# Análise e Desenvolvimento de Sistemas

# Programação Orientada a Objeto

Aula 3

Introdução a Programação Orientada a Objetos

Prof. Ivan Marcelo Pagnoncelli

## Conversa inicial

Uma das diferenças entre **programação estruturada** e **programação orientada a objetos**, conforme vimos, é que os objetos têm como "esconder" os atributos e métodos que não podem ser acessados por outros objetos que fazem parte do sistema. Vamos ver como.

Esta aula está abordará os seguintes tópicos:

- Visibilidade de Classes, Métodos e Atributos
- Visibilidade aplicada no Java
- A Visibilidade na prática

**Vamos começar?**

**Saiba mais sobre os conteúdos desta aula, assistindo ao vídeo de introdução que o professor Ivan preparou para você! Acesse o material *on-line*!**

## Contextualizando

A possibilidade de os objetos que criamos terem a capacidade de não deixar acessíveis alguns atributos e métodos é muito importante e interessante, já que essa capacidade faz com que tenhamos sistemas mais seguros e sem erros, pois as ações que serão executadas serão apenas as que permitirmos.

**Entenda melhor essas ideias, assistindo ao vídeo que está disponível no material *on-line*.**

## Tema 1: A Visibilidade aplicada a Classes, Métodos e Atributos

**Visibilidade** (ou acessibilidade) é a capacidade que uma classe tem de alterar quem pode, ou não, ter acesso aos seus atributos e métodos.

Esta possibilidade vem da teoria por trás do paradigma da **orientação a objetos**. Como estamos lidando com objetos que abstraem o mundo real, nada mais real que as classes serem responsáveis por quem pode ter acesso aos seus atributos e métodos. A validação que deve ser feita quando um novo valor vai ser atribuído a um atributo da classe deve ser responsabilidade dela mesma.

Por vezes, um atributo tem sentido apenas para a classe que o declarou, não sendo necessário que outras classes saibam de sua existência.

Um exemplo de por que utilizamos os modificadores de acesso pode ser visto na classe a seguir:

```
public class ContaCorrente {
            private double saldo;
            private double limite;

            public double saque(double valor) {
                  if(valor >= saldo + limite) {
                        saldo -= valor;
                  }
                  return saldo;
            }
            public double deposito(double valor) {
                  saldo += deposito;
                  return saldo;
            }
            public double getSaldo() {
                  return saldo;
            }
      }
```

Como podemos ver, o atributo **saldo** está declarado como privado, não permitindo seu acesso direto por outras classes. Dessa forma, o acesso não é permitido dessa maneira: conta.saldo.

Fazemos isso para que outras classes não possam fazer o acesso direto ao atributo saldo, evitando que o mesmo seja alterado sem o conhecimento e a permissão da classe. Ou seja, apenas utilizando as ações que a classe

disponibiliza, de saque ou depósito, é que podemos alterar o valor do saldo, que pode ser consultado por um método *getter*.

O outro atributo que a classe tem, mas não mostra, é o atributo **limite**, que só é utilizado, em nossa classe, quando fazemos um saque. Como esse atributo não é utilizado, a não ser por nossa classe, não há motivo para ele seja externalizado.

Atributos também podem ser públicos, mas não é normal. O normal é termos os atributos privados e os métodos de acesso a eles como públicos.

Modificar o acesso a alguns itens de uma classe oferece algumas vantagens, como por exemplo:

- ✓ O usuário da classe (aquele que instancia a classe para utilizar seu objeto) passa a conhecer apenas aquilo que é necessário para a utilização do objeto criado.
- ✓ Detalhes sobre a forma de implementação são ocultos dos usuários da classe, permitindo resguardar esforços de desenvolvimento ou tecnologias proprietárias.
- ✓ Pode-se limitar os valores assumidos por alguns atributos a um determinado conjunto de valores possíveis, assegurando a integridade e a consistência dos dados armazenados na classe.
- ✓ Passa a existir uma garantia de que determinadas operações sejam realizadas em uma certa sequência ou tenham seu funcionamento condicionado a estados da classe.

Os modificadores de acesso, então, podem ser:

- **Públicos**

  Quando expõem os atributos ou métodos que eles modificam para outros objetos, sendo possível sua visualização e alteração.
- **Privados**

Quando não permitem que os métodos e atributos por eles modificados sejam visíveis e alteráveis pelos outros objetos que compõe o sistema.

- **Protegidos**

  Que funciona de forma semelhante ao modificador privado, mas permite que os atributos e métodos sejam visíveis às subclasses.

Todos os métodos de uma classe podem ter seu acesso modificado, inclusive os construtores. Quando declaramos um construtor de uma classe como privado, essa classe não pode ser construída por outro objeto. Assim a classe consegue também ter acesso à criação de objetos de si própria. Essa técnica é chamada de *singleton* e constitui um *design pattern* bastante útil e utilizado.

A utilização de modificadores de visibilidade ou de acesso permite a utilização de um dos paradigmas mais importantes da POO, que é o **Encapsulamento**. Por meio dele, as características do objeto são responsabilidade total do objeto.

Veremos mais sobre encapsulamento futuramente.

**Tire suas dúvidas sobre o assunto, assistindo ao vídeo que está disponível no material *on-line*.**

## Tema 2: A Visibilidade no Java

A linguagem Java permite que utilizemos os modificadores de acesso, conforme a POO preconiza. No Java, as palavras chave para os modificadores de acesso são:

- ✓ *Public*, para o modificador de acesso público.
- ✓ *Private*, para o modificador de acesso privado.
- ✓ *Protected*, para o modificador de acesso protegido.

Quando, no Java, não especificamos que modificador de acesso será aplicado a determinada classe, atributo ou método, sua visibilidade será pública, mas apenas dentro do pacote no qual a classe se encontra.

Esse modificador é chamado de "*friendly*" ou então "*package*".

**Para se aprofundar mais nessas questões, assista ao vídeo no material *on-line* com as explicações do professor Ivan.**

## Trocando ideias

Pesquise sobre o *design pattern* chamado *Singleton* e sua aplicação prática.

Depois, procure responder à questão:

**Como resolvemos o fato do construtor da classe ser privado?**

Discuta a questão no fórum desta disciplina, disponível no Ambiente Virtual de Aprendizagem.

## Na prática

Vamos colocar em prática o que aprendemos nesta aula, executando no Eclipse a criação de uma classe com modificadores de acesso setados e a utilização de seu objeto.

Vamos criar a classe a seguir:

```
package laboratório;
public class Cachorro {
	private String nome;
	private String raca;
	private double peso;

	public String getNome() {
		return nome;
	}
```

```
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getRaca() {
		return raca;
	}
	public void setRaca(String raca) {
		this.raca = raca;
	}


	public double getPeso() {
		return peso;
	}
	public void setPeso(double peso) {
		if(peso > 0) {
			this.peso = peso;
		} else {
			this.peso = 0.0;
		}
	}
```

Agora, vamos criar esta outra classe e executá-la:

```
	package laboratório;
	public class Principal {
		public static void main(String[] args) {
			Cachorro cao = new Cachorro();

			cao.setNome(“Toto”); // Ok
			cao.nome = “Toto”; // ERRO DE ACESSO AO
ATRIBUTO NOME
		}
	}
```

Como o Eclipse irá nos mostrar, a segunda classe não irá compilar, pois temos um erro no acesso ao atributo nome, que está sendo acessado

corretamente, via método *setter* e posteriormente está sendo acessado diretamente, o que não é possível devido ao fato desse atributo ser privado.

A classe cachorro também nos mostra que, ao fazermos acesso ao atributo **peso** via método de acesso, estamos garantindo que não tenhamos um objeto do tipo Cachorro como peso menor que zero, garantindo assim a consistência de negócio da classe.

Confira, no material *on-line*, o vídeo que o professor Ivan preparou para você, no qual ele demonstrará os exemplos sendo construídos em tempo real no programa Eclipse, propício para criações em Java!

## Síntese

**Chegamos ao final desta aula!**

Nesta aula, vimos que atributos e métodos podem ser “escondidos” de outras classes através dos modificadores de acesso.

Fazemos isso para que os valores dos atributos das classes, ou seja, o estado do objeto, fique protegido de alterações indevidas.

**Até a próxima!**

Assista às considerações finais do professor Ivan no vídeo que está disponível no material *on-line*.

## Referências

JANDL JUNIOR, P. **Introdução ao Java**. Berkeley Brasil, 2002.