

2 - A tecnologia Java

Uma ideia base da linguagem JAVA é a de que,

“um programa em JAVA deve poder ser executado em qualquer tipo de computador sem ter que ser alterado ou compilado novamente”.

Em Java o código fonte da aplicação é compilado para uma representação intermédia, independente do sistema de execução e da arquitectura da máquina.

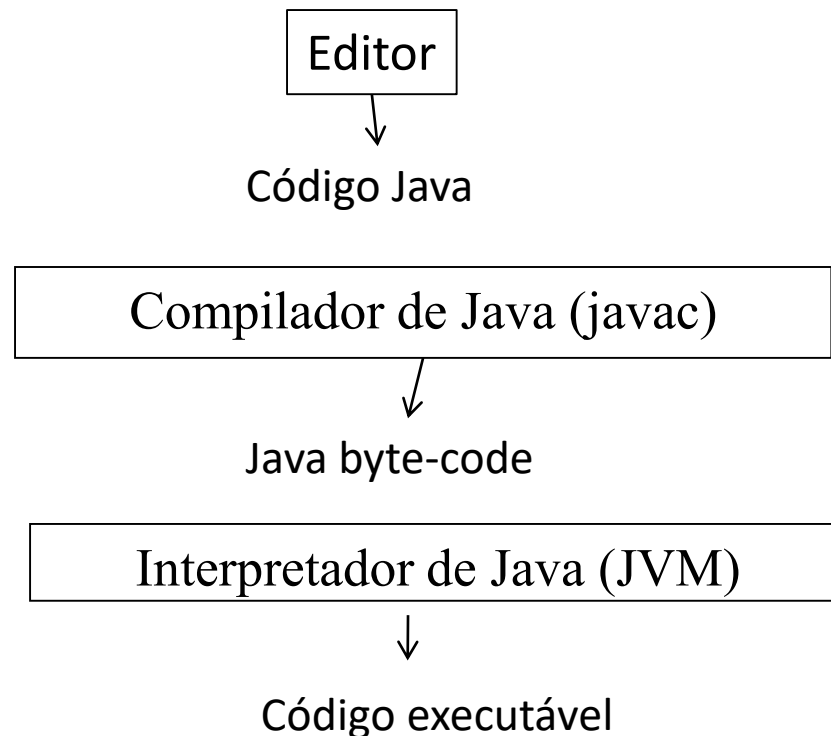
Essa representação intermédia é designada por byte-code.

De seguida este código pode ser interpretado sobre o ambiente de cada máquina específica.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Para cada plataforma em que se pretende executar um programa em Java é necessário um “motor de execução” designado por Java Virtual Machine (JVM).

A JVM recebe byte-code e transforma-o em instruções executáveis na máquina onde o ambiente Java é instalado.



Programação Orientada a Objectos - P. Prata, P. Fazendeiro

JVM – Java Virtual Machine

JRE – Java Runtime Environment

JDK – Java Development Kit

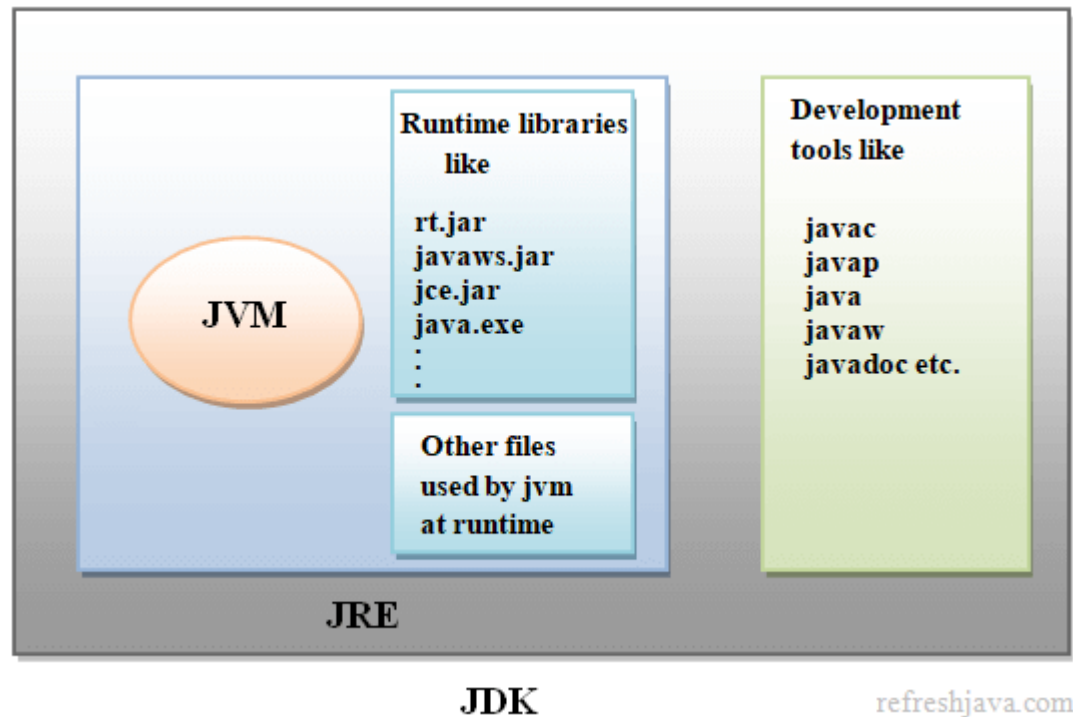


Imagem retirada de: <https://refreshjava.com/java/jdk-jre-jvm>

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

JDK	JRE	JVM
JDK is a software package.	JRE is also a software package.	JVM is not a separate software package, it's part of JRE.
Using JDK you can do both, compilation and execution of java programs.	Using JRE you can do only execution of java programs.	JVM is part of JRE, responsible for execution of your java programs.
JDK exists physically in your computer.	JRE also exists physically in your computer.	JVM does not exist physically in your computer.
You can download and install JDK in your computer.	JRE can also be downloaded and installed in your computer.	JVM comes as part of JRE.
JDK is a superset of JRE.	JRE is a subset of JDK but superset of JVM.	JVM is a subset of JRE.
JDK contains tools like <code>javac</code> , <code>javap</code> , <code>java</code> , <code>javadoc</code> etc.	JRE contains runtime tools like <code>java</code> , <code>javaw</code> etc.	JVM doesn't contains any such tools.

Retirado de: <https://refreshjava.com/java/jdk-jre-jvm>

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

JVM é uma especificação.

JRE e JDK são implementações.

Uma implementação é um programa que realiza os requisitos definidos na especificação.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

O JAVA pode ser usado para criar dois tipos de programas:

Aplicações e Applets.

Aplicações Java são programas que após serem compilados apenas requerem uma JVM para serem interpretados e executados.

Applets são porções de código Java não executável por si próprio. Requerem a existência de um “browser” que incorpore e execute a JVM.

A linguagem Java

A linguagem Java foi criada no início dos anos 90 por uma equipa da Sun Microsystems liderada por James Gosling.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Aquela equipa ao pretender desenvolver software para pequenos equipamentos de electrónica deparou-se com dificuldades em encontrar uma linguagem adequada para criar programas que pudessem ser executados em dispositivos com arquitecturas muito diferentes.

A tentativa de criar uma linguagem independente da arquitectura a programar deu origem ao Java (inicialmente designada por Oak).

Java é uma linguagem Orientada a Objectos de uso geral

Java segue a sintaxe e a estrutura geral do C++

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

A JVM está embutida nos principais “Web-browsers”

Java começou por ser uma marca comercial da Sun Microsystems

Em 2010 a Sun foi comprada pela Oracle Corporation

Java é também um esforço de marketing bem conseguido pela Sun

O nome “Java”

Java é o nome de uma ilha da Indonésia
(130.000Km², 145.000.000 de habitantes, capital Jakarta)

Java é o nome de um tipo de café criado em Java e ilhas vizinhas

Java é uma palavra inventada pela Sun para substituir o nome inicial da linguagem (Oak)

Classes em Java

- Em Java os programas são constituídos por diversas classes.
- Algumas classes são escritas por nós, outras fazem parte de bibliotecas.
- As classes são agrupadas em pacotes (packages).
- As classes possuem atributos (variáveis) e métodos (funções).

Classes em Java

- As classes são tipos de dados.
- Em Java cada objecto pertence a um determinado tipo. O tipo de um objecto é a sua classe.
- Algumas classes são *Applets* e podem ser executadas num *browser*.

Um **construtor** é uma operação especial da classe.

- Serve para criar um objecto da classe.
- Todos os construtores de uma classe têm o mesmo nome que a classe.
- Os construtores são identificados e diferenciados pela sua lista de parâmetros.
- Um construtor é uma função em que não é declarado o tipo do resultado (o resultado é um objeto da classe).

Tipos referenciados – Objectos e Arrays

Designam-se por tipos referenciados as entidades (objetos ou arrays) que são accedidas através de uma variável que contém o seu endereço.

Note-se que, sendo estes objectos referenciados através de variáveis, atribuições entre estas variáveis assumem uma *semântica de apontadores*, ou seja, o que é atribuído ou manipulado são referências (isto é, endereços).

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Arrays

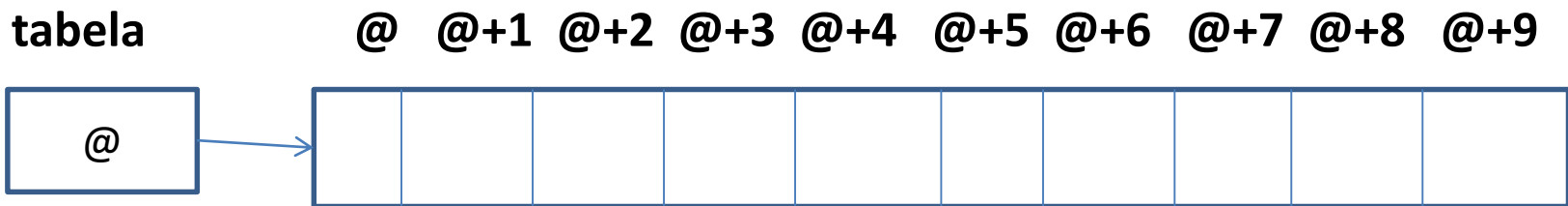
Os arrays são entidades *referenciadas* mas *não são objectos*

A criação de tabelas (arrays) assemelha-se à criação de objectos

```
int [] tabela;  
tabela = new int [10];
```

ou

```
int [] tabela = new int [10];
```



Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Os arrays são criados dinamicamente (em tempo de execução) e o seu espaço é automaticamente reaproveitado quando deixam de estar referenciados

Exemplos:

```
int lista[]; // declaração "à la C"
```

```
int[] lista; // declaração equivalente à anterior
```

```
int[] lista = { 10, 12, 33, 23, 56, 67, 89, 12 };  
/* declaração com inicialização */
```

```
int[] lista = new int[20];  
// array de inteiros com 20 componentes
```

```
byte[] pixels = new byte[600*800];  
/* pixels é um array de 480000 bytes */
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
int[] potencias2 = {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096};
```

```
String[] texto = new String[200];
```

```
/* texto é um array de 200 Cadeias de caracteres (ou Strings) */
```

```
String [] moedas = { "Dólar", "Yene", "Euro", "Florim" };
```

```
// String é um tipo de objecto (classe String) !!!
```

A multi-dimensionalidade é implementada como aninhamento
(array de arrays)

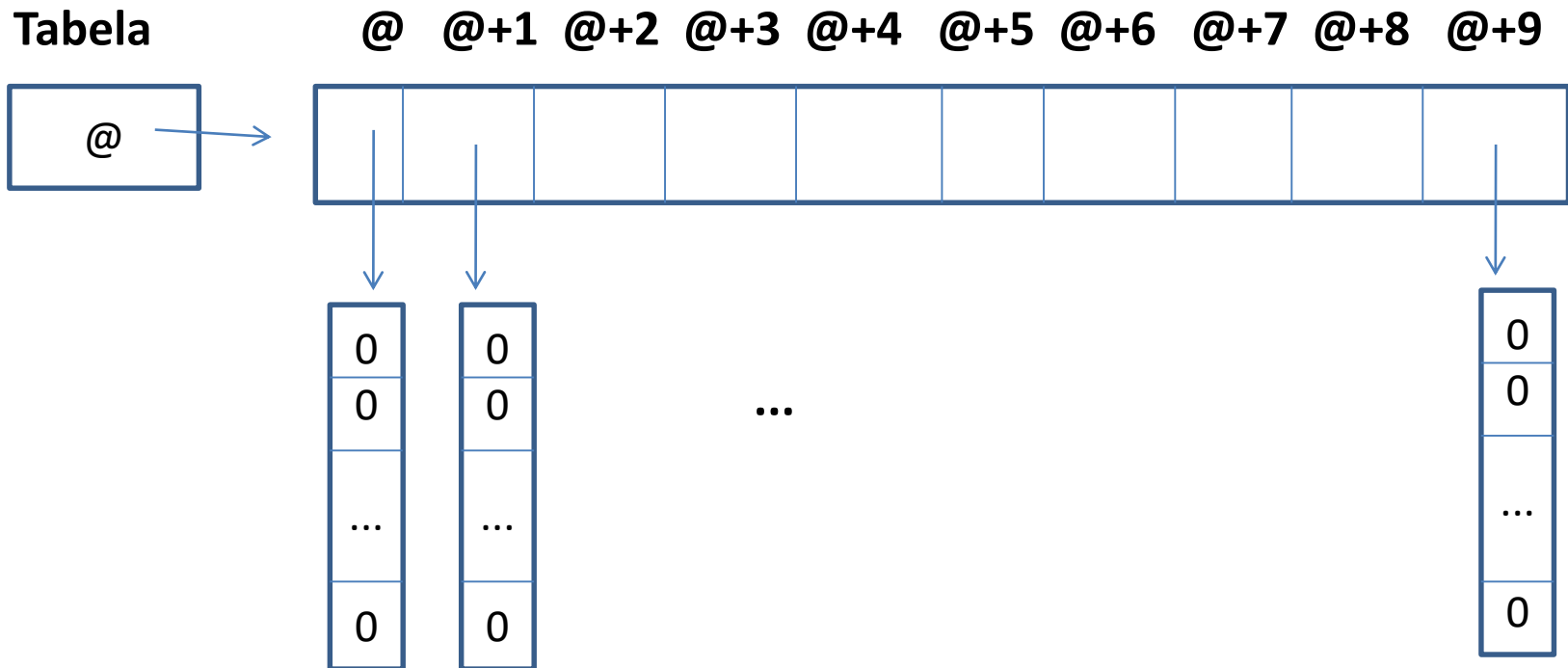
```
int [] [] tabela = new int[10] [20];
```

```
/*tabela é um array bidimensional de 10 linhas e 20 colunas*/
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

```
int tabela[] [] = new int[10] [20];
```

```
/* declaração equivalente à anterior */
```



Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

```
int[][] matriz = new int[3][]; ?
```

Neste caso, seria alocado um array de 3 posições, com referências de valor null, de momento, para cada um dos elementos a alocar nestas posições, sabendo-se, que tais elementos devem ser do tipo int[], ou seja, arrays de inteiros.

Os elementos do tipo int[] deste array podem ter dimensões distintas...

```
int[][] matriz = { {1,2}, {10, 20, 30}, {19, 2, 34, 21, 16}};
```

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Acesso aos elementos de um array

```
int[] apostaTotoloto = new int[6] ;
```

```
for (int i = 0; i < 6; i++) {  
    // alternativa -> for (int i = 0; i < apostaTotoloto.length; i++) {  
        apostaTotoloto[i] = (int) (Math.random() * 49 )+ 1;  
    }
```

```
for (int i = 0; i < apostaTotoloto.length; i++) {  
    System.out.println (apostaTotoloto[i]);  
}
```

O método `Math.random()` devolve um valor do tipo `double` pertencente ao intervalo `[0, 1[`.

...com `array.length` evita-se o acesso a elementos inexistentes
(**`ArrayIndexOutOfBoundsException`**)

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Exercícios:

- a) Declarar e alocar um array de inteiros, lista, de dimensão 20 e atribuir valores aleatórios entre 20 e 200 ao array usando a função `Math.random()` :
- b) Escrever na consola os valores do array lista;
- c) Calcular a soma dos elementos do array lista.
- d) Criar um novo array, y, que é calculado a partir do array lista, multiplicando cada elemento de lista por 0.5.
- e) Somar os arrays lista e y.
- f) Calcular o produto interno de lista por y.

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

a)

```
int lista[]; // declarar array
```

```
lista = new int [20]; // alocar memória
```

```
for (int i = 0; i < lista.length; i++) {
```

```
    // atribuir valores
```

```
    lista[i] = (int) ( Math.random () * 180 ) + 20;
```

```
    // como eliminar duplicados?
```

```
}
```

Que variáveis existem neste bloco de código, e qual o seu âmbito?

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

b) // escrever valores

```
for (int i = 0; i < lista.length; i++) {
```

```
    System.out.println(lista[i]);
```

```
}
```

Output de **uma** execução:

57

54

84

170

119

45

...

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

c) Somar elementos da lista

```
int soma = 0;
```

```
for (int i = 0; i < lista.length; i++) {
```

```
    soma = soma + lista[i];
```

```
}
```

```
System.out.println("Soma = " + soma);
```

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

d)

```
int lista[]; // declarar array
lista = new int [20]; // alocar memória
for (int i = 0; i < lista.length; i++) {
    lista[i] = (int) ( Math.random () * 180 ) + 20;
}
// criar nova lista com valores multiplicados por 0.5
double y [] = new double [20];

for (int i = 0; i < y.length; i++) {

    y[i] = lista[i] * 0.5;

}
```

- Quantas variáveis existem no neste bloco do código?

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

e) //somar arrays

```
double listaMaisY [] = new double [20];
```

```
for (int i = 0; i < y.length; i++) {
```

```
    listaMaisY[i] = lista [i] + y[i] ;
```

```
}
```

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Resolução:

f) Produto interno

```
double produtoInterno = 0 ;
```

```
for (int i = 0; i < y.length; i++) {
```

```
    produtoInterno = produtoInterno + lista[i] * y[i];
```

```
}
```

```
System.out.println( produtoInterno);
```

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Supondo:

```
int [] a = new int[20];  
for (int i = 0; i < lista.length; i++) {  a[i] = i; }
```

Como fazer uma copia da tabela **a**?

i) Assim?

```
int[] b;  
b = a;
```

ii) Ou assim?

```
int[] b = new int[20] ;  
for(int i = 0; i < a.length ; i++ ) b[i] = a[i] ;
```

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Cópia de uma tabela ?

i)

```
int [] b;
```

```
int [] a;
```

```
a = new int[5];
```

```
for(int i = 0; i < a.length ; i++ )
```

```
    a[i] = i * i ;
```

```
b = a;
```

Quantas variáveis existem e qual o seu valor? Represente graficamente.

Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Cópia de uma tabela ?

i)

```
int [] b;
```

```
int [] a;
```

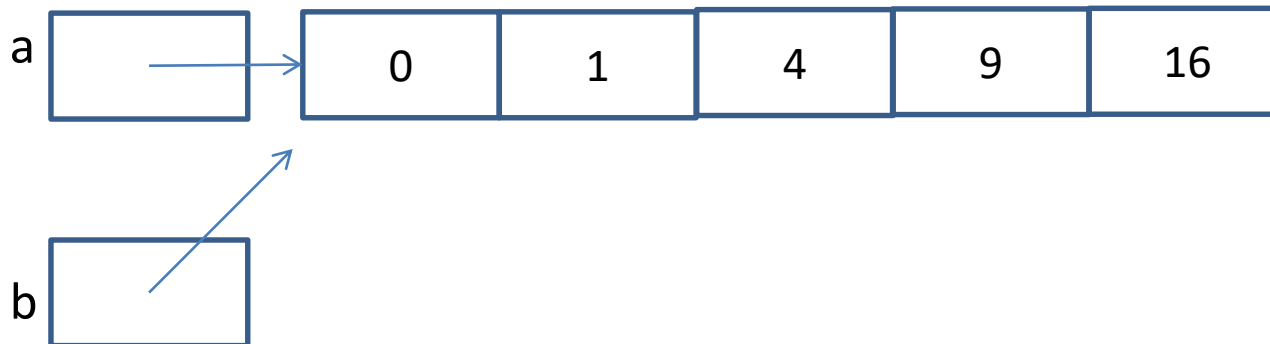
```
a = new int[5];
```

```
for(int i = 0; i < a.length ; i++ )
```

```
    a[i] = i * i ;
```

```
b = a;
```

Quantas variáveis existem e qual o seu valor? Represente graficamente.



Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

Cópia de uma tabela ?

ii)

```
int [] b;
```

```
int [] a;
```

```
a = new int[5];
```

```
for(int i = 0; i < a.length ; i++ )
```

```
    a[i] = i * i ;
```

```
b = new int[5] ;
```

```
for(int i = 0; i < a.length ; i++ ) b[i] = a[i] ;
```

Quantas variáveis existem e qual o seu valor? Represente graficamente.

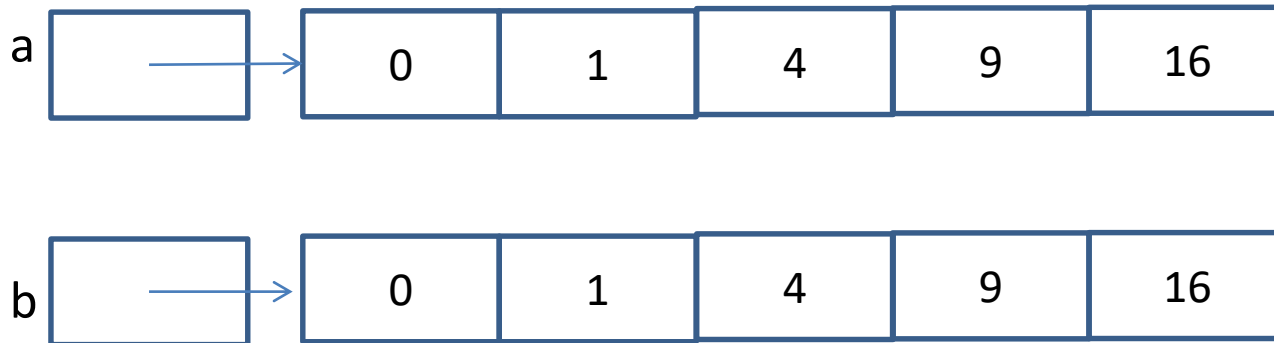
Programação *Orientada a Objectos* - P. Prata, P. Fazendeiro

ii)

```
b = new int[5] ;
```

```
for(int i = 0; i < a.length ; i++ ) b[i] = a[i] ;
```

Quantas variáveis existem e qual o seu valor? Represente graficamente.



Os objetos também são tipos de dados referenciados, isto é são entidades acedidas pelo seu endereço.

Suponhamos uma classe Veiculo definida, de uma forma simplificada, por:

```
public class Veiculo {  
    // variáveis ou atributos que representam um veículo  
    int pot;  
    int ano;  
  
    Veiculo ( int p , int a) { ← construtor  
        pot = p;  
        ano = a;  
    }  
    // outras funções ...  
}
```

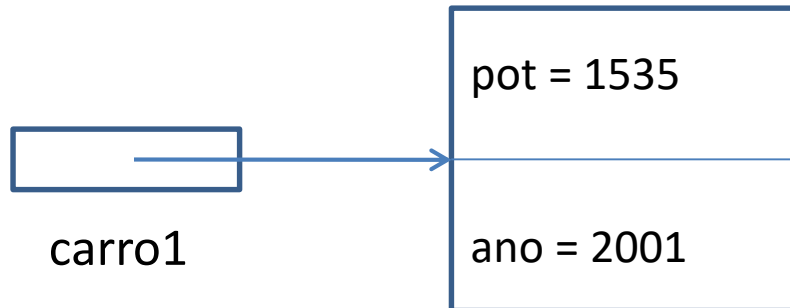
Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Após a definição da classe Veiculo, ela constitui um novo tipo de dados.

Podemos declarar variáveis do tipo Veiculo, que irão ser objetos

```
Veiculo carro1;           // é reservado um espaço de memória,  
                           // associado ao identificador carro1, e é-lhe  
                           // atribuído o valor null.
```

```
carro1 = new Veiculo(1535, 2001); // invocar o construtor;
```



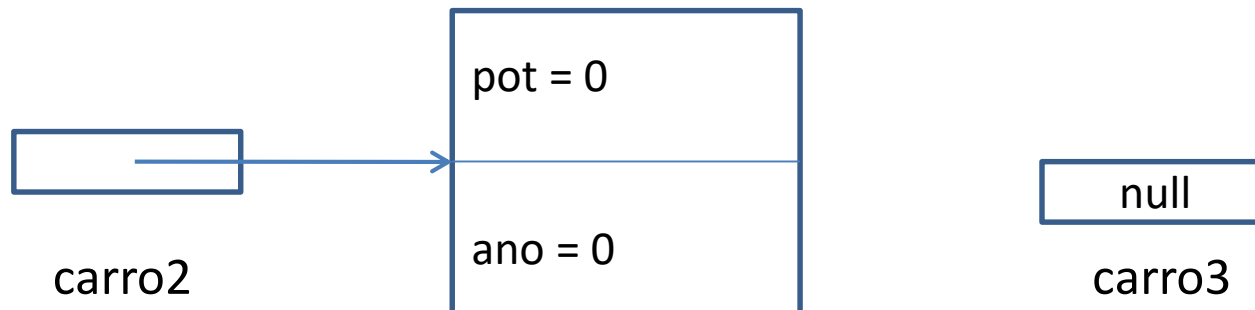
// é criado (instanciado) um objecto do tipo Veiculo, sendo alocada a memória para as variáveis de instância do objeto, e o endereço do objeto é colocado em carro 1.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

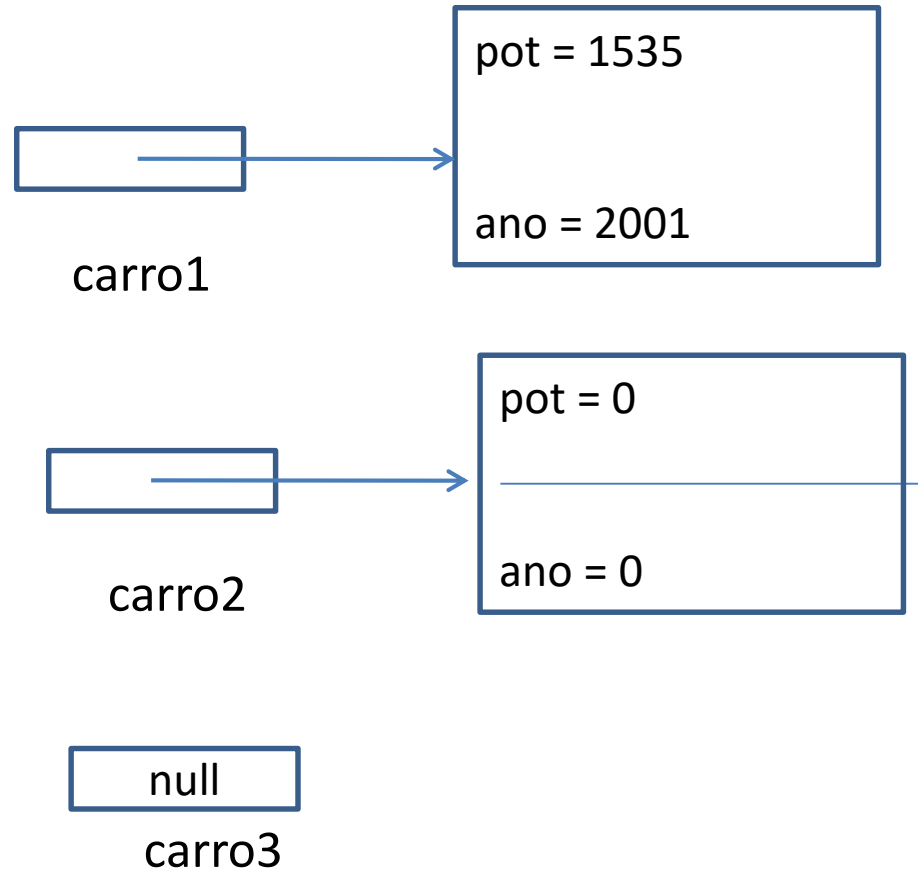
Suponhamos:

Veiculo carro2 = **new** Veiculo();

Veiculo carro3;

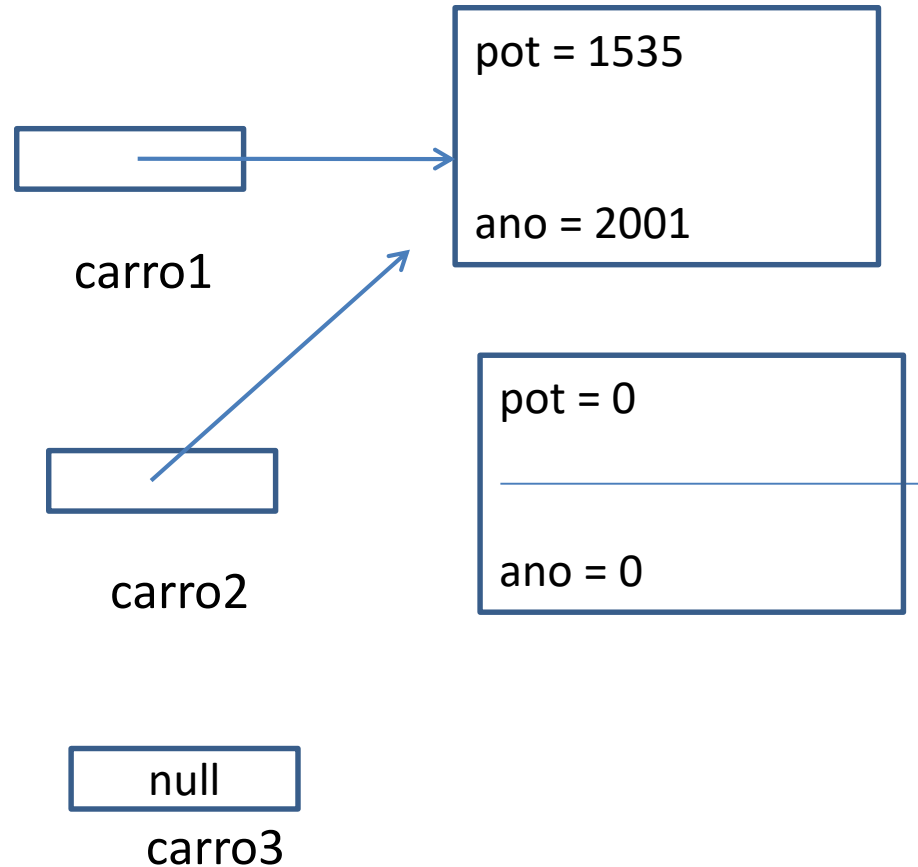


Programação Orientada a Objectos - P. Prata, P. Fazendeiro



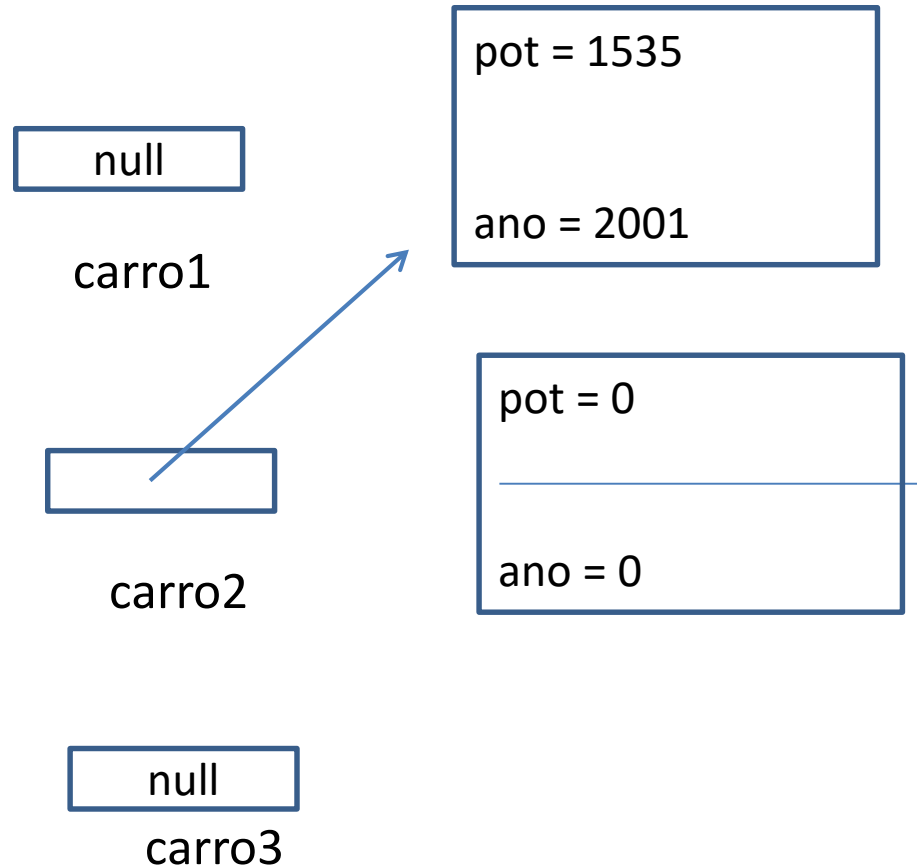
carro2 = carro1; ? O que acontece?

Programação Orientada a Objectos - P. Prata, P. Fazendeiro



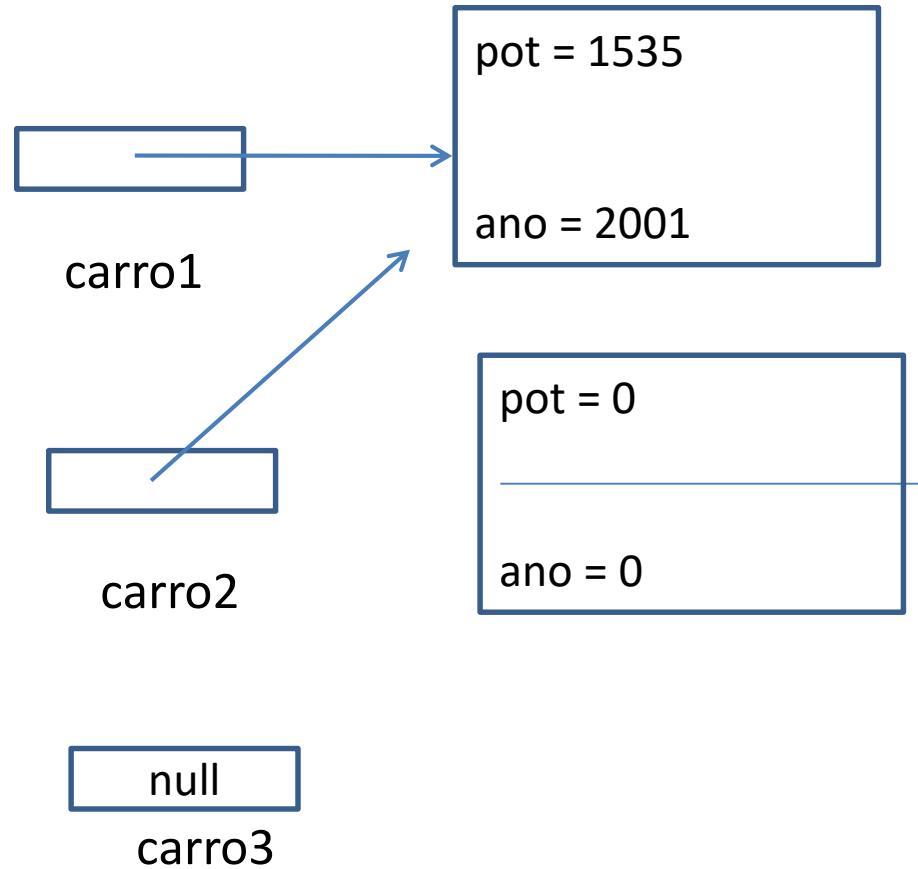
carro1 = carro3; ? O que acontece?

Programação Orientada a Objectos - P. Prata, P. Fazendeiro



carro1 = carro2; ? O que acontece?

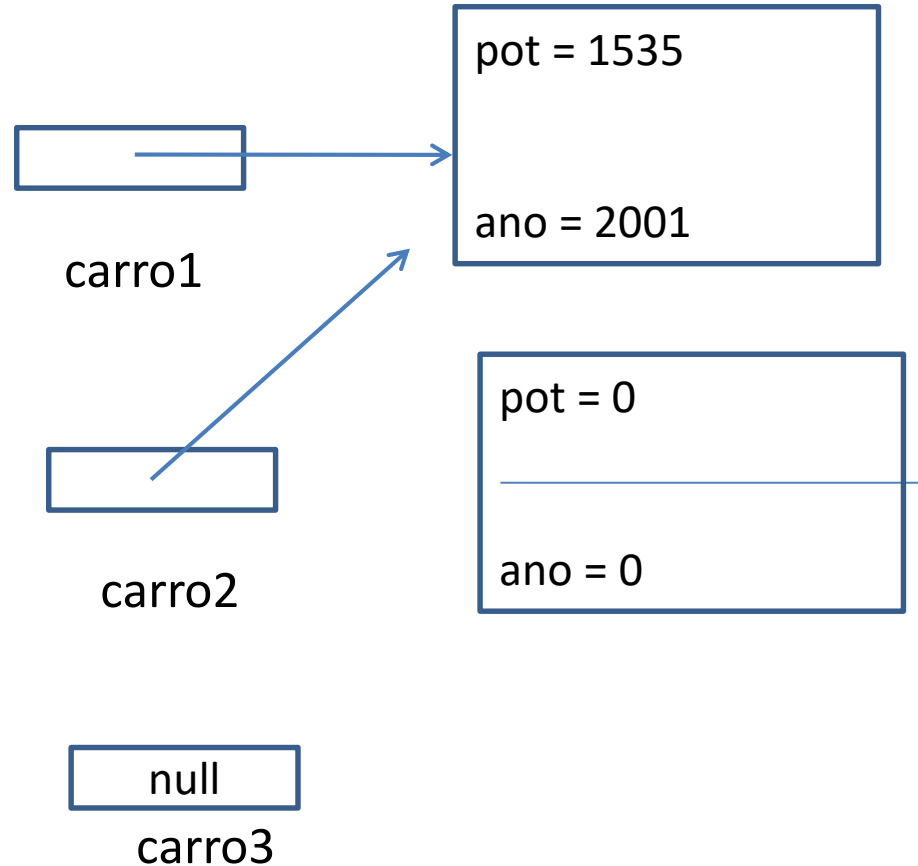
Programação Orientada a Objectos - P. Prata, P. Fazendeiro



Qual o valor de `carro1 == carro2`;

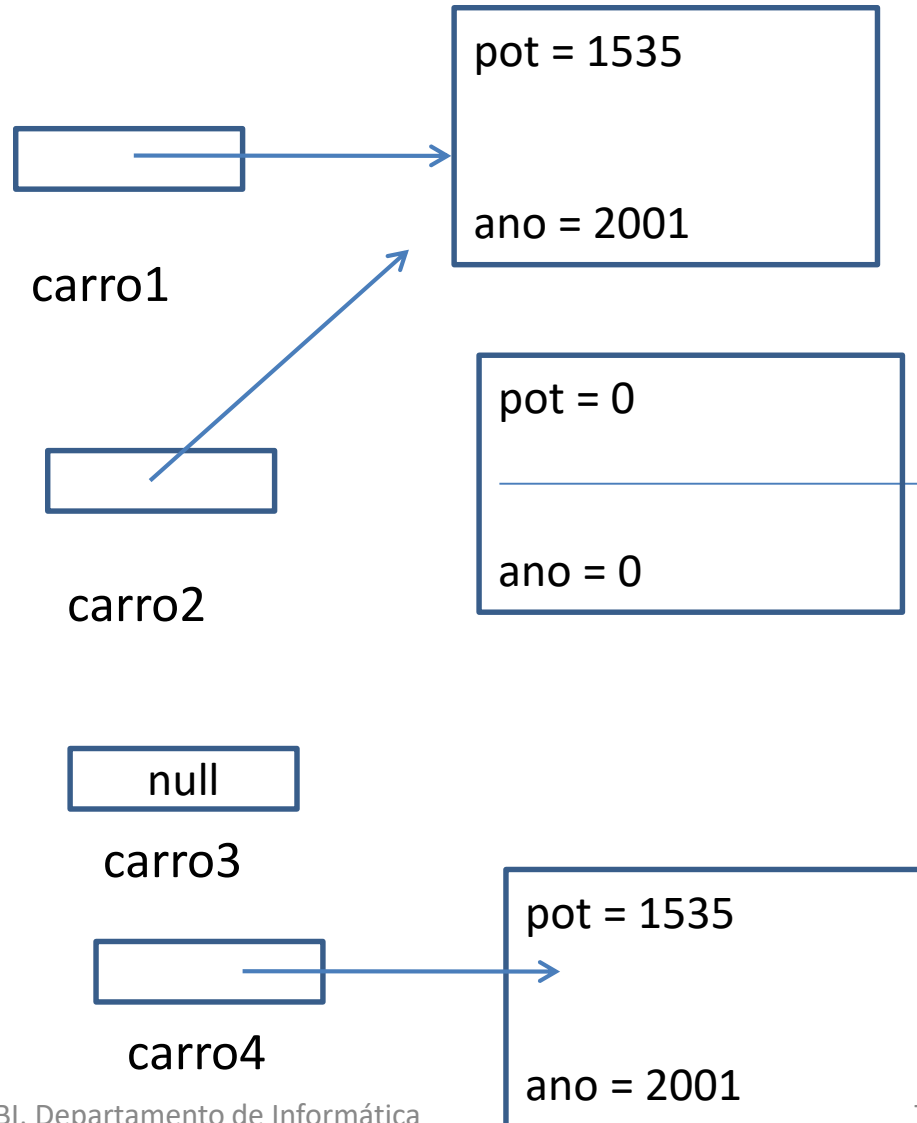
Programação Orientada a Objectos - P. Prata, P. Fazendeiro

carro1 == carro2; **True**



Veiculo carro4 = **new** Veiculo(1535, 2001); o que acontece?

Programação Orientada a Objectos - P. Prata, P. Fazendeiro



`carro1 == carro4 ??`

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

carro1 == carro4 **False**

