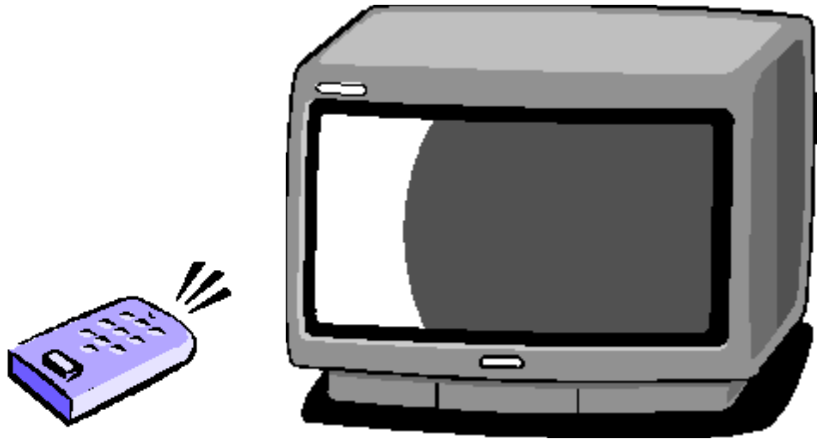
Programação Orientada a Objetos I

CÁSSIO CAPUCHO PEÇANHA – 06

Manipulação de Objetos

Manipulação de Objetos

- Analogia:
  - A TV é o objeto;
  - O controle é a referência;
  - Você só carrega a referência;
  - A referência pode existir sem o objeto.



Referência e objeto

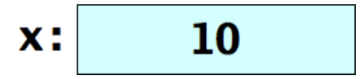
```
public class Coordenadas {
  int x;
  int y;
  int z;

  public static void main(String[] args) {
    Coordenadas coord;  // Só a referência.
                        // Não dá pra fazer nada...

    // Agora temos um objeto, podemos usá-lo.
    coord = new Coordenadas();
    coord.x = 10;
    coord.y = 15;
    coord.z = 18;
  }
}
```

Atribuição de valores

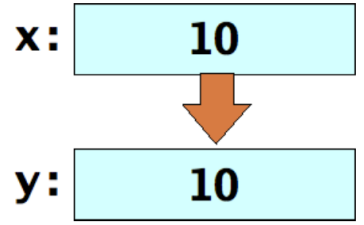
- Quando realizamos uma atribuição: **X=Y;**
- Java faz a cópia do valor da variável da direita para a variável da esquerda;
  - Para tipos primitivos, isso significa que alterações em x não implicam alterações em y;
  - Para objetos, como o que é copiado é a referência para o mesmo objeto, alterações no objeto que x referência altera o objeto que y referência, pois é o mesmo objeto!

Atribuição de valores primitivos

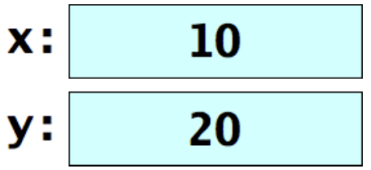
```
int x = 10;
```




```
int y = x;
```

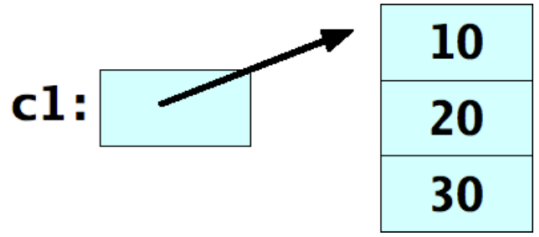



```
y = 20;
```

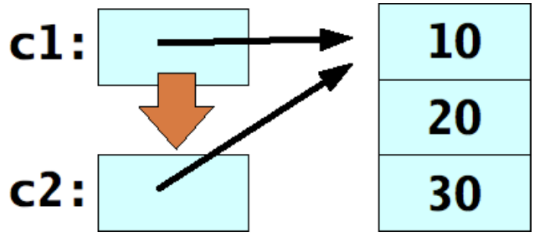


Atribuição de objetos

```
Coordenada c1;
c1 = new Coordenada();
c1.x = 10;
c1.y = 20;
c1.z = 30;
```




```
Coordenada c2;

// Erro comum:
// c2 = new Coordenada();

c2 = c1;
```



Comparações entre objetos

```
public class Comparacoes {
  public static void main(String[] args) {
    Coordenadas c1 = new Coordenadas();
    c1.x = 10; c1.y = 15; c1.z = 20;

    Coordenadas c2 = new Coordenadas();
    c2.x = 10; c2.y = 15; c2.z = 20;

    // O que imprime?
    System.out.println(c1 == c2);
  }
}
```

false

Comparações entre objetos

```
public class Comparacoes {
  public static void main(String[] args) {
    Coordenadas c1 = new Coordenadas();
    c1.x = 10; c1.y = 15; c1.z = 20;

    Coordenadas c2 = c1;
    c2.x = 11; c2.y = 16; c2.z = 21;

    // O que imprime?
    System.out.println(c1 == c2);
  }
}
```

true

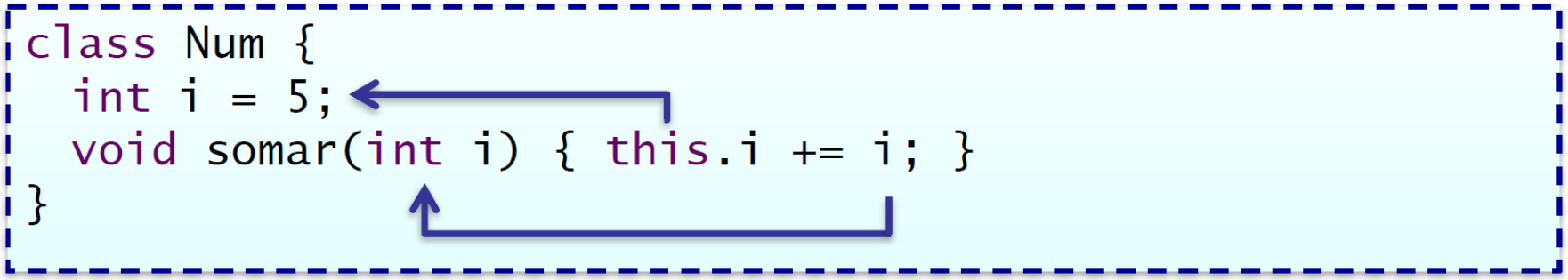
A palavra reservada this

- Não é necessário usar **this** quando acessamos membros do objeto de dentro do mesmo.

```
class Num {
  int i = 5;
  void somar(int j) { this.i += j; }
}
```

A palavra reservada this

- **this** pode ser usado para diferenciar um atributo do objeto de um parâmetro do método:



- Neste caso, o **this** é necessário!

Valores default para atributos

- Um atributo pode ser inicializado:

```
class Conta {
  int numero;    // 0
  String dono;   // null
  double saldo; // 0.0
  double limite = 1000.0;
}
```

- Quando não inicializamos explicitamente, um valor *default* é atribuído a ele:

Tipo	Valor
boolean	false
char	'\u0000'
byte	(byte) 0
short	(short) 0

Tipo	Valor
int	0
long	0l
float	0.0f
double	0.0

Implementando associações entre classes

- Atributos podem ser referências para objetos de outras classes (ou da mesma classe):

```
class Conta {
  int numero;
  double saldo;
  double limite = 1000.0;
  Cliente titular;
}

class Cliente {
  String nome;
  String sobrenome;
  String cpf;
}

// ...
```

Implementando associações entre classes

- Atributos podem ser referências para objetos de outras classes (ou da mesma classe):

```
public class Teste
  public static void main(String[] args) {
     Cliente larry = new Cliente();
     larry.nome = "Larry";
     larry.sobrenome = "Ellison";

     Conta conta = new Conta();
     conta.saldo = 50_400_000_000.0;
     conta.titular = larry;

     // Navegando no grafo de objetos...
     System.out.println(conta.titular.nome);
  }
}
```