Programação Orientada a Objetos

Aula 06 - Composição ou Agregação

Apresentação

Compor, juntar, reunir, formar, constituir... Essas ações nos levam a uma ideia muito interessante na programação orientada a objetos, chamada *Composição*. Nesta aula, você vai ver esse interessante conceito da POO, que nos dá a grande possibilidade da reutilização de código. Além disso, a ideia de composição também aproxima a ideia da Orientação a Objetos ao senso comum de sistematização das coisas, em que um objeto mais complexo pode ser composto de partes mais simples. Boa aula!!

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Saber o que significa Composição ou Agregação;
- Criar objetos na linguagem Java que são compostos por outros objetos, fazendo uso prático do conceito de Composição;
- Entender como se comporta os modificadores de acesso de Java na Composição.

Composição ou Agregação

No Universo da Programação Orientada a Objetos: eu sou uma Composição ou Agregação!!!

> Composição ou Agregação é um mecanismo de **reaproveitamento (reutilização) de classes** utilizado pela POO para aumentar a **produtividade** e a **qualidade** no desenvolvimento de software.

Reaproveitamento ou reutilização de classes significa que você pode usar uma ou várias classes para compor outra classe. Já o aumento de produtividade está relacionado com a possibilidade de não ser necessário reescrever código de determinadas classes, se alguma outra já existe com estado (atributos) e comportamento similar. Finalmente, com a composição, é possível também aumentar a qualidade dos sistemas gerados, porque há a possibilidade clara de reutilizar classes que já foram usadas em outros sistemas, e, portanto, já foram testadas e têm chances de conter menos erros.

Vejamos um exemplo: um carro é um objeto **COMPOSTO** por vários outros objetos. Ele é composto pelos objetos motor, pneus, direção, faróis etc. A Figura 1 dá uma ideia do conceito de composição para um objeto do tipo Carro.

Figura 01 - Composição do objeto Automóvel

Legal não é?!! Isso mesmo que você entendeu, você pode criar um objeto a partir de vários outros objetos. E isso é muito natural.

Veja o computador que você está usando, ele é um objeto composto por outros objetos: teclado, monitor, placa mãe, memória, mouse etc.

E tem mais, quando uma classe é composta de outras classes, ela pode tanto usar os objetos que são gerados pelas classes que a compõem, como pode também usufruir dos atributos e métodos dessas classes.

Vídeo 02 - Composição

Atividade 01

1. Para praticar o conceito de composição que você acabou de aprender, tente olhar o mundo ao seu redor, outros exemplos de objetos do mundo real que são compostos por outros objetos menores. Anote tais exemplos para que possamos usá-los em outros exercícios ao longo da nossa aula.

Exemplos na Prática

Vejamos agora um exemplo na prática! Vamos utilizar a mesma ideia do objeto Automóvel, dado uma classe Automóvel que é composta pelas classes Direção e Motor.

Figura 02 - Composição da classe Automóvel: motor e direção

Nas linhas de código, essa Composição é expressa da seguinte maneira, veja Listagem 1:

```
class Motor{
    private int potencia;
}
```

```
class Direcao{
    private String cor;
}
```

```
class Automovel{
    private Motor motor;
    private Direcao direcao;
}
```

Listagem 1 - Classes Motor, Direção e Automóvel

Observe que a classe **Automóvel** é composta pelas classes **Motor** e **Direção**, eis aqui a nossa Composição! Mas, lembre-se de que primeiro criamos as classes **Motor** e **Direção** para só a partir daí podermos criar a classe **Automóvel**.

Atividade 02

1. Crie, baseado no exemplo apresentado, outras classes compostas de várias outras classes. Lembre-se: para que uma classe possa compor uma outra, é necessário que ela já exista. Elabore também classes para os exemplos que você mesmo criou na Atividade 01. As classes são:

Classe Composta	Classes Componentes (“Partes”) da Classe Composta
Computador	Teclado, Monitor, Memória, Placa Mãe...
Livro	Título, Autor, Capítulo, Editora ...
Monstro	Cabeça, Olho, Boca, Braço, Perna...

Mas, o conceito e vantagens do uso da Composição não acabam por aí, analisando as classes, vamos incrementá-las um pouco mais e descobrir os benefícios de se utilizar a Composição.

Observe que a classe Automovel possui atributos que são referências para instâncias (Ver aula 03 (Objetos e Construtores).) (objetos) das classes Motor e Direção. Observe que as classes possuem a propriedade do *Encapsulamento* (visto na aula 05, **Encapsulamento**), consequentemente, para acessar os atributos das classes usaremos os métodos **get** e **set** (vistos na aula 2, **Classes, atributos e métodos**). Assim, nossas classes ficam conforme indicam as listagens a seguir: Listagem 2, Listagem 3 e Listagem 4:

```
class Motor{
   private int potencia;
   public int getPotencia(){
      return this.potencia;
   }
   public void setPotencia(int potencia){
      this.potencia = potencia;
   }
}
```

Listagem 2 - Classe Motor

```
class Direcao{
   private String cor;
   public int getCor(){
      return this.cor;
   }
   public void setCor(String cor){
      this.cor = cor;
   }
}
```

Listagem 3 - Classe Direção

```
class Automovel{
   private Motor motor;
   private Direcao direcao;

   public Motor getMotor(){
      return this.motor;
   }
   public void setMotor(Motor motor){
      this.motor = motor;
   }
   public Direcao getDirecao(){
      return this.direcao;
   }
   public void setDirecao(Direcao direcao){
      this.direcao = direcao;
   }
}
```

Listagem 4 - Classe Automóvel

Da maneira que está o código das classes, se nós criarmos um objeto (instanciarmos) da classe **Automóvel**, os **atributos** motor e **direção** continuam com valor nulo (null).

```
Automovel automovel = new Automovel();}
```

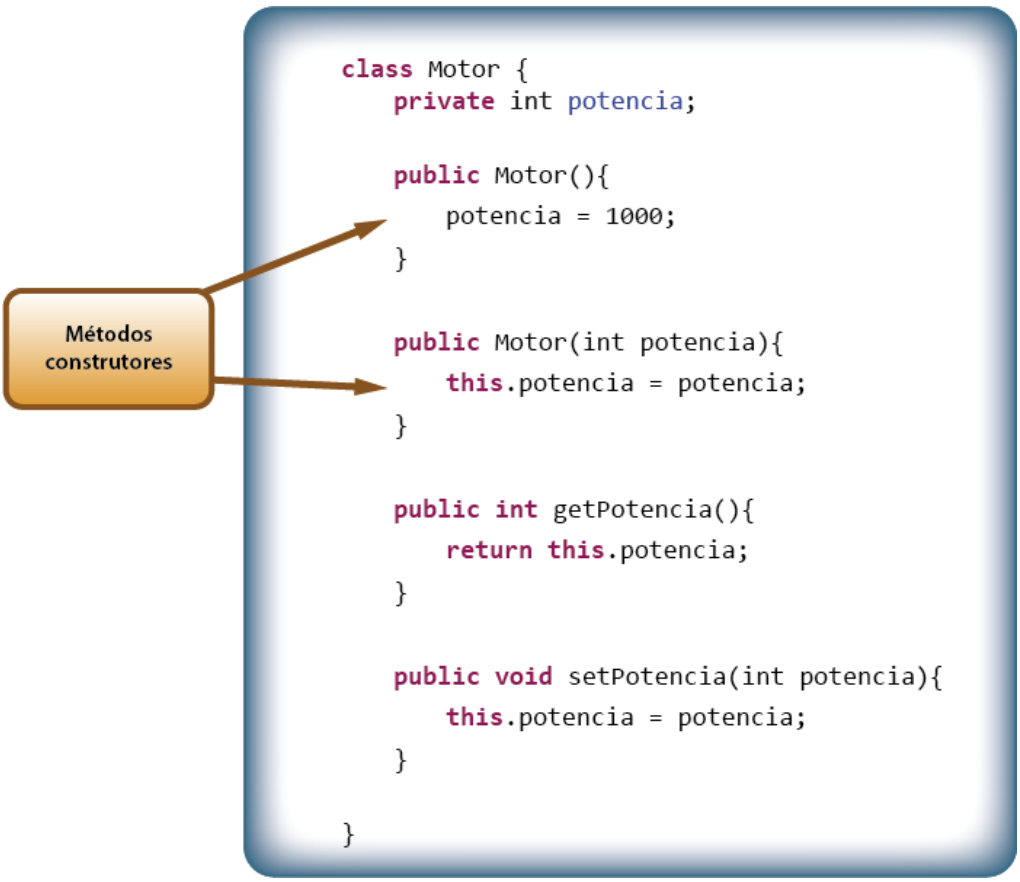
Isso significa que para instanciarmos os objetos que compõem a classe **Automóvel** podemos decidir de que maneira faremos.

A Composição e o Método Construtor

Os métodos construtores das classes componentes (Motor,Direção) que fazem parte da classe composta (Automóvel) podem ser chamados de três maneiras diferentes, são elas:

- **CASO 1:** chamadas nos construtores da classe que é composta;
- **CASO 2:** chamadas em qualquer método da classe que é composta;
- **CASO 3:** chamadas fora da classe que é composta.

Para mostrar cada uma dessas situações, vamos considerar inicialmente que a classe **Automóvel** é composta apenas pela classe **Motor**. Acrescentamos dois métodos construtores para a classe **Motor**. O primeiro utiliza a potência do motor 1000 (por *default*), já o segundo espera que o usuário forneça a informação da potência do motor quando criado o objeto **Motor**, veja a Listagem 5.



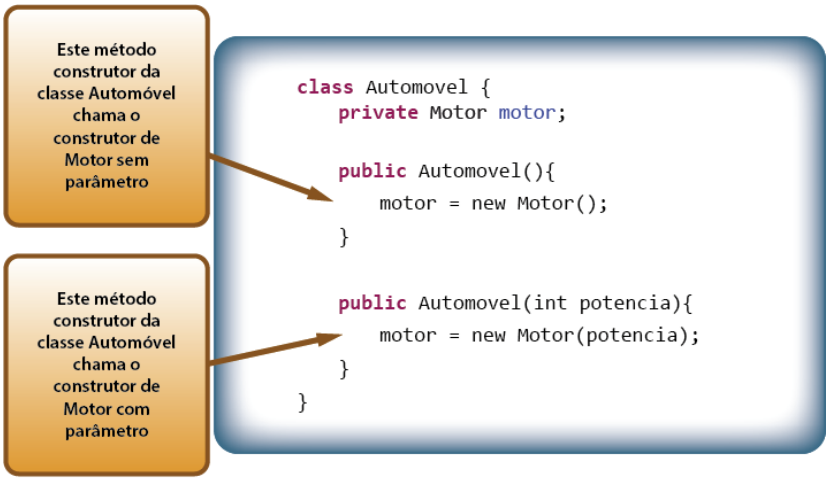
Listagem 5 - Classe Motor com dois construtores

```
1  class Automovel{
2     Private Motor motor;
3
4  public Motor getMotor(){
5     return this.motor;
6  }
7
8  public void setMotor(Motor motor){
9        this.motor=motor;
10 }
11 }
```

Listagem 6 - Classe Automóvel composta pela classe Motor

- **CASO 1:** Chamadas nos construtores da classe que é composta

Para a primeira situação, quando o construtor da classe componente é chamado no construtor da classe que usa a composição, ilustrada na Listagem 7, temos:



Listagem 7 - Classe Automóvel com dois métodos construtores

Nesse exemplo, os construtores da classe Motor são chamados dentro dos construtores da classe Automóvel.

- **CASO 2:** chamadas em qualquer método da classe que é composta

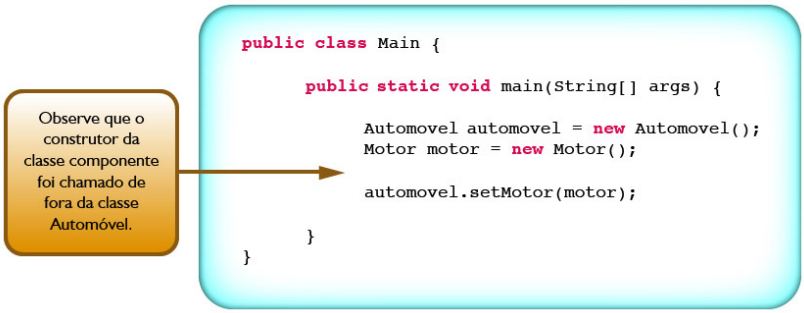
Outra maneira seria a situação, na qual os construtores são chamados em qualquer método da classe que é composta. Vejamos a Listagem 8:

```
1  class Automovel{
2     private Motor motor;
3
4     public ligarPrimeiraVez(){
5           motor = new Motor;
6     }
7
8     public ligarPrimeiraVez(int potencia){
9        motor = new Motor(potencia);
10    }
11 }
```

Listagem 8 - Construtor chamado a partir de um método

- **CASO 3:** chamadas fora da classe que é composta

Por fim, a terceira situação, onde o construtor da classe componente é chamado de fora da classe que usa a composição, veja a Listagem 9. Considere que a classe **Automóvel** utilizada possui o código apresentado na Listagem 6.



Listagem 9 - Criação da classe Motor fora da classe Automóvel

Observe que o construtor da classe Motor foi chamado de fora da classe Automóvel, dentro do método main() da classe. Ou seja, antes de usar o método setMotor() o objeto motor não tem nenhum vínculo com o objeto automóvel.

Vídeo 03 - Formas de Implementar a Composição

Comparando as Possibilidades

Observando a Listagem 10, analisaremos as possíveis maneiras de criar o objeto da classe componente (Motor) através da classe composta (Automóvel) ou dentro do método main() que define o comportamento de execução de um programa Java.

```
1  public class Main{
2     public static void main(String[] args){
3        //CASO 1
4        Automovel automovel = new Automovel();
5        Automovel automovel = new Automovel(1600);
6
7        //CASO 2
8        Automovel automovel = new Automovel();
9        automovel.ligarPrimeiraVez();
10          //outra maneira
11       automovel.ligarPrimeiraVez(1600);
12
13       //CASO 3
14       Automovel automovel = new Automovel();
15       Motor motor = new Motor();
16          //outra maneira
17       Motor motor = new Motor (1600);
18       automovel.setMotor(motor);
19    }
20 }
```

Listagem 10 - Possibilidades de criação da classe Motor para uso pela classe Automóvel

Para o **caso 1**: temos duas possibilidades, na linha **1**, é ocultado do usuário a existência de um objeto motor que compõe a classe Automóvel. Já na linha **2**, o usuário define a potência a ser adotada pelo motor do automóvel.

Para o **caso 2**, podemos observar na linha **3** que quando criamos o objeto automóvel o objeto motor ainda não existe. Ele permanece nulo (null) dentro do objeto motor até que o automóvel seja ligado pela primeira vez. Para isso, pode-se usar um dos dois métodos ligarPrimeiraVez()(linhas **4** e **5**), sem ou com parâmetro, respectivamente.

Finalmente, para o **caso 3**, apresentado a partir da linha **6**, tem-se a mesma observação da linha 3. Nesse caso, o usuário cria normalmente o objeto motor, escolhendo um dos dois construtores (linhas **7** e **8**) e, em seguida, define que esse objeto componha o objeto automóvel (linha **9**.)

Atividade 03

1. Seguindo o exemplo apresentado para as classes Automóvel e Motor, aplique o caso 1 para a classe Computador que agrega uma placa mãe, o caso 2 para a classe Livro que agrega seu título e autor, e o caso 3 para a classe Monstro que agrega cabeça e boca.

Composição e os Modificadores de Acesso

Duas observações importantes que podemos considerar com relação à composição e aos modificadores de acesso (Veja sobre modificadores de acesso na Aula 05, que trata de Encapsulamento.) são mencionadas a seguir.

- Quando declaramos **atributos públicos** nas classes e reutilizamos essas classes dentro de outras, esses atributos **podem não ser acessados facilmente, através da classe de composição**, veja Listagem 11.

```
class Direcao{
   public String cor;
}
```

```
class Automovel{
   private Direcao direcao;
}
```

```
public class Main{
   public static void main(String[] args){
      Automovel automovel = new Automovel();
      automovel.direcao.cor ="preta"; // ERRO!
   }
}
```

Listagem 11 - Tentativa de uso de um atributo público de um objeto privado

No exemplo acima, apesar do atributo **cor** da classe **Direção** ser **public**, esse atributo não é diretamente acessível a partir do método main(), porque o objeto **Direção** está encapsulado (é um atributo privado) dentro da classe **Automóvel**.

- Quando temos **atributos privados** nas classes e reusamos essas classes, declarando suas instâncias como públicas, os atributos **não passam a ser públicos**, veja a Listagem 12.

```
class Direcao{
   private String cor;
}
```

```
class Automovel{
   public Direcao direcao;
}
```

```
public class Main{
   public static void main(String[] args){
      Automovel automovel = new Automovel();
      automovel.direcao.cor ="preta"; // ERRO!
   }
}
```

Listagem 12 - Tentativa de uso de um atributo privado de um objeto público.

Aqui ocorre o inverso do caso anterior, mas reforça que private não deixa de ser privado mesmo quando na classe que usa da Composição (**Automóvel**) torna seu atributo (**Direção**) public.

Anote a Dica!

Quando uma classe usa a Composição para agregar outras classes, podemos dizer que ela tem um relacionamento chamado **"Tem um"**, o qual descreve um relacionamento em que uma classe contém uma instância de outra classe.

Vídeo 04 - Composição e Modificadores de Acesso

Leitura Complementar

Como dissemos, na POO, Agregação é um conceito bastante similar à Composição, veja um pouco mais sobre agregação no link a seguir:

<http://www.dca.fee.unicamp.br/cursos/POOCPP/node14.html>

Na literatura, também encontramos autores que fazem diferença entre Agregação e Composição. Eles usam o mesmo raciocínio usado para modelagem de Banco de Dados. É também uma ideia válida e interessante. Veja no link a seguir:

<http://pt.wikipedia.org/wiki/Classe_%28programa%C3%A7%C3%A3o%29>

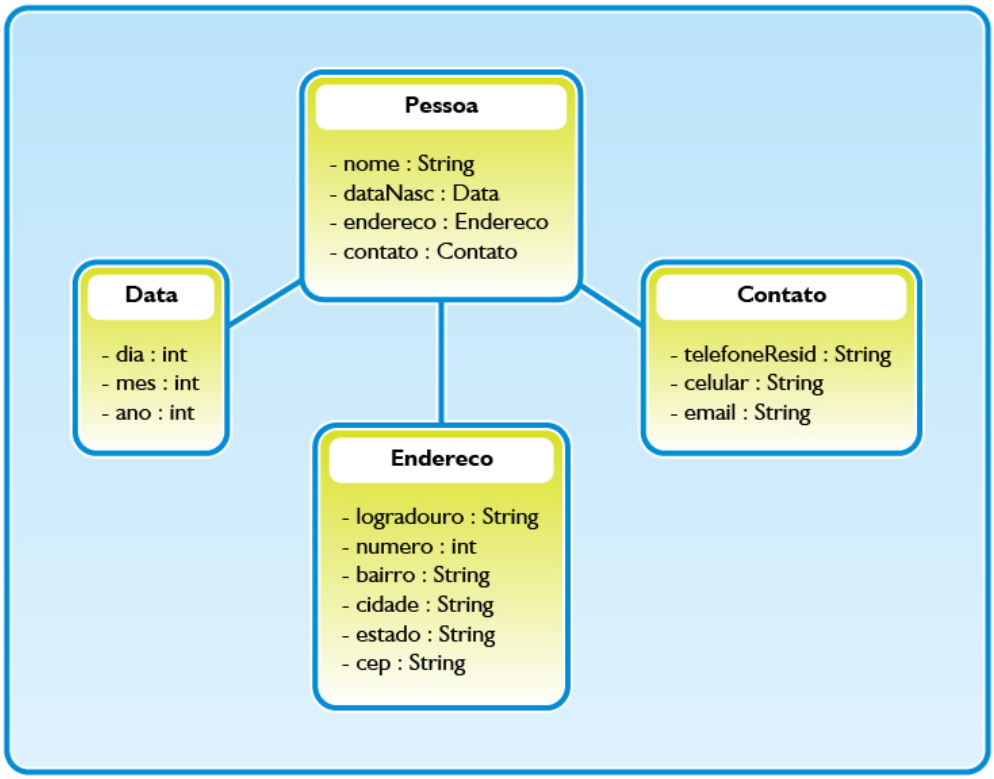
Resumo

Nesta aula, você estudou o que é **Composição**. Viu que composição é a capacidade de fazer com que uma classe seja composta de vários objetos de outras classes. A Composição oferece a possibilidade de fazer uso do comportamento das classes que ela agrega (como atributos) de maneira implícita ou explícita, dependendo da maneira que se escolhe para instanciar as classes componentes.

Autoavaliação

1. Sem consultar o material, responda: o que você entendeu por Composição?
2. Quais são as vantagens de se usar a Composição?
3. Dos três casos de instanciação dos objetos componentes, qual você usaria se quisesse omitir a presença da composição para quem irá criar os objetos das classes que usa a composição?

4. Crie as classes apresentadas no diagrama abaixo e aplique a Composição para a classe Pessoa, que além de possuir um atributo **Nome** será composta pelas classes **Data**, **Endereço** e **Contato** para os atributos dataNasc, endereço e contato, respectivamente.



Referências

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

SANTOS, Rafael. **Introdução à programação orientada a objetos usando Java.** Rio de Janeiro: Editora Campus, 2003.