Curso Programação Orientada a Objetos com Java

Capítulo: Comportamento de memória, arrays, listas

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Tipos referência vs. tipos valor

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Classes são tipos referência

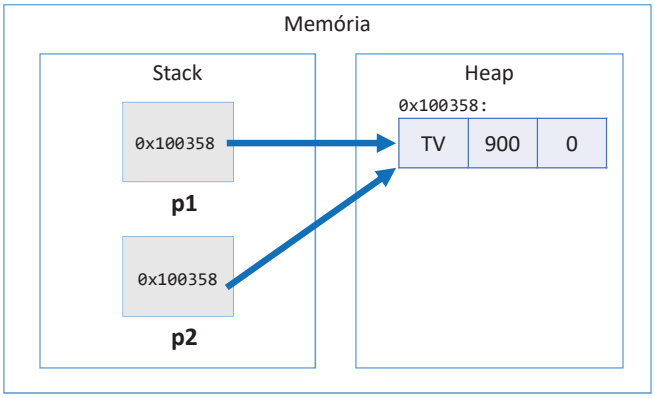
Variáveis cujo tipo são classes não devem ser entendidas como caixas, mas sim "tentáculos" (ponteiros) para caixas

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = p1;
```

p2 = p1;
"p2 passa a apontar para onde p1 aponta"

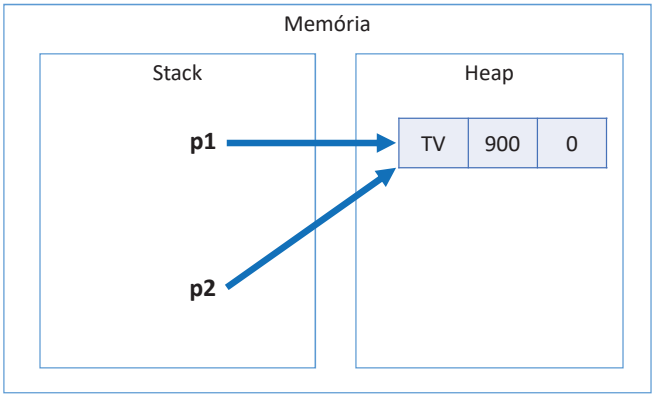


Desenho simplificado

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = p1;
```



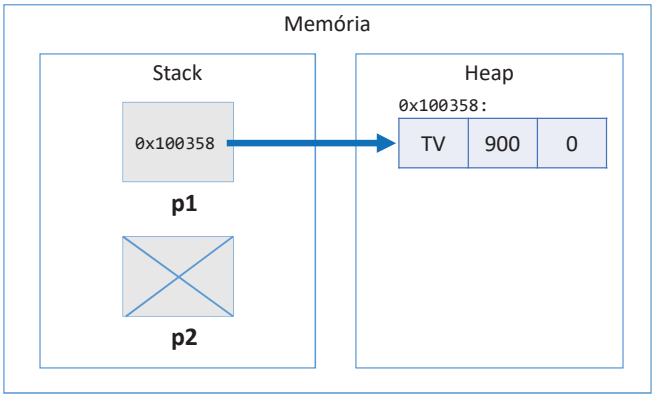
Valor "null"

Tipos referência aceitam o valor "null", que indica que a variável aponta pra ninguém.

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = null;
```



Tipos primitivos são tipos valor

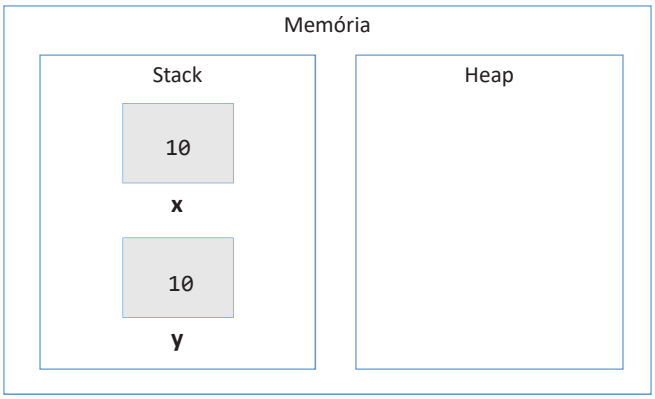
Em Java, tipos primitivos são tipos valor. Tipos valor são CAIXAS e não ponteiros.

```
double x, y;

x = 10;

y = x;
```

y = x;
"y recebe uma CÓPIA de x"



Type	Contains	Default	Size	Range
`boolean`	`true` or `false`	`false`	1 bit	NA
`char`	Unicode character	`\u0000`	16 bits	`\u0000` to `\uFFFF`
`byte`	Signed integer	0	8 bits	-128 to 127
`short`	Signed integer	0	16 bits	-32768 to 32767
`int`	Signed integer	0	32 bits	-2147483648 to 2147483647
`long`	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
`float`	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
`double`	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

Tipos primitivos e inicialização

- Demo:

```
int p;
System.out.println(p); // erro: variável não iniciada

p = 10;
System.out.println(p);
```

Valores padrão

- Quando alocamos (new) qualquer tipo estruturado (classe ou array), são atribuídos valores padrão aos seus elementos
  - números: 0
  - boolean: false
  - char: caractere código 0
  - objeto: null

```
Product p = new Product();
```

Tipos referência vs. tipos valor

CLASSE	**TIPO PRIMITIVO**
Vantagem: usufrui de todos recursos OO	Vantagem: é mais simples e mais performático
Variáveis são ponteiros	Variáveis são caixas
Objetos precisam ser instanciados usando new, ou apontar para um objeto já existente.	Não instancia. Uma vez declarados, estão prontos para uso.
Aceita valor null	Não aceita valor null
Y = X; "Y passa a apontar para onde X aponta"	Y = X; "Y recebe uma cópia de X"
Objetos instanciados no heap	"Objetos" instanciados no stack
Objetos não utilizados são desalocados em um momento próximo pelo garbage collector	"Objetos" são desalocados imediatamente quando seu escopo de execução é finalizado