Programação Orientada a Objetos

Aula 04 - Sistemas OO

Apresentação

Agora, você será apresentado ao verdadeiro mundo da Orientação a Objetos. Você terá a oportunidade de estudar de que maneira construímos sistemas de grande porte utilizando princípios básicos desse paradigma e mecanismos da linguagem Java.

Vídeo 01 - Apresentação

Objetivos

Nesta aula, você será capaz de:

- Entender o que é realmente um Sistema OO.
- Compreender o mecanismo de troca de mensagem entre os objetos.

Sistema = Mundo OO

Quando falamos em sistema, estamos falando em um programa complexo que implementa a solução de um problema complexo. Até agora, vimos pequenos programas, específicos e simplificados, pois trata-se de uma situação didática. Mas, iremos adicionando complexidade pouco a pouco e logo você será capaz de desenvolver suas próprias soluções.

O mundo em que vivemos está repleto de objetos (ou coisas) que nos cercam e fazem a nossa vida ser como é, pois sem eles não conseguimos sequer imaginar como seria. Tais objetos definem nosso mundo por possuírem características próprias e se relacionarem entre si.

Nele, vimos carros trafegando pelas ruas, com edifícios que possuem escadas e elevadores; utilizamos nossos objetos pessoais, como relógios, óculos e bolsas. Todos têm uma finalidade, atribuímos nomes a eles e os ligamos a pessoas e a outros objetos. É o nosso mundo! E esse mundo está sujeito a leis naturais e humanas. Essa analogia é importante para entendermos que um sistema computacional é abstrato, mas segue um princípio de construção semelhante ao mundo que nos cerca.

Um sistema é um verdadeiro mundo orientado a objetos, se a linguagem assim o definir, como é o caso da linguagem Java. Pois, nela, todo programa por mais simples que seja, até um complexo sistema de controle de tráfego aéreo, por exemplo, são feitos por um conjunto de muitos objetos de várias classes diferentes.

Atividade 01

1. Verifique agora a sua volta a quantidade de objetos que o cerca, identifique um conjunto de 5 a 10 deles e descreva as seguintes características: nome e para que servem.

   Por exemplo, aqui eu tenho um computador na minha frente, então aí vai:

Nome: Computador

Serve para: Trabalhar (inclui preparar esta aula), comunicar-me com meus amigos e me divertir.

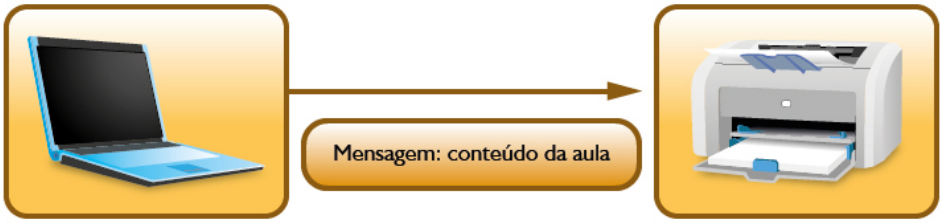
Como foi visto na atividade anterior, praticamente todos os objetos estão relacionados a outros, pois nenhum faz tudo sozinho, nem mesmo meu computador de última geração, pois sem a tomada na parede ele não é nada.

Os objetos precisam se "comunicar", ou seja, um objeto aciona um método de outro, e este, por sua vez, de um outro, e assim por diante. Essa comunicação é realizada através do mecanismo de **troca de mensagem.**

Vejamos a seguinte situação.

Estou digitando esta aula e sei que está prestes a faltar energia na área do meu escritório, mas preciso revisá-la, então, decido imprimi-la, para poder ler o documento da aula enquanto estiver faltando energia. O meu objeto **Computador** não é capaz de sozinho fazer essa tarefa, ele precisa interagir com a impressora. Como mostrado na Figura 1, o computador envia os dados (conteúdo desta aula) em forma de mensagem para a impressora, que realiza a sua tarefa de imprimir o conteúdo do documento em papel, para ser lido.

Figura 01 - Objetos do mundo real trocando mensagens



Agora, vejamos outro exemplo, este no mundo OO – Java, relembrando nosso programa:

```
1 meuCarro.setCor("preto");
2 System.out.println("placa: " +meuCarro.getPlaca());
```

Listagem 1 - Exemplo de troca de mensagem com um objeto da classe Carro

Esse código escrito dentro da classe Main envia uma mensagem para definir um atributo e outra para recuperar a placa do carro. **Em Java, a troca de mensagem representa: (i) a mudança ou leitura do estado interno do objeto através da alteração de um de seus atributos;** ou (ii) a chamada a um dos métodos do objeto que representam seu comportamento e as tarefas que são capazes de desempenhar.

Durante a execução de um sistema, que se inicia sempre pelo método main() (não se esqueça!), há várias, milhares de trocas de mensagem entre objetos! Logo, precisamos projetar bem os sistemas e gerenciar a complexidade para que não se transforme em uma tarefa árdua diária.

Atividade 02

---

1. Envie mensagens para o objeto meuCarro definindo seus outros atributos (modificando seu estado interno), e outras mensagens recuperando esses novos valores (acionando o comportamento de informar ao mundo exterior seu estado interior).

Nos sistemas orientados a objetos, quando objetos de classes distintas possuem uma troca de mensagens muito frequente, por possuírem uma autodependência forte, ou cooperarem entre si por objetivos mútuos, é comum estabelecermos um **relacionamento** entre essas classes.

O nosso exemplo do mundo real expresso através do computador e da impressora possui um relacionamento entre ambos, pois, quando solicitada a impressão de um documento, meu computador já possui um cadastro das impressoras de que disponho, caso contrário, terei que instalar uma nova. Logo, a classe **Computador** possui um relacionamento com a classe **Impressora**, se assim formos traduzir para o mundo OO.

Esses relacionamentos costumam ser definidos através de suas cardinalidades.

Mas, o que é isso?

Bem, quando relacionamos coisas no mundo real, fazemos isso de forma natural e espontânea, teremos que ser mais observadores daqui pra frente.

Vejamos, quantas pessoas cabem em um automóvel? Na maioria deles, a lotação é de no máximo 5 pessoas, incluindo seu condutor. Mas, uma pessoa pode estar em quantos automóveis ao mesmo tempo? Até onde as leis da física permitem, só em um, de cada vez!

Chamamos esse relacionamento de *um-para-muitos*, um (automóvel) para muitos (pessoas). Em outras situações, vemos que um marido é para uma, e somente uma, esposa, e a recíproca é verdadeira, temos então um relacionamento *um-para-um*. A outra combinação possível seria *muitos-para-muitos*, o que verificamos no nosso exemplo computador e impressora, pois do meu computador posso imprimir em várias impressoras e cada uma dessas podem receber documentos dos outros computadores do escritório.

Vídeo 02 - Relacionamento entre Objetos

Atividade 03

1. Complemente a atividade anterior adicionando relacionamentos aos objetos listados por você. Tome como exemplo:

   Computador

   Relacionado com: é pessoal, é meu. E está conectado a um mouse e a rede elétrica do meu escritório.

Continuando Nosso Programa...

Vamos ligar a classe Carro à classe Pessoa, fazendo com que uma pessoa possua um carro e esse carro só possa pertencer a uma única pessoa. É o que chamados de relação *um-para-um*.

Camila possui um carro vermelho

Figura 02 - Exemplo de relacionamento entre objetos

A classe Carro precisa “saber que pertence” a alguém. Iremos adicionar um atributo chamado dono, que é do tipo Pessoa. Exibindo a lista de atributos de Carro após a mudança:

```
1 class Carro {
2    String tipo;
3    String cor;
4    Strin placa;
5    int numPortas;
6    Pessoa dono;
7    ...
8 }
```

Listagem 2 - Codificação do relacionamento entre as classes Carro e Pessoa

E os respectivos métodos get e set:

```
1 class Carro {
2 Pessoa getDono(){
3 return dono;
4 }
5 void setDono(Pessoa dono){
6 this.dono = dono;
7 }
8 }
```

Listagem 3 - Métodos que relacionam Carro a Pessoa

Vamos utilizar esse relacionamento no exemplo seguinte:

```
1  public class Main{
2
3     public static void main(String[] args){
4
5        Carro meuCarro = new Carro();
6
7        meuCarro.setCor("preto");
8        meuCarro.setNumPortas(4);
9        meuCarro.setPlaca("MHX 1234);
10       meuCarro.setTipo("esportivo");
11
12       Pessoa pessoa = new Pessoa();
13       pessoa.setNome("Camila");
14       pessoa.setIdade(27);
15
16       System.out.println("Cor: "+meuCarro.getCor());
17       System.out.println("Número de portas: "+meuCarro.getNumeroPortas());
18       System.out.println("Placa: "+meuCarro.getPlaca());
19       System.out.println("Tipo: "+meuCarro.getTipo());
20       System.out.println("Pertence a: "+meuCarro.getDono().getNome());
21
22    }
23 }
```

Listagem 4 - Programa exemplo do relacionamento Carro e Pessoa

Observe a linha:

```
1  System.out.println("Pertence a: "+meuCarro.getDono().getNome());
```

Listagem 5 - Comando com exemplo de troca de mensagem

Neste ponto, a classe Main envia uma mensagem para o objeto carro, recuperando o seu dono, um objeto da classe Pessoa, e esse recebe uma mensagem para que retorne o seu nome.

A **troca de mensagens** é comumente denominada de **chamada de método**.

Vídeo 03 - Implementando Relacionamentos

Até agora, você estudou a troca de mensagens entre objetos para definir ou recuperar algum atributo, unicamente. Os sistemas OO precisam de métodos mais complexos, que verifiquem a validade dos dados, realizem atualizações em outras entidades, pesquisem em banco de dados, escrevam em arquivos de log, atualizem objetos da interface etc.

Vamos colocar um pouco da complexidade do mundo real no nosso programa?! Vamos lá!

Faremos as seguintes alterações:

1. Adicionar os métodos ligar(), desligar(), acelerar(), frear() à classe Carro.

```
1  class Carro{
2      void ligar(){
3          System.out.println("carro ligado.");
4      }
5      void desligar(){
6          System.out.println("carro desligado.");
7      }
8      void acelerar(){
9          System.out.println("carro acelerando");
10     }
11     void frear(){
12         System.out.println("carro freando.");
13     }
14 }
```

Listagem 6 - Novos métodos para a classe Carro

2. Vamos adicionar à classe Carro, ainda, um atributo câmbio que indicará qual a “marcha” que o carro está em um dado momento, para tanto, tal atributo irá guardar um valor inteiro de 0 (zero) a 5 (cinco).

    0 - Neutro (ponto morto)

    1 a 5 - marchas

```
class Carro{
   int cambio;
   void setCambio(int marcha){
      this.cambio = marcha;
   }
   int getCambio(){
      return this.cambio;
   }
}
```

Listagem 7 - Novo atributo e métodos da classe Carro

3. Precisamos colocar uma referência do automóvel dentro da classe Pessoa.

```
class Pessoa{
   Carro carro;
   Carro getCarro(){
      return carro;
   }
   int setCarro(){
      this.carro = carro;
   }
}
```

Listagem 8 - Métodos que implementam o relacionamento entre Carro e Pessoa

4. Vamos adicionar outros métodos à classe Pessoa para interagir (trocar mensagens) com seu carro.

```
class Pessoa{
   void ligarCarro(){
      carro.ligar();
   }
   void desligarCarro(){
      carro.desligar();
   }
   void acelerarCarro(){
      carro.acelerar();
   }
   void frearCarro(){
      carro.frear();
   }
   void setCambioCarro(int marcha){
      carro.setCambio(marcha);
   }
}
```

Listagem 9 - Classe Pessoa recebe métodos para interagir com a classe Carro

5. Vamos ao Main! Nosso programa precisa:

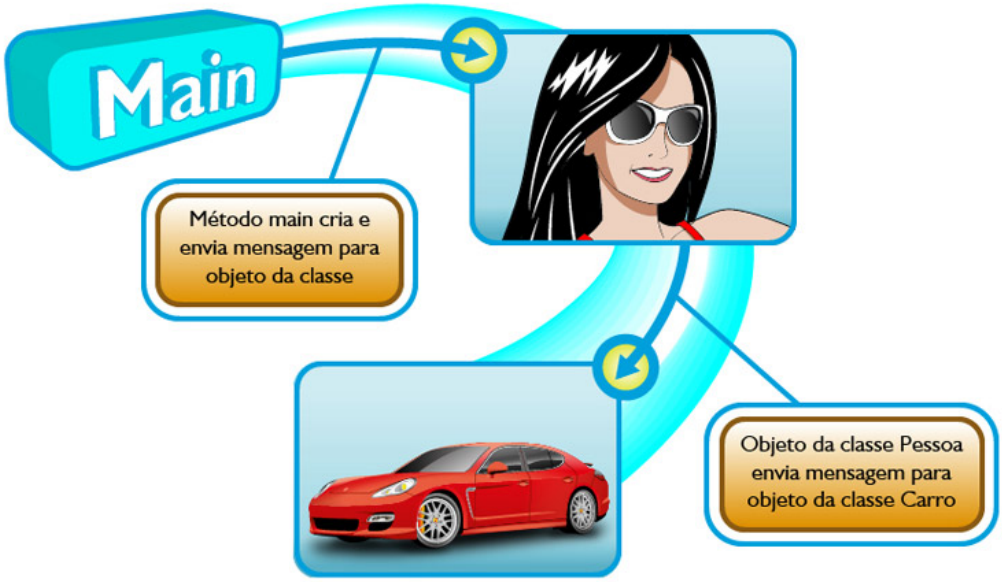
1. Criar uma Pessoa;
2. Criar um Carro e atribuí-lo a essa Pessoa;
3. Fazer com que a pessoa ligue o carro, troque de marcha, acelere, reduza a velocidade, troque de marcha e desligue o carro.

```
1  public class Main{
2     public static void main(String[] args){
3
4     Carro meuCarro = new Carro();
5
6     meuCarro.setCor("preto");
7     meuCarro.setNumPortas(4);
8     meuCarro.setPlaca("MHX 1234);
9     meuCarro.setTipo("esportivo");
10
11    Pessoa pessoa = new Pessoa();
12    pessoa.setNome("Camila");
13    pessoa.setIdade(27);
14
15    pessoa.setCarro(carro);
16
17    pessoa.ligarCarro();
18    pessoa.setCambioCarro(1);
19    pessoa.acelerarCarro();
20    pessoa.setCambioCarro(2);
21    pessoa.acelerarCarro();
22    pessoa.setCambioCarro(3);
23    pessoa.acelerarCarro();
24    pessoa.setCambioCarro(2);
25    pessoa.acelerarCarro();
26    pessoa.setCambioCarro(1);
27    pessoa.acelerarCarro();
28    pessoa.setCambioCarro(0);
29    pessoa.frearCarro();
30    pessoa.desligarCarro();
31    }
32 }
```

Listagem 10 - Programa exemplo para testar o relacionamento entre Carro e Pessoa

O que podemos ver é que o método Main envia uma mensagem para o objeto pessoa, que, por sua vez, manda ou repassa essa mensagem para seu objeto da classe Carro, que executa a ação final desejada, que consiste em guiá-lo. Veja a ilustração a seguir:

Figura 03 - Envio de mensagens entre objetos



Vídeo 04 - Adicionando outros Métodos

Leitura Complementar

No site http://www.leandro.wives.nom.br/java/oojava.htm, você encontrará informações sobre o que vimos nesta aula e um breve resumo do que ainda veremos. Aconselho que leia apenas o que se sentir confortável e possa assimilar no momento, pois veremos mais técnicas de programação orientada a objetos na sequência das aulas.

Resumo

Nesta aula, você ampliou os horizontes. É importante que neste ponto você revise o que foi visto até agora, pois os elementos fundamentais da orientação a objetos e os mecanismos básicos da linguagem Java foram apresentados e combinados para gerar uma aplicação simples, mas bastante significativa. Os sistemas são arranjos complexos de objetos que se relacionam trocando mensagens entre si e cooperando para atingir objetivos planejados pelo seu programador. É necessário um bom planejamento para manter a complexidade do sistema sob controle.

Autoavaliação

1. O que é a **troca de mensagens** entre objetos?
2. Como os sistemas OO implementam sua solução?
3. Os exemplos apresentados, identifique os trechos onde há troca de mensagens envolvendo objetos das classes Carro e Pessoa.

Referências

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