

Programa  o Orientada a Objetos

Aula 03 - Objetos, construtores

Apresentação

Nesta aula, você vai colocar seus conhecimentos em prática escrevendo um programa real na linguagem Java. Além disso, iremos aprender como acontece a criação de um objeto, figura central da Programação Orientada a Objetos.



Vídeo 01 - Apresentação

Objetivos

Após a leitura desta aula, você será capaz de:

- Conhecer o conceito e a criação dos métodos Construtores.
- Aprender como criar um objeto em Java.

O Ponto de Partida



Oi pessoal, vamos pôr a mão na massa?!

Iremos agora fazer nosso primeiro programa Java... uhoooooooooooo

Para que um programa Java execute ou “rode”, como costumamos dizer, é necessário que exista um método especial chamado **main(...)**, o qual veremos em detalhe na sequência. Esse método é responsável pelo início da execução do **Aplicativo ou Programa escrito** em Java.

Aplicativos

São programas que podem ser executados sozinhos. A classe serve de ponto de partida para o aplicativo. Todo aplicativo em Java precisa ter um método *main()*. O método main tem a seguinte assinatura.

```
public static void main(String arguments[]){  
    //corpo do método  
}
```

Se em um mesmo aplicativo houver mais de uma classe com o método *main()*, quando um *main* de uma das classes for executado, os demais serão ignorados enquanto o programa é executado.

Veja só, falamos de aplicativo (ou programa)! Vamos escrevê-lo. Para isso, precisamos do arquivo Carro.java da aula anterior, veja a **Listagem 1**:

```
1 class Carro {
2
3     String tipo;
4     String cor;
5     String placa;
6     int numPortas;
7
8     void setCor(String c){
9         cor = c;
10    }
11    String getCor(){
12        return cor;
13    }
14
15    String getTipo(){
16        return tipo;
17    }
18
19    void setTipo(String tipo){
20        this.tipo = tipo;
21    }
22
23    String getPlaca(){
24        return placa;
25    }
26    void setPlaca(String placa){
27        this.placa = placa;
28    }
29
30    int getNumPortas(){
31        return numPortas;
32    }
33    void setNumPortas(int numPortas){
34        this.numPortas = numPortas;
35    }
36 }
```

Listagem 1 - Classe Carro

Precisamos criar outra classe, que irá conter o método main(), para podermos executar nosso programa, como mostra a Listagem 2:

```

1  /**
2  *ClasseResponsavelPorExecutarONossoPrograma
3  *
4  * @author nome_do_autor
5  *
6  */
7  public class Main {
8      public static void main (String[] args){
9          /* aqui estará todo o código responsável por executar o programa:
10         *1- Criação de objetos
11         *2- Manipulação dos seus dados
12         *3- Finalização do programa
13     }
14 }

```

Listagem 2 – Classe Main

Vamos salvar esse arquivo como Main.java, agora temos dois arquivos Carro.java e Main.java. O método main está vazio, não faz nada, iremos editá-lo para que realize a seguinte sequência:



Vídeo 02 - Construindo Objetos

Métodos Construtores

Antes de tudo, temos que ter em mente que objetos são construídos, você **NÃO** pode criar um novo objeto sem invocar um construtor. Construtores representam o código que roda sempre que você usa a palavra-chave new.

Toda classe **DEVE** ter um construtor. Mas isso não significa que o programador tem que necessariamente codificar um. Caso não seja explicitamente declarado, o compilador criará um por padrão. Exemplo de construtor:

```

1  class Carro{
2      Carro(){ } //O Construtor para a classe Carro.
3  }

```

Você consegue notar o que está faltando? Cadê o tipo de retorno? Existem dois pontos importantes que devemos notar a respeito dos construtores. Primeiro, é que eles não têm tipo de retorno, segundo, que o nome deve ser igual ao nome da classe. Você pode estar se perguntando: se eu quiser, no momento da criação do objeto, passar valores para alguns de seus atributos, como placa e cor, por exemplo? É possível?

É sim! Esta é a função básica dos construtores!

Mas, se a classe não tiver método construtor? O objeto ainda pode ser criado usando a instrução (ou operador) new. Mas, nesse caso, será provavelmente necessário chamar métodos set, como fizemos anteriormente. Considere a seguinte definição de um método construtor na classe Carro adicionando o método construtor mostrado na Listagem 3:

```
1 class Carro {  
2  
3     String cor;  
4     String placa;  
5     Carro(String placa, String cor){  
6         this.placa = placa;  
7         this.cor = cor;  
8     }  
9  
10 }
```

Listagem 3 – Construtor para classe Carro

Observe que esse método possui o mesmo nome da classe Carro e não possui tipo de retorno, pois seu retorno, como já mencionamos, é justamente a referência para o objeto na memória, lembra-se disso?

Atenção, se fizermos apenas essa alteração, iremos nos deparar com um erro de compilação na classe Main. Você sabe por quê?

Então, como finalmente criamos uma instância da classe Carro utilizando o seu construtor? Exemplo:

```
1 Carro meuCarro = new Carro();
```

Observe que quando utilizamos o operador new não existem argumentos (ou parâmetros) dentro dos parênteses. Isso significa que podemos estar usando o construtor padrão (ou, em inglês, default).



Vídeo 03 - Construtores

Sobrecarga de Construtores

Vamos aprofundar um pouco mais a respeito dos construtores. Imagine agora se você quisesse dar mais flexibilidade para a criação dos seus objetos. Nesse caso, estamos usando a classe Carro como base, então, imagine que queremos dar a possibilidade de criar carros informando a sua cor, porém, também queremos ter a possibilidade de criar carros sem informar a sua cor, e que ao não informar a cor, que o carro seja considerado branco por padrão. Como faremos isso?

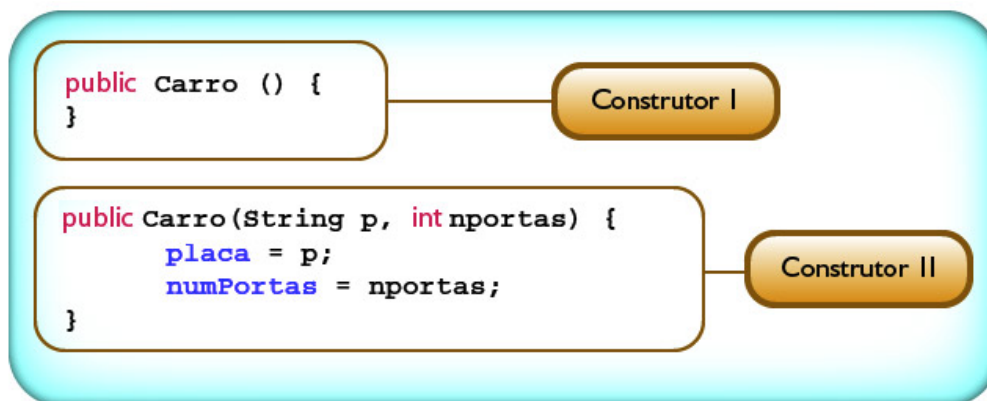
A resposta está na sobrecarga de construtores, como veremos a seguir, essa técnica é tipicamente utilizada para formas alternativas de instanciar objetos da sua classe, como pode ser visto na Listagem 4.

```
public Carro () {  
    this.cor = "branco";  
}  
  
public Carro(String cor) {  
    this.cor = cor;  
}
```

→
Iniciamos a cor do carro com o valor "branco" como padrão.

Listagem 4 – Sobrecarga de construtor na classe Carro

Quando em uma classe define-se outro construtor, não podemos mais usar o construtor padrão, a não ser que se defina explicitamente um construtor sem parâmetros, como mostra o construtor 1 na Listagem 5. Com isso, a classe Carro ficará com dois **construtores**.



Listagem 5 – Dois construtores para uma mesma classe

Observe que no primeiro construtor nada foi feito, enquanto no segundo os atributos placa e num Portas foram inicializados (receberam valores). Isso significa que existem duas maneiras de se construir um objeto derivado da classe Carro, ou do tipo Carro, com ou sem inicialização dos valores dos atributos.

Com isso, fechamos a base para se criar um objeto a partir de uma classe e ainda executar um programa que utilize esse objeto.

O que são construtores?

- Construtores são métodos especiais para a criação e inicialização de novas instâncias de classe (objetos).
- Inicializam o novo objeto e seus atributos.
- Criam todos os outros objetos de que esse objeto necessita.
- Realiza todas as outras operações que ele precisa para ser inicializado.

O que faz o operador new?

- Ele inicializa o novo objeto e seus atributos, cria todos os outros objetos de que esse objeto necessita e realiza todas as outras operações de que ele precisa para ser inicializado.

Agora que sabemos um pouco mais sobre construtores, vamos criar um objeto da classe Carro, no mundo da programação orientada a objetos, chamamos isso de instanciação, ou seja, criaremos uma instância da classe Carro.



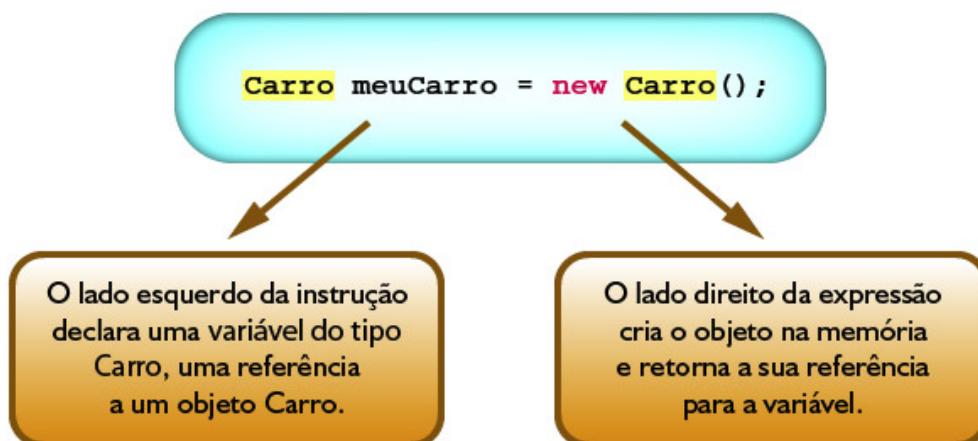
Vídeo 04 - Sobrecarga

Criando um Objeto

Instância = objeto

Um objeto ou instância é criado através do operador **new**. Vejamos a seguinte instrução na Figura 1:

Figura 01 - Esquema de criação de um Objeto



Toda a manipulação e consulta dos atributos do objeto serão feitas através de sua referência, ou seja, da variável que possui essa referência para os dados e métodos armazenados na memória.

Nesse instante, nosso carro foi criado, mas nenhum de seus atributos foi ainda definido. Vamos utilizar os métodos da classe Carro para defini-lo como quisermos, para isso, vamos utilizar o conjunto de métodos **setAtributo(valor)**. Veja a **Listagem 6**.

```
1 meuCarro.setCor("preto");
2 meuCarro.setNumPortas(4);
3 meuCarro.setPlaca("MHX 1234");
4 meuCarro.setTipo("esportivo");
```

Listagem 6 – Definição de atributos usando o método set

Agora, vamos exibir cada atributo com seu respectivo método **getAtributo()**. Como mostra a **Listagem 7** a seguir.

```
1 System.out.println("Cor: "+meuCarro.getCor());
2 System.out.println("Número de portas: "+meuCarro.getNumeroPortas());
3 System.out.println("Placa: "+meuCarro.getPlaca());
4 System.out.println("Tipo: "+meuCarro.getTipo());
```

Listagem 7 – Exibição dos valores dos atributos

Logo, a nossa classe Main ficará como mostra a Listagem 8:

```

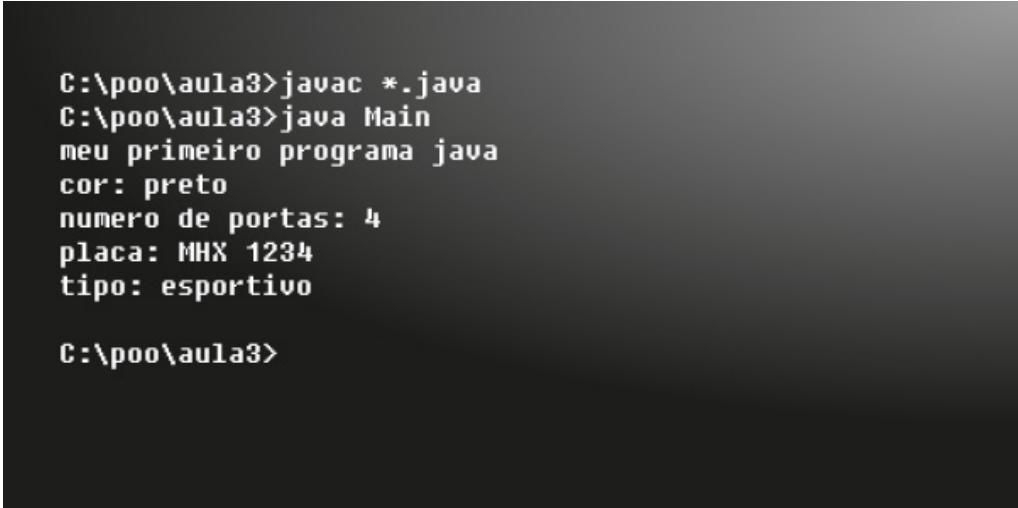
1  /**
2  *ClasseResponsavelPorExecutarONossoPrograma
3  *
4  * @author nome_do_autor
5  *
6  */
7
8  public class Main{
9
10     public static void main(String[] args){
11         /*aqui estará todo o código responsável por executar o programa:
12          *1- Criação de objetos
13          *2 - Manipulação de seus dados
14          *3- Finalização do programa
15          */
16
17         System.out.println("Meu primeiro programa java");
18
19         Carro meuCarro = new Carro();
20         meuCarro.setCor("preto");
21         meuCarro.setNumPortas(4);
22         meuCarro.setPlaca("MHX 1234");
23         meuCarro.setTipo("esportivo");
24
25         System.out.println("Cor: "+meuCarro.getCor());
26         System.out.println("Número de portas: "+meuCarro.getNumeroPortas());
27         System.out.println("Placa: "+meuCarro.getPlaca());
28         System.out.println("Tipo: "+meuCarro.getTipo());
29
30     }
31
32 }

```

Listagem 8 – Método main completo

Por fim, compile a classe que você acabou de criar e depois execute-a. Você verá a saída do programa apresentada na Figura 2.

Figura 02 - Execução do método main



```
C:\poo\aula3>javac *.java
C:\poo\aula3>java Main
meu primeiro programa java
cor: preto
numero de portas: 4
placa: MHX 1234
tipo: esportivo

C:\poo\aula3>
```

Atividade 01

1. Crie no exemplo apresentado outra variável Carro chamada outroCarro.
2. Defina valores distintos aos seus atributos através de seus respectivos métodos set e os exiba como foi feito com a variável meuCarro.

Leitura Complementar

WIKIPÉDIA. **Java:** linguagem de programação. Disponível em: http://pt.wikipedia.org/wiki/Java_%28linguagem_de_programa%C3%A7%C3%A3o%29. Acesso em: 13 maio 2010.

A Wikipédia em português apresenta uma introdução a Java, discutindo inclusive o método main.

WIKIPÉDIA. **Main function:** programming. Disponível em: http://en.wikipedia.org/wiki/Main_function_%28programming%29. Acesso em: 13 maio 2010.

Se você quiser ver como o método main é definido em várias linguagens de programação existentes, acesse o link acima.

Resumo

Com esta aula, você já é capaz de se aventurar na programação orientada a objetos em Java, pois você já sabe como escrever um código simples e transformá-lo em um código executável. Vimos que podemos alterar e acrescentar novas classes aos nossos programas e aos poucos transformá-los em um grande sistema. Você viu também a estrutura de criação de um objeto em Java, bem como a definição dos valores para os atributos dos objetos. Por fim, você estudou o conceito e a prática sobre os métodos construtores e sua utilidade para a criação dos objetos. Nossa caminhada está apenas começando, mas já demos excelentes passos!

Autoavaliação

1. Edite o arquivo Carro.java adicionando mais atributos à classe, como marca, modelo, ano de fabricação, tipo de combustível (álcool, gasolina ou flex).
2. Edite o arquivo Main.java para definir e exibir os novos atributos.
3. Crie um segundo carro na classe Main utilizando o construtor com parâmetros.

Referências

BARNES, David J.; KÖLLING, Michael. **Programação orientada a objetos com Java**. São Paulo: Pearson Prentice Hall, 2004.

DEITEL, H. M.; DEITEL, P. J. **Java como programar**. Porto Alegre: Bookman, 2003.

LEMAY, Laura. **Aprenda Java em 21 dias**. Tradução: Daniel Vieira. Rio de Janeiro: Campos, 2003.