# Análise e Desenvolvimento de Sistemas

## Programação Orientada a Objetos

**Prof. Ivan Marcelo Pagnoncelli**

**Aula 2**

## Conversa inicial

Olá! Seja bem-vindo(a) à segunda aula da disciplina **Programação Orientada a Objetos!**
Como vimos anteriormente, a POO tem dois elementos principais, a classe e o objeto. Nesta aula veremos esses elementos mais a fundo, utilizaremos a linguagem Java para construir classes e objetos e verificaremos como chamar métodos e atributos através dos objetos. Confira os temas que serão tratados:

**A classe na POO**
**O objeto na POO**
**A classe na linguagem Java**
**Os tipos de classe no Java**
**Utilização de classes e objetos na prática**

Acesse o material *on-line* e confira a introdução em vídeo do professor Ivan!

## Contextualizando

Vimos que a programação orientada a objetos foi criada com o propósito de fazer com que a construção de sistemas de informação seja mais próxima das relações que temos em nosso dia a dia. As linguagens de programação mais utilizadas atualmente aderiram ao paradigma da orientação a objetos e fornecem suporte para utilizarmos a programação orientada a objetos.

Mas a programação orientada a objetos também pode oferecer riscos devido a desvantagens, que, quando ignoradas, podem levar à construção de um sistema muito grande e complexo,

fazendo com que os custos de execução e manutenção fiquem mais altos do que o previsto. Para evitar maiores problemas, é preciso ter verdadeiro domínio da linguagem. Por isso, vamos explorar um pouco mais os alicerces da programação orientada a objetos: a classe e o objeto.

Antes de entrarmos de vez no estudo do conteúdo, vamos ouvir o professor Ivan! Para isso, acesse o material *on-line*!

## A classe na POO

Observe a figura a seguir, que mostra o projeto de uma casa com todas as características que o produto final construído a partir dele deve ter. Se construirmos várias casas a partir deste projeto, teremos casas semelhantes, mas não idênticas, pois elas estarão, no mínimo, em lugares diferentes.

Embora o paradigma da orientação a objetos tenha “objeto” em seu nome, o conceito fundamental deste paradigma é a classe, visto que ela é um modelo para a criação dos objetos que vão fazer parte do sistema.

Uma classe pode ser definida como um modelo a ser seguido para criação e agrupamento de objetos semelhantes, uma

abstração para os objetos do dia a dia que farão parte do sistema. Para a POO, utilizamos a classe para modelar o sistema e depois na sua construção.

**Modelar** o sistema significa criar modelos adequados para os objetos que serão representados pelo sistema. Como no exemplo da casa, que tem na planta baixa características como quantidade de quartos, tamanho da garagem, etc.

Na construção do sistema se dará a criação dos objetos propriamente ditos, baseados nas classes modeladas. Estes objetos terão novas questões associadas a eles, como inter-relacionamentos e utilização em outros sistemas.

As classes também podem ser consideradas abstratas, visto que podem apenas conter métodos que não contém implementação, apenas a declaração.

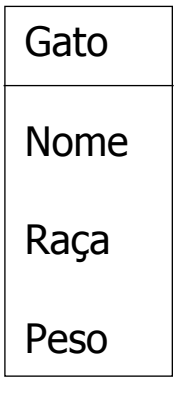
**Estrutura das Classes**

Uma classe, como visto na planta baixa anterior, tem determinadas características, também chamadas de atributos, e também pode definir ações (métodos) que os objetos podem ter.

**Atributos**

Os atributos, também chamados variáveis-membro ou variáveis de classe, são criados para manter as características que o objeto terá quando construído.

Por exemplo, podemos definir que uma classe chamada Gato terá os seguintes atributos:



A classe Gato representada em UML. Lembrando que a classe irá definir, através de modificadores de visibilidade, quem poderá visualizar esses atributos. Veremos essa habilidade das classes nos encontros seguintes.

**Métodos**

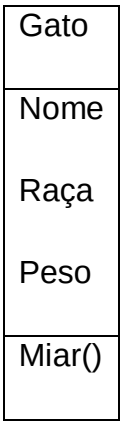
Enquanto os atributos servem para armazenar **dados** relacionados aos objetos, os métodos realizam as **ações** associadas aos objetos. Essas ações normalmente serão realizadas sobre os atributos.

Os objetos também podem ser utilizados pensando na ideia de que eles poderão enviar e receber mensagens de outros objetos. A ideia central contida na mensagem que pode ser enviada ao objeto é a mesma de quando ligamos o rádio.

Por exemplo: acionar o botão que liga esse aparelho corresponde a enviar a seguinte mensagem ao rádio: “ligue”. Ao fazermos isso, não precisamos entender como o rádio funciona, mas sim que existe a possiblidade de ligá-lo.

Voltando aos gatos, podemos criar um método chamado “miar” em nossa classe, que será utilizado por outras classes para enviar uma mensagem a nossa classe.

A classe Gato representada em UML, agora com o método miar.



As classes podem conter dois tipos especiais de métodos, o **Construtor** e o **Destrutor**.

O método Construtor será chamado quando criarmos um objeto a partir da classe, para inicialização dos atributos. Toda classe tem um método construtor que, mesmo não criado explicitamente, será criado implicitamente pela linguagem de programação. Isso se faz necessário porque o método construtor é o primeiro método da classe que é chamado.

Para finalizar esse tema sem restar dúvidas, é fundamental ouvir as explicações do professor Ivan! Acesse o material *on-line* e confira!

## O objeto na POO

Como vimos anteriormente, uma classe é um modelo utilizado para criação de elementos chamados objetos. Então, na programação orientada a objetos, podemos dizer que um objeto é uma instanciação de uma classe, ou seja, é a representação "física" do modelo que a classe descreve, como na figura:

Uma instância do projeto Casa

Como podemos ver, a figura representa uma das implementações possíveis do projeto apresentado na figura do começo da aula. Dizemos então que a casa é uma **instância** do projeto de casa.

Como dito anteriormente, todos os objetos criados a partir de uma classe são diferentes entre si, pois os objetos criados serão únicos devido aos valores atribuídos aos atributos dos objetos.

O conjunto desses valores é denominado **estado do objeto**, que pode ser alterado a qualquer momento através dos métodos que a classe disponibiliza.

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

## A classe na linguagem Java

Como dissemos anteriormente, a linguagem Java é uma linguagem que implementa o paradigma de orientação a objetos. Portanto, as características de POO mostradas até agora são todas aplicadas na linguagem Java.

Para criarmos uma classe na linguagem Java, a sintaxe é a seguinte:

*<modificador de visibilidade>* ***class*** *[Nome da classe] {*

*// atributos*

*// métodos*

*}*

*<modificador de visibilidade> (iremos ver esse tópico específico mais para frente)*

**Boas Práticas**

Existem boas práticas de programação que devem ser levadas em consideração quando desenvolvemos em Java. As classes, no Java, são definidas em arquivos *".java"* e só pode haver uma classe pública dentro de um destes arquivos. Por exemplo, para

utilizarmos no Java, a classe a seguir deve estar dentro de um arquivo chamado AloMundo.java:

```
public class AloMundo {
        public static void main(String[] args) {
                System.out.println("Alo mundo");
        }
}
```

Para a execução de uma classe no Java, devemos criar um método dentro da classe como a assinatura a seguir. Não devemos confundir este método com os métodos construtores da classe, que veremos mais adiante.

*public static void main(String args[])*

O Java possui itens chamados *pacotes* para organizar o código. É uma boa prática de programação utilizar pacotes, ou seja, não criar novas classes, mas deixá-las em pacotes e iniciar o nome dos pacotes com letra minúscula. Para definir que uma classe pertence a determinado pacote, utilizamos:

*package laboratorio;*

Esta instrução indica ao Java que a classe que será definida na sequência pertence ao pacote "laboratorio".

Atente para alguns detalhes que não são regras de linguagem, mas boas práticas de programação que devem ser observadas:

1. No Java, os nomes das classes sempre começam com letra maiúscula e, se forem compostos, os outros nomes também iniciarão com letra maiúscula, como pudemos notar anteriormente.
2. Os atributos das classes, na linguagem Java, são escritos em letra minúscula, sem "-" ou "_".

Os métodos devem iniciar sempre com letras minúsculas e as palavras subsequentes, caso existam, deverão ter a primeira letra maiúscula. Por exemplo, este método tem duas palavras em sua formação: *public void meuMetodo();*

3. Além disso, também podemos dizer que é uma boa prática manter a estrutura apresentada, ou seja, declarar primeiro os atributos e depois os métodos.

No material *on-line,* o professor Ivan fala mais um pouco a respeito das classes e boas práticas de programação! Não deixe de acessar!

## Tipos de classe na linguagem Java

No Java, além das classes reais, podemos também ter **classes abstratas** e definir **interfaces**. Vamos falar sobre elas neste tema!

As interfaces, na linguagem Java, têm algumas particularidades:

1. Não possuem implementação, apenas assinatura, ou seja, apenas a definição dos seus métodos, sem o corpo;
2. Todos os métodos são abstratos, ou seja, têm apenas declaração, nunca código;
3. Seus métodos são implicitamente Públicos e Abstratos;
4. Não há como fazer uma instância de uma Interface e nem como criar um Construtor;
5. Funcionam como um tipo de "contrato", onde são especificados os atributos, métodos e funções que as classes que implementem essa interface são obrigadas a implementar.

O Java não permite herança múltipla. Uma das maneiras de burlar isto é a implementação de várias interfaces, o que é

permitido. Não podemos criar instâncias de interfaces propriamente ditas, apenas das classes que as implementam.

Para implementar uma interface, usamos a palavra reservada *implements*, como no exemplo a seguir:

***class** [Nome da Classe] **implements** [Nome da Interface] {*

**Classes Abstratas**

As classes abstratas, na linguagem Java, seguem a regra de ter pelo menos um método abstrato definido na classe, método que não contém código, apenas declarações, como em uma interface.

A maior diferença entre uma classe abstrata e uma interface é que a classe abstrata é uma classe, que contém métodos normais, com códigos e métodos abstratos, ou seja, apenas declarativos.

Na interface temos apenas métodos declarados, sem código. As classes abstratas também não podem ser instanciadas, apenas herdadas. Sua subclasse pode ser instanciada normalmente, desde que não seja uma classe abstrata.

Segue um exemplo de uma classe abstrata em Java. Vemos que existe um método com código na classe e um método apenas declarado. Note também que, para indicar uma classe abstrata, devemos utilizar a palavra-chave *abstract,* tanto na classe quanto no método. **E o método abstrato deve ser implementado na classe que herdará a classe abstrata.**

```
package laboratorio;

abstract public class Veiculo {

    private int velocidade;
```

```
    public void acelerar() {
        velocidade++;
    }
    abstract public void trocarMarchas();
}
```

Para finalizar o tema, acesse o material *on-line* e preste atenção às considerações do professor Ivan sobre os tipos de classe na linguagem Java!

## Objeto na linguagem Java

Na linguagem Java, toda a teoria da POO é aplicada. Principalmente pelo fato do Java ser uma linguagem totalmente orienta a objetos, ou seja, no Java tudo é um objeto.

Para criarmos um objeto no Java, ou instanciarmos uma classe, utilizamos o operador *new*, como no exemplo:

Classe nome_do_objeto = *new* Classe();

Onde ‘Classe’ é o nome da classe que será instanciada e ‘nome_do_objeto’, o nome que o objeto terá durante sua existência.

Quando necessitamos chamar algum método de algum objeto na linguagem Java, utilizamos o operador '.', como no exemplo a seguir, onde estamos chamando o método 'metodo()' do objeto 'nome_do_objeto':

```
nome_do_objeto.metodo();
```

Este operador também pode ser utilizado para referenciar atributos quando estes são visíveis (veremos posteriormente o que isso significa).

Uma característica dos objetos na linguagem Java é que todos derivam indiretamente da classe Object, ou seja, alguns métodos são padronizados, como o método 'toString()', por exemplo.

No material *on-line*, o professor Ivan esclarece a criação e utilização dos objetos na linguagem Java! Acompanhe!

## Trocando ideias

Tanto a interface quanto a classe abstrata são importantes no desenvolvimento de sistemas em Java. Pensando nisso, reflita sobre a seguinte questão:

**Quais seriam as principais diferenças na utilização de interface e classe abstrata em um sistema?**
DICA: Converse com seus colegas de curso e procure exemplos e opiniões em fóruns e *sites* de informática para fundamentar seus argumentos, realizando uma comparação sólida! E, claro, não se esqueça de levar essa bagagem de informação para sua vida profissional.

## Na prática

Vamos agora realizar um exercício prático para reforçar os conhecimentos sobre linguagem Java apreendidos nessa aula!

Defina uma interface que represente uma conta bancária que declare os métodos *saque* e *deposito*. A seguir, implemente a interface representando três classes:

- Uma classe para conta corrente;
- Uma classe para conta poupança;
- Uma classe para conta salário.

DICA: Leve em consideração as boas práticas estudadas e, se quiser, pesquise outras!

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 dessa aula!**

Estudamos mais a fundo os elementos considerados como os pilares da POO, que são a classe e o objeto. Vimos também como esses elementos são utilizados quando declarados na linguagem Java, além da utilização da interface e da classe abstrata.

É importante que você leve os conhecimentos adquiridos e padrões para o cotidiano, se habituando a trabalhar com eles desde o começo de sua carreira como programador. Pode ter certeza de que isso aumentará a qualidade, rapidez e facilidade de seu trabalho!

Antes de ir, não deixe de acessar o material *on-line* e assistir à síntese em vídeo do professor Ivan!

**Até a próxima!**

## Referências

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