

Orientação a Objetos



Prof. Dr. João Paulo Lemos Escola
Copyright© 2022

Tópicos da aula

- Objetos;
- Classes;
- Escopo;
- Encapsulamento;
- Construtor;
- Classe como atributo.

Introdução

- O paradigma da Orientação a Objetos (OO) é amplamente utilizado no mercado de trabalho;
- Possibilita melhor organização do código, tornando-o mais prático para manutenção;
- Conceitos que fazem parte da OO:
 - Classes;
 - Objetos;
 - Métodos;
 - Atributos;
 - Modificadores de acesso etc.

Objeto

- Abstração de objetos (coisas) do mundo real;
- Todo objeto possui características físicas, formas, atributos;
- Exemplo:
 - Duas casas são dois objetos diferentes;
 - Uma é azul e a outra é branca. (atributos);
 - Uma tem 3 banheiros e a outra tem apenas um. (atributos).



Classe

- Objetos são gerados a partir das classes;
- Instanciar um objeto significa gerar um objeto de uma classe especificada;
- Classes são modelos, moldes que resultarão em objetos. Devemos declarar nas classes as características que queremos ter nos objetos;
- Apesar vários objetos poderem ser gerados a partir de uma classe, cada objeto é único, com espaço único reservado na memória, valores específicos em seus atributos.

Estrutura de uma classe

```
qualificador class nome-da-classe{
```

```
    // declaração dos atributos
```

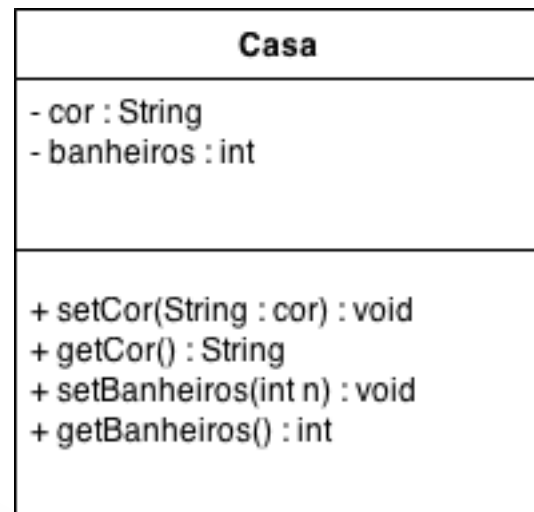
```
    // declaração dos métodos construtores (opcional)
```

```
    // declaração dos métodos
```

```
}
```

- Qualificador indica sua visibilidade e pode ser: **public** (pode ser acessado de qualquer classe), **private** (pode ser acessado somente pela classe atual), **protected** (pode ser acessado somente pela classe atual e pelas classes filhas).

Classe UML



Código da classe Casa

```
public class Casa{  
    private String cor;  
    private int banheiros;  
    public String getCor(){  
        return cor;  
    }  
    public int getBanheiros(){  
        return banheiros;  
    }  
    public void setCor(String cor){  
        this.cor = cor;  
    }  
    public void setBanheiros(int n){  
        this.banheiros = n;  
    }  
}
```

Atributos ou campos

Métodos de acesso

Padrão de declaração de métodos

- Para criar métodos de acesso, por padrão, utilizamos o prefixo get (pegar) e set (especificar);
- Todo método que começa com “set” vai receber um valor (parâmetro) e armazená-lo no atributo:

```
setNome("Maria");
```

```
setCor("Azul");
```

```
setSaldo(100);
```

- Todo método que começa com “get” vai retornar um valor de um ou mais atributos:

```
getNome(); // retorna o nome (valor do atributo da classe)
```

```
getCor(); // retorna a cor
```

```
getSaldo(); // retorna o saldo
```

this

- A palavra reservada “this” faz referência ao objeto corrente;
- No exemplo abaixo, a palavra this.banheiro referece ao atributo “banheiros” da classe e a outra variável “banheiros” refere-se ao parâmetro que foi utilizado ao chamar o método:

```
public void setBanheiros(int banheiros){  
    this.banheiros = banheiros;  
}
```

Criando objetos da classe

- Criar objetos da classe significa instanciar objetos da classe:

```
Casa casa1; // declaração de objeto
```

```
casa1 = new Casa(); // instancição do objeto
```

```
Casa casaDaMaria = new Casa();
```

```
Casa casaVermelha = new Casa();
```

Uso de objetos

- Após instanciar os objetos, podemos utilizá-los por meio de seus métodos:

```
casa1.setCor("branca");
```

```
casa1.setBanheiros(1);
```

```
System.out.println(casa1.getCor());
```

```
System.out.println(casa1.getBanheiros());
```

Escopo dos atributos e métodos

- Por padrão, os atributos devem ser privados e os métodos públicos;
- Assim, garantimos que os atributos serão acessados externamente somente por meio dos métodos;
- Se alterarmos o modificadores dos atributos, poderemos mudar seus valores diretamente, o que não é recomendável.

Código da classe ContaBancaria

```
public class ContaBancaria{  
    public int numero;  
    public double saldo;  
    public void setNumero(int numero){  
        this.numero=Agencia.buscarCliente(numero); // confirma se o cliente existe  
    }  
    public void efetuarSaque(double valor){  
        double tarifa = valor * 0.01; // calcula desconto de 1% como tarifa  
        Agencia.addTarifa(tarifa); // atribui a tarifa à agência  
        this.saldo = this.saldo - (valor+tarifa); // desconta do usuário  
    }  
    public int getSaldo(){  
        return saldo;  
    }  
}
```

O correto seria utilizar o modificador **private**

Uso da classe ContaBancaria

- Errado:

```
ContaBancaria conta = new ContaBancaria();  
conta.numero = 112222;  
conta.saldo = 0;  
System.out.println("Saldo: "+conta.getSaldo());
```

- Correto:

```
ContaBancaria conta = new ContaBancaria();  
conta.setNumero(111222);  
conta.efetuarSaque(10);  
System.out.println("Saldo: "+conta.getSaldo());
```

Encapsulamento

- Quando temos código definido em um método de forma a executar as atividades de que o nome do método se refere, estamos fazendo o encapsulamento;
- Assim, não importa, no futuro, o que é executado dentro do método. O importante é o que seu nome indica como tarefa que será executada:
 setNome("Maria");
 conectarBanco("meuBD");
 listarClientes("2015-01-13");
 enviarEmail("jpescola@gmail.com", "Assunto: Aviso", msg);
- Como vimos, os procedimentos bancários que devem ser executados quando se efetua um saque, estão codificados no método **efetuarSaque**;
- Então, basta utilizar o método, que todas as tarefas necessárias serão realizadas.

A7ex1.jar

- Crie a classe Computador com os atributos:
 - processador, memoria, marca, monitor
- Instancie os objetos pcMaria e pcJoaquim;
- Especifique os atributos de cada objeto;
- Mostre os atributos de cada objeto;

A7ex2.jar

- Crie a classe Carro com os atributos:
 - cor, marchas, cavalos, portas, marca e aro
- Instancie os objetos fusca e camaro;
- Especifique os atributos de cada objeto;
- Mostre os atributos de cada objeto;

Construtores

- Os métodos construtores permitem instanciar objetos com valores inicializados:

```
// cor e banheiros indefinidos
```

```
Casa casa1 = new Casa(); // cor e banheiros indefinidos
```

```
// cor definida na instânciação do objeto
```


```
Casa casa1 = new Casa("Amarela");
```

```
// cor e banheiros definidos na instânciação do objeto
```

```
Casa casa1 = new Casa("Amarela", 3);
```

Código da classe Casa com construtor

```
public class Casa{  
    private String cor;  
    private int banheiros;  
    public Casa(){  
    }  
    public Casa(String cor, int banheiros){  
        this.cor = cor;  
        this.banheiros = banheiros;  
    }  
    public String getCor(){  
        return cor;  
    }  
    (...)  
}
```



Sobrecarga de
métodos construtores

A7ex3.jar

- A partir do código do exercício 1, crie um construtor vazio e um construtor completo;
- Instancie o objeto casa1 usando o construtor vazio e o objeto casa2 usando o construtor completo;
- Mostre os valores dos atributos dos dois objetos;

Classe como atributo

- Podemos utilizar classes como atributos em uma nova classe;
- EX: na classe Casa, o atributo 'cor' é do tipo String, ou seja, 'cor' é um objeto da classe String dentro da classe Casa;
- EX2: na classe Casa podemos adicionar o atributo Cidade, que é uma classe que contém atributos 'nome' e 'estado'.

A7ex4.jar

- Crie as classes Estado (nome, sigla), Cidade (nome, Estado) e instancie os objetos sp, mg, rj e rs. Instancie também os objetos barretos e frutal. Mostre, a partir do objeto barretos, o seu nome e a sigla do seu estado. Os construtores vazios e completos devem ser adicionados nas classes.

O que aprendemos?

- Objetos;
- Classes;
- Escopo;
- Encapsulamento;
- Construtor;
- Classe como atributo.

Na próxima aula...

- Pacotes;
- Herança;
- Polimorfismo;
- Classe abstrata;
- Interfaces.