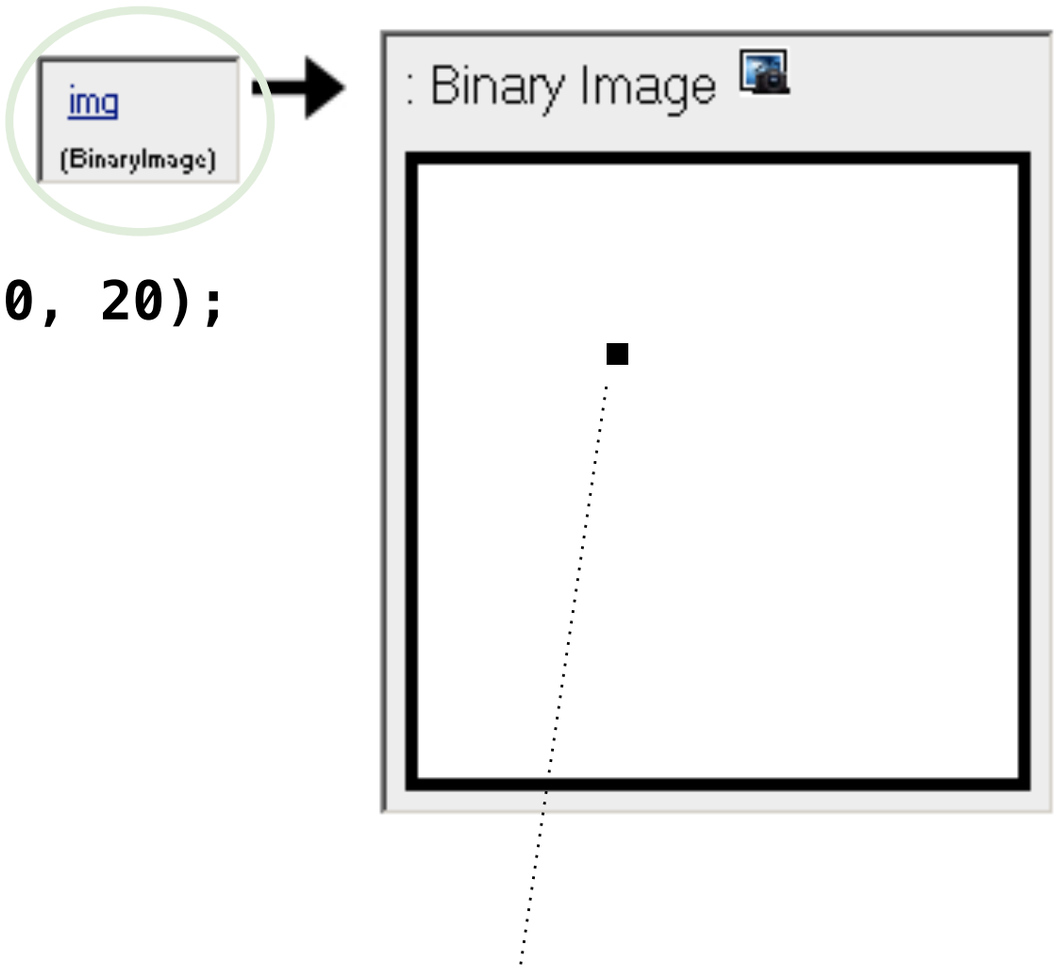
MANIPULAÇÃO DE OBJECTOS: OPERAÇÕES

- Os objetos são manipulados através da **invocação de operações**, as quais se podem caracterizar em:

- Funções (não modificam o objeto)

- Procedimentos (modificam o estado do objeto)

IMAGENS BINÁRIAS: OPERAÇÕES



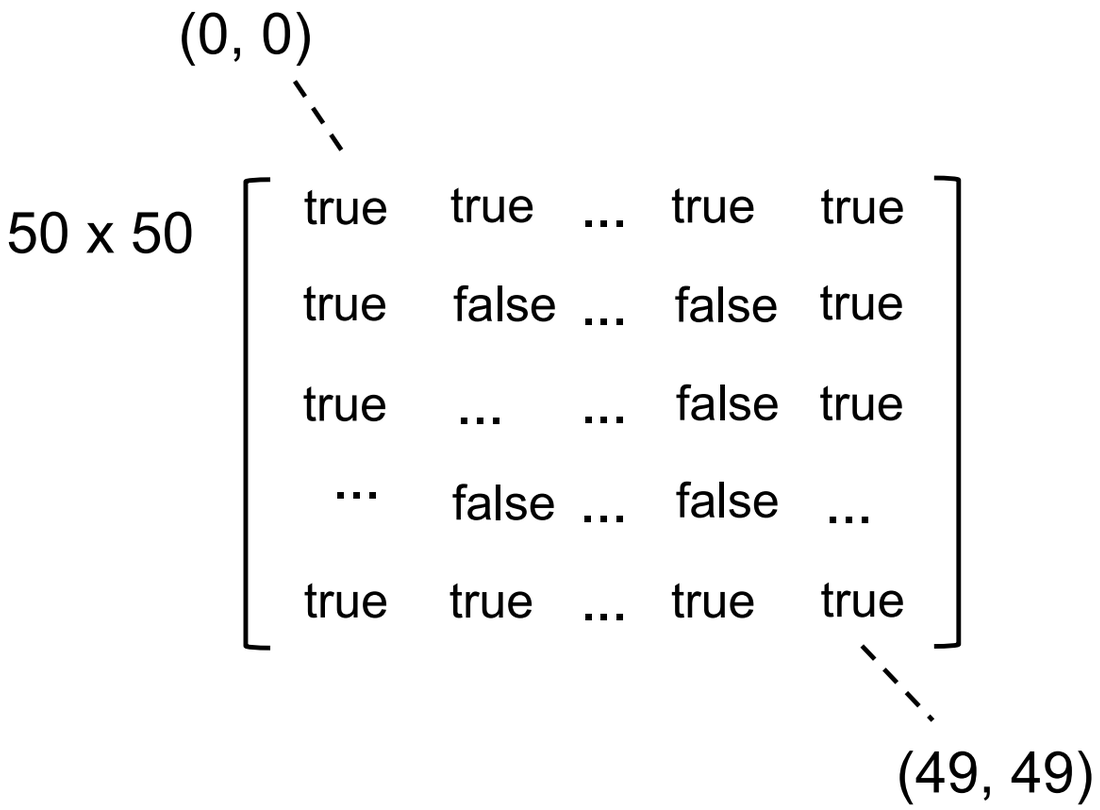

```
boolean black = img.isBlack(20, 20);

int width = img.getWidth();
```

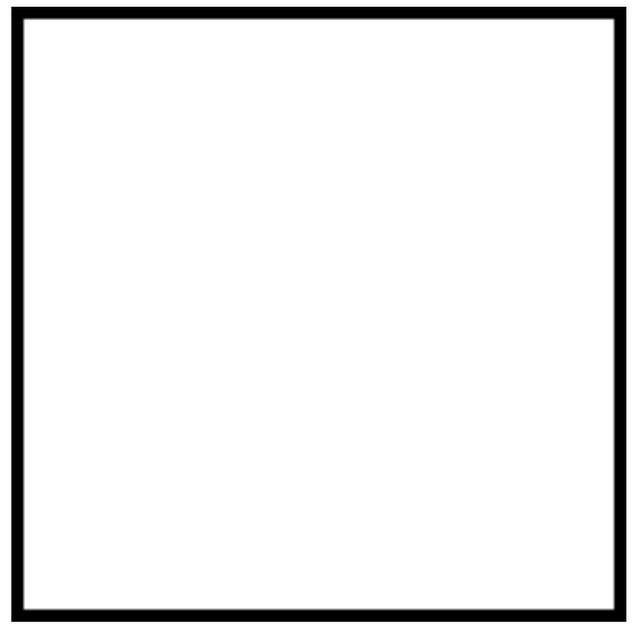
black true

width 50

```
img.setBlack(20, 20);
```

IMAGENS BINÁRIAS: ESTADO





IMAGENS BINÁRIAS

aguiaj.iscte.BinaryImage

Construção (cria uma imagem binária com dimensão *width* x *height*)

- `new BinaryImage(int width, int height)`

Funções:

- `int getWidth()`
  devolve a largura da imagem

- `int getHeight()`
  devolve a altura da imagem

- `boolean isBlack(int x, int y)`
  devolve verdadeiro caso o pixel na coordenada (*x*, *y*) seja preto, falso caso contrário

Procedimentos:

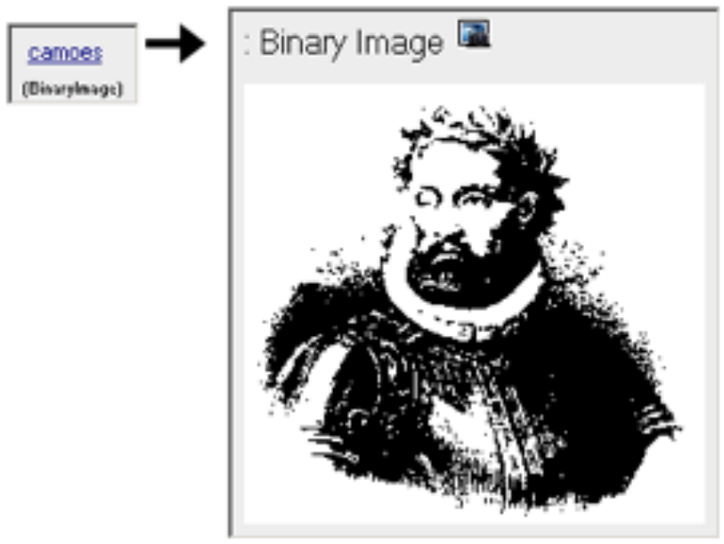
- `void setBlack(int x, int y)`
  altera o pixel na coordenada (*x*, *y*) para preto

- `void setWhite(int x, int y)`
  altera o pixel na coordenada (*x*, *y*) para branco

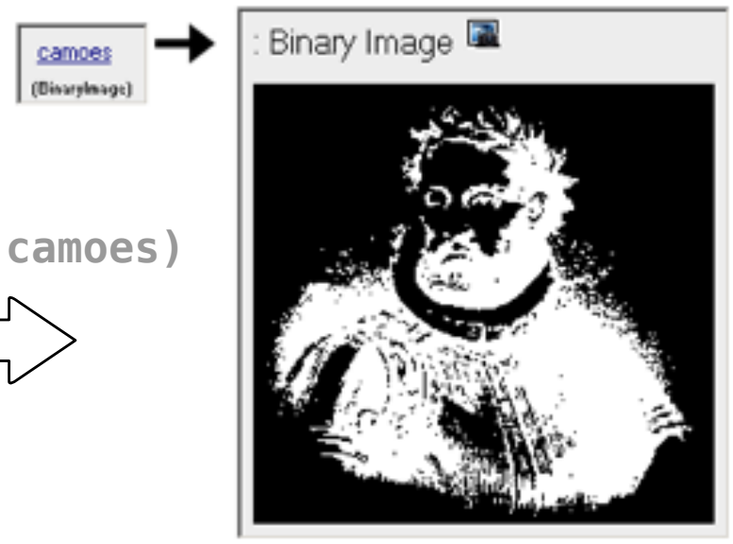
IMAGENS BINÁRIAS: EXEMPLO

Inverter a cor dos píxeis (procedimento)

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

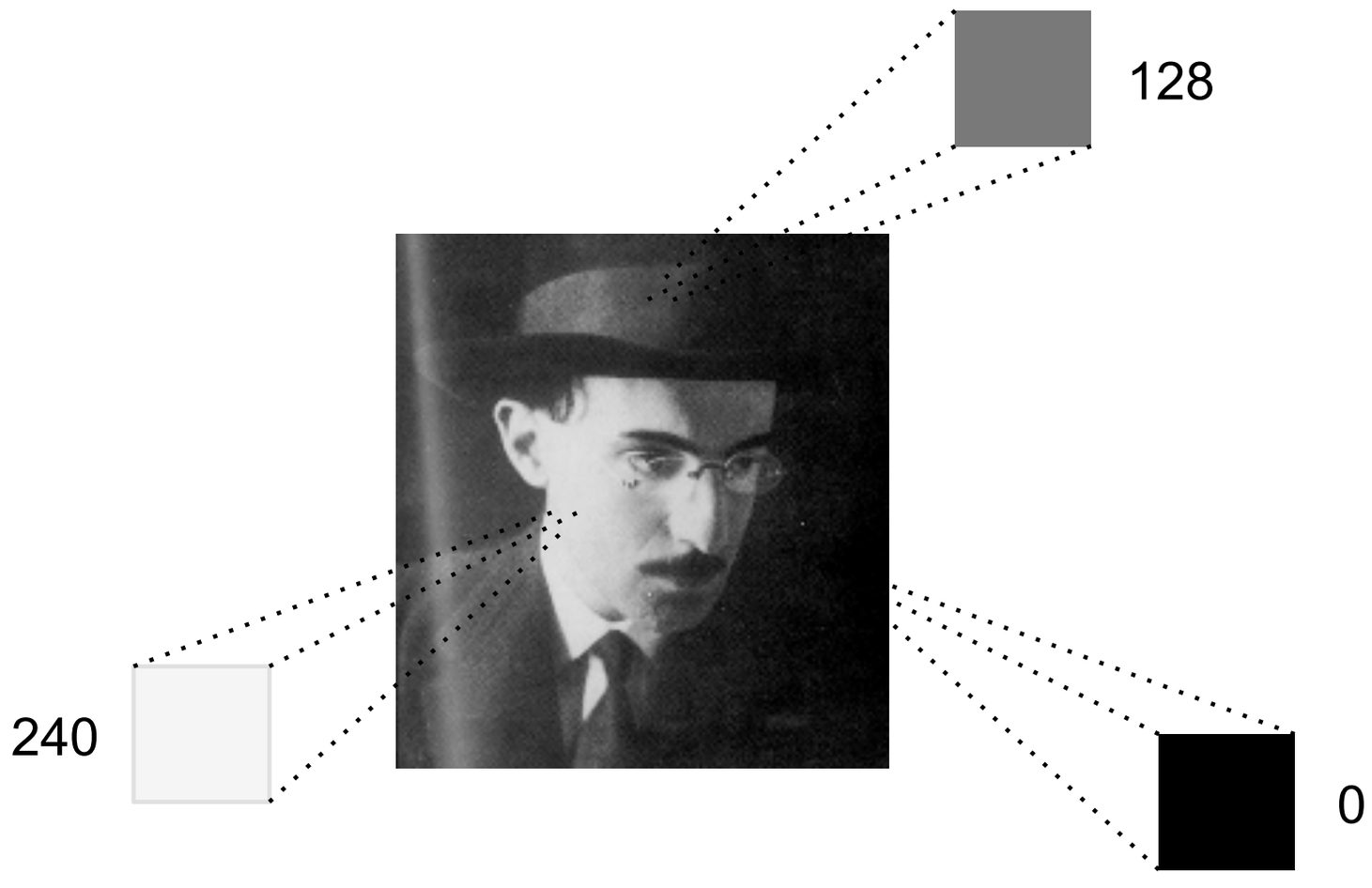


`invert(camoes)`



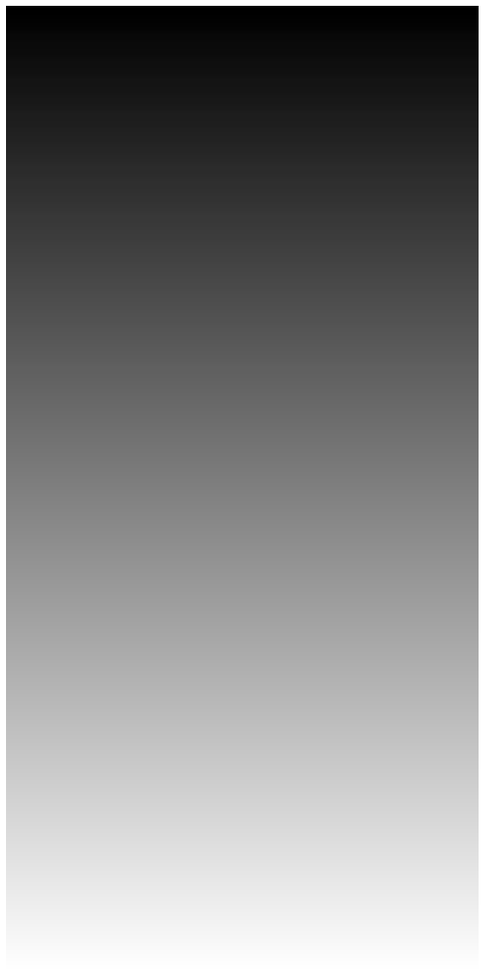
OBJECTOS: IMAGENS A PRETO E BRANCO (TONS DE CINZENTO)

Numa imagem a preto e branco cada pixel é um tom de cinzento, que pode ser representado numericamente com um valor entre 0 (preto) e 255 (branco).



IMAGENS A PRETO E BRANCO: ESTADO

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 1 & \dots & 1 \\ 2 & 2 & 2 & 2 & \dots & 2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$



IMAGENS A PRETO E BRANCO

aguiaj.iscte.GrayscaleImage

Construção (cria uma imagem binária com dimensão *width* x *height*)

- `new GrayscaleImage(int width, int height)`

Funções:

- `int getWidth()`
  devolve a largura da imagem

- `int getHeight()`
  devolve a altura da imagem

- `int getGraytone(int x, int y)`
  devolve o tom de cinzento do pixel na coordenada (*x*, *y*)

Procedimentos:

- `void setGraytone(int x, int y, int tone)`
  altera o pixel na coordenada (*x*, *y*) para o tom de cinzento *tone*

IMAGENS A PRETO E BRANCO: EXEMPLO

Inverter a cor dos píxeis (função)

```
static GrayscaleImage invert(GrayscaleImage img) {
    GrayscaleImage inv = new GrayscaleImage(img.getWidth(),
                                            img.getHeight());
    for(int x = 0; x != img.getWidth(); x++) {
        for(int y = 0; y != img.getHeight(); y++) {
            inv.setGraytone(x, y, 255 - img.getGraytone(x, y));
        }
    }

    return inv;

}
```

```
GrayscaleImage xray = invert(pessoa);
```

A RETER

- Objectos
  - Criação
    - Construtor
  - Manipulação (operações)
    - Funções
    - Procedimentos
  - Estado
- Exemplos