Análise e Desenvolvimento de Sistemas

Programação Orientada a Objetos

Prof. Ivan Marcelo Pagnoncelli

Aula 4

Conversa inicial

Olá! Seja bem-vindo(a) à quarta aula da disciplina **Programação Orientada a Objetos!**
Nesta aula, faremos uma introdução aos paradigmas da orientação a objetos, que serão explorados mais a fundo nas aulas subsequentes. Sendo assim, estes são os tópicos da aula:

- Paradigmas da Orientação a Objetos
- Encapsulamento
- Associação, Agregação e Composição
- Encapsulamento na Linguagem Java
- Tipos de Associação na Linguagem Java

Você encontra a introdução do professor Ivan sobre o assunto no material *on-line*! Acesse!

Contextualizando

A programação orientada a objetos tem algumas características próprias, os chamados paradigmas, que nos permitem desenvolver sistemas de grande porte e complexos com menos esforço e maior assertividade. Esses paradigmas foram sendo enunciados e aperfeiçoados ao longo do desenvolvimento da POO. Atualmente, são eles:

1. Encapsulamento
2. Associação, Agregação e Composição
3. Herança
4. Polimorfismo

O professor Ivan fala mais um pouco sobre a introdução aos paradigmas da POO no material *on-line*!

Os Paradigmas da Orientação a Objetos

Antes de iniciarmos nossa viagem pelos paradigmas da programação orientada a objetos, vamos elucidar o que vem a ser um paradigma, segundo o filósofo Thomas Kuhn:

“Paradigmas são modelos, representações e interpretações de mundo universalmente reconhecidas que fornecem problemas e soluções modelares para uma comunidade científica. ”

Conheça as características dos paradigmas a seguir:

Encapsulamento: neste paradigma, utilizamos os conhecimentos que adquirimos no encontro sobre modificadores de visibilidade. Como o nome sugere, os atributos devem ser escondidos dos outros objetos presentes no sistema, sendo encontrados apenas através de métodos de acesso (*getters* e *setters*) setados (definidos) como públicos.

Associação, Agregação e Composição: a associação representa uma relação entre dois objetos sem pertencimento, por exemplo: um aluno está associado a um professor e um professor está associado a vários alunos. Outras duas associações possíveis são:

- Agregação, quando um objeto agrega outros objetos para criar um todo;
- Composição, onde um objeto é composto por outros, mas, ao contrário da associação, sem pertencimento.

Herança: como o nome sugere, esse paradigma sugere uma relação de hereditariedade entre duas classes: temos a superclasse, que é herdada, e a subclasse, que recebe os atributos e métodos da superclasse como herança.

Polimorfismo: ao pé da letra, polimorfismo significa “várias formas”, ou seja, esse paradigma sugere que um mesmo

método pode assumir várias “formas”, ou implementações na mesma classe ou em classes diferentes, sendo denominado polimorfismo estático ou dinâmico.

O professor Ivan nos conta mais sobre os paradigmas da POO no material *on-line*! Não deixe de acessar!

O Encapsulamento

O primeiro paradigma que iremos ver se chama **Encapsulamento.** Como vimos em aulas passadas, as classes têm um mecanismo chamado **modificadores de visibilidade** que permitem que escolhamos apenas o que deve ser visualizado por outros objetos que fazem parte do sistema, protegendo, assim os atributos da classe e, consequentemente, o estado do objeto, de alterações não permitidas, que possam deixar nosso objeto inconsistente.

A teoria por trás da ideia de encapsular um objeto é de que os atributos sejam protegidos e externalizados apenas a partir de métodos de acesso. O encapsulamento também é importante para o projeto de sistemas orientados a objetos, pois permite que definamos bem qual é a *interface da classe*, o conjunto de métodos públicos da classe.

Externalizar **o que** nossa classe faz, e não **como** ela faz, é essencial na POO, pois garante que o utilizador da classe não precise saber de detalhes sobre sua implementação. É como ligar a TV. Sabemos que existe um botão que, ao clicarmos, fará uma imagem aparecer. Não nos interessa como essa imagem foi gerada na TV, e essa funcionalidade também não está disponível na interface da classe.

Vejamos um exemplo de uma classe que implementa o Encapsulamento. Neste exemplo, criamos uma classe chamada

Pessoa que tem seus atributos privados, ou seja, não visíveis para o mundo externo, sendo que esses atributos só podem ser acessados pelos métodos de acesso, os *getters* e *setters.*

```
public class Pessoa {
    private String nome;
    private int tipo;
    private String endereco;
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public int getTipo() {
        return tipo;
    }
    public void setTipo(int tipo) {
        this.tipo = tipo;
    }
    public String getEndereco() {
        return endereco;
    }
    public void setEndereco(String endereco) {
        this.encereco = endereco;
```

```
    }
}
```

Ainda tem dúvidas a respeito desse tópico? Acesse o material *on-line* e confira o vídeo do professor Ivan, ele está aqui para ajudar!

Associação, Agregação e Composição

A associação entre classes é um paradigma bastante utilizado na POO. A teoria por trás dele é que uma associação representa um relacionamento entre objetos.

Por exemplo, vários alunos podem estar associados à um único professor e um único aluno pode estar associado à vários professores. Nesse caso, não existe um relacionamento de posse entre esses objetos, todos eles são independentes. Um aluno pode existir sem a necessidade de um professor e o contrário também é verdadeiro.

Veja um exemplo na relação a seguir:

```
public class Pneu {
    private int raio;
    private double pressao;
    // getter e setters
}
public class Carro {
    private Pneu[] pneus;
    // getters e setters
}
```

Nas classes declaradas, onde o código relacionado aos *getters* e *setters* foi omitido, uma classe Carro é uma associação de Pneus, ou, vários Pneus estão associados a um Carro.

Para identificarmos uma associação em um código, o normal é que a classe associada seja um atributo da outra classe.

Outros dois tipos de associação possíveis são a Agregação e a Composição.

Na agregação, temos um todo que é composto de partes, sendo que as partes existem sem o todo, mas o todo não existe sem as partes. Por exemplo, um conselho é um agregado de membros. O conselho só existe devido aos membros, mas os membros podem existir sem o conselho.

No exemplo a seguir, uma venda é um agregado de comprador, vendedor e produtos. A venda só existirá se existir um comprador, um vendedor e produtos, mas essas classes existirão sem uma venda.

```
public class Vendedor {
    private double comissao;
    // getters e setters
}
public class Comprador {
    private double budget;
    // getters e setters
}
public class Produto {
    private double preco;
```

```
    // getters e settes
}
public class Venda {
    private Vendedor vendedor;
    private Comprador comprador;
    private Produto[] produtos;
    // getter e setters
}
```

Outra forma de associação é a Composição, em que o objeto que contém e o objeto contido só existem em conjunto. Um exemplo de composição é um banco que contém contas corrente e contas poupança. Se o objeto ‘banco’ deixa de existir, os objetos ‘conta corrente’ e ‘conta poupança’ também deixam de existir.

Outro exemplo pode ser encontrado a seguir, no qual uma nota fiscal é composta por determinados itens. Se a nota fiscal não existir, os itens também deixam de existir.

```
public class ItemNotaFiscal {
    private int tipo;
    private double valor;
    // getters e setters
}
public class NotaFiscal {
    private int numero;
    private ItemNotaFiscal[] itens;
```

```
    // getters e setters
}
```

Para fixar seus conhecimentos sobre relações de associação, agregação e composição, recomendamos o seguinte artigo:

http://douglascrp.blogspot.com.br/2008/06/associao-agregao-e-composio.html

E o professor Ivan fala mais um pouco sobre o assunto no material *on-line*! Não esqueça de conferir!

Trocando ideias

Baseando-se nas ideias apresentadas hoje, faça uma análise sobre as possíveis relações em que você está envolvido no seu dia a dia e enumere quantas e quais são as associações, agregações e composições que você consegue identificar! Se quiser, troque ideias com amigos e colegas de trabalho.

DICA: Pesquise na internet e em sites de programação mais exemplos e informações sobre os paradigmas! Isso pode te ajudar a identificar mais relações do seu cotidiano.

Na prática

Baseando-se nas ideias apresentadas e após estudar os exemplos do professor, tente criar as classes e a relação descritas na especificação a seguir:

Crie um sistema de software que mantenha um trem como um conjunto de locomotivas e vagões, no qual você tenha a possibilidade de adicionar e excluir locomotivas.

Qual é o tipo de associação que teremos?

No material *on-line* você encontra o vídeo do professor Ivan, no qual ele demonstra exemplos criados no programa Eclipse! Acesse e pratique um pouco mais, tirando eventuais dúvidas!

Síntese

Chegamos ao final dessa aula!

Vimos aqui que a programação orientada a objetos nos traz paradigmas que nos permitem otimizar nosso desenvolvimento. Também começamos a ver mais a fundo esses paradigmas, começando pelo paradigma do encapsulamento, seguido pelos tipos de associação.

Daqui em diante, tente aplicar os paradigmas no seu cotidiano como programador ao utilizar orientação a objetos. Logo você descobrirá a agilidade e flexibilidade que eles trazem para o trabalho!

Até a próxima!

E para finalizar, acompanhe a síntese em vídeo do professor Ivan no material *on-line*!

Referências

Jandl Junior, P. **Introdução ao Java.** Berkeley Brasil, 2002.