

transforme ■ se



JAVA

AULA 6 - Orientação a Objeto



Orientação a Objeto



Orientação a Objeto

A teoria da Orientação a Objetos (OO) é extremamente importante para o mercado atual de software, e, além disso, a linguagem Java se baseia nesse paradigma de desenvolvimento. Os conceitos da OO são essenciais para os padrões atuais de desenvolvimento de software.

Definições sobre objetos

Na OO, objeto é uma abstração dos objetos reais existentes. Em uma sala de aula, por exemplo, existem diversos objetos: alunos, cadeiras, mesas, lousa etc. Se for necessário manter controle de uma sala de aula, pode ser elaborado um software que manipula objetos desse tipo.

Na OO, objeto é uma abstração dos objetos reais existentes. Em uma sala de aula, por exemplo, existem diversos objetos: alunos, cadeiras, mesas, lousa etc. Se for necessário manter controle de uma sala de aula, pode ser elaborado um software que manipula objetos desse tipo. No cotidiano vivemos cercados por objetos de diversos tipos e formas. O contato com esses objetos nos leva a identificar suas características físicas, sua forma etc.

Definições sobre objetos

Ao visualizar um objeto qualquer, como uma conta bancária, por exemplo, reconhecemos seu número, cliente, saldo, enfim, suas diversas propriedades. Outros objetos possuem diferentes propriedades. Associado às propriedades do objeto existe outro fator: as ações que podem ser realizadas com ele. Voltando ao exemplo da conta bancária, ela pode estar bloqueada ou desbloqueada. Existe um procedimento para consultar, depositar e sacar. A programação orientada a objetos procura modelar, internamente no computador, a realidade dos objetos.

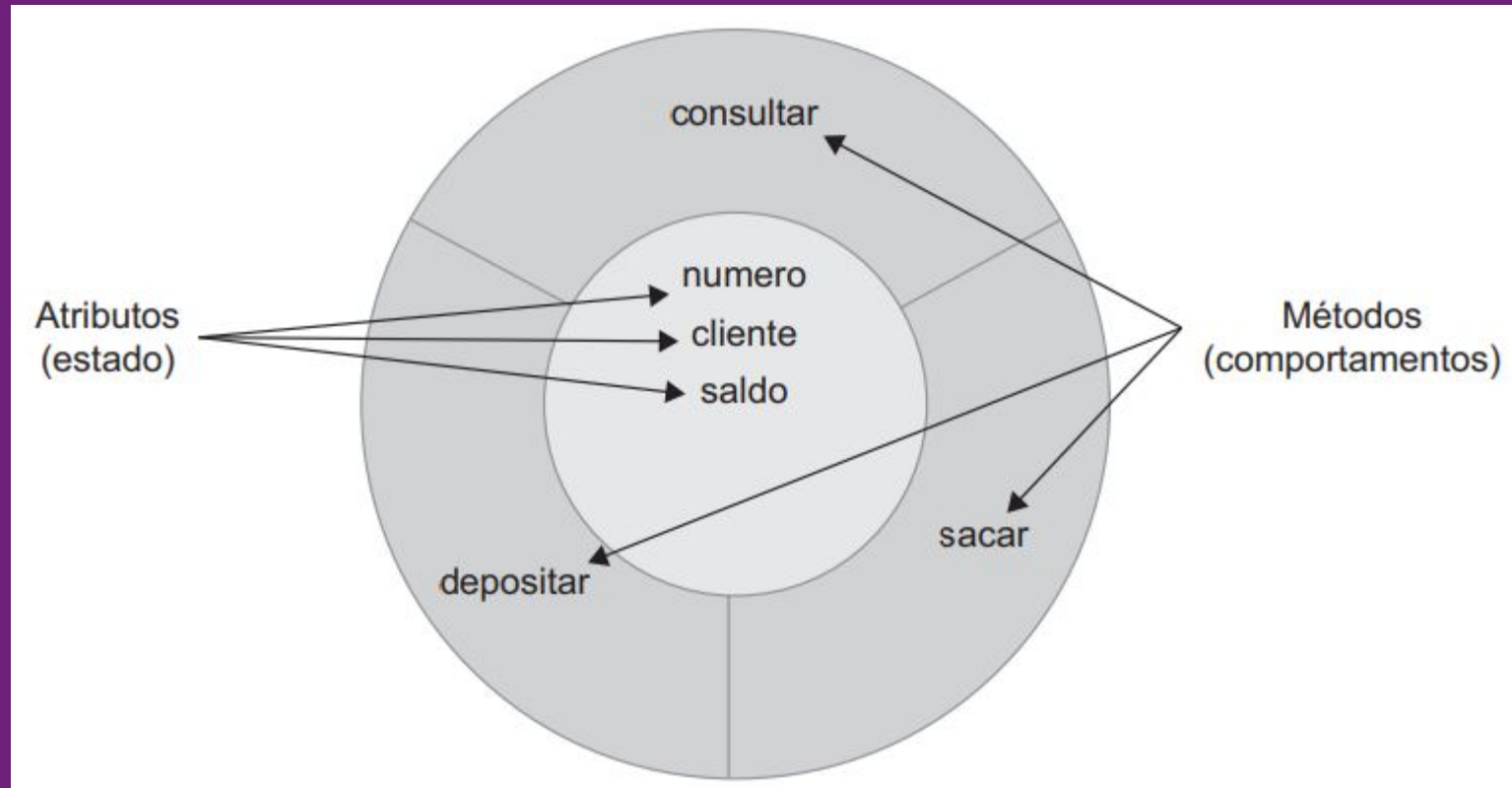
Definições sobre objetos

Apesar de existirem diferentes tipos de objetos, eles compartilham duas características principais: todos possuem um estado (conjunto de propriedades do objeto) e um comportamento (as ações possíveis sobre o objeto).

No caso da conta bancária, o estado poderia ser o conjunto de propriedades “numero”, “cliente”, “saldo” etc.; já o comportamento poderia ser composto por “consultar”, “depositar”, “sacar” etc. Voltando ao estado, um objeto conta bancária pode ter um atributo chamado “status” que assuma um dos valores seguintes: bloqueada, desbloqueada etc. Em cada etapa da vida de um objeto, ele pode assumir um estado diferente.

Para armazenar o estado, um objeto de software utiliza uma ou diversas variáveis. Já o comportamento do objeto é definido pelo conjunto de métodos que ele possui.

Definições sobre objetos



Classes


Os objetos descritos no item anterior são criados a partir das classes. Os objetos são instâncias de classe que permanecem em memória e mantêm seus valores de maneira individual. A criação de uma classe deve anteceder a de um objeto, uma vez que este é criado a partir daquela. Uma classe é um molde, um modelo, um protótipo a partir do qual os objetos podem ser criados.

Ao definir uma classe, podem ser criados muitos objetos a partir dela. Imagine a classe como o projeto de um veículo criado por uma empresa montadora. Depois de definidos todos os aspectos importantes do projeto do veículo (partes que o compõem, lista de peças, ações a serem executadas sobre o veículo etc.), podem ser criados muitos veículos, sempre tendo como base o projeto original.

Classes

Na orientação a objetos também é assim, pois uma vez definidas todas as partes que compõem uma classe, isto é, seus atributos (estado) e métodos (comportamento), é possível criar objetos que receberão suas características. No entanto, apesar de todos os objetos serem criados a partir de uma mesma classe, eles podem manter conteúdos individuais em suas variáveis, como, por exemplo, a cor da lataria, a potência do motor etc.

Para definir uma classe é usada a palavra reservada `class` seguida do nome da classe, mais o par de chaves, como a declaração a seguir:




```
1  qualificador class Nome-da-classe{  
2      // atributos  
3      // métodos  
4  }
```

Classes

O qualificador da classe indica como a classe (ou outro identificador) pode ser usada por outras classes, isto é, define sua visibilidade. Existem diversos tipos de qualificador, entre os quais serão citados no momento `public` e `private`.

O qualificador `public` indica que o conteúdo da classe pode ser usado livremente por outras classes do mesmo pacote ou de outro pacote, isto é, a classe será visível a todas as classes, estejam elas no mesmo pacote ou não.



```
1  qualificador class Nome-da-classe{  
2      // atributos  
3      // métodos  
4  }
```

Classes

Vamos apresentar uma classe para simular um objeto do mundo real. Considere um televisor. Existem muitos modelos e formatos para um televisor. Vamos definir quais atributos e métodos são essenciais para o sistema no qual o objeto do tipo Televisor será usado.

Essa classe não é executável, uma vez que não possui o método main. Por esse motivo, não é possível executá-la, como vínhamos fazendo. Em outras palavras, se você tentar executar a classe vai aparecer uma mensagem de erro informando que o método main não foi encontrado. Essa classe será usada para criar objetos que serão declarados em outras classes.

```
1  public class Televisor {
2
3      int volume;
4      int canal;
5
6      public void aumentarVolume() {
7          volume++;
8      }
9
10     public void reduzirVolume(){
11         volume--;
12     }
13
14     public void trocarCanal(int c){
15         canal = c;
16     }
17
18     public void mostrar(){
19         System.out.println("Canal: " + canal + " Value: " + volume);
20     }
21
22 }
```

Utilização de objetos da classe

Conforme já citado, uma classe permite criar objetos que podem ser utilizados em outras classes ou aplicações. Para utilizar um objeto, existem três partes a serem consideradas:

A **declaração do objeto**: segue o mesmo padrão de declarações para tipos primitivos, isto é, nome- -do-tipo nome-da-variável. Para declarar objeto, é usada a seguinte sintaxe: nome-da-classe nome-do-objeto. No caso, como desejamos gerar um objeto a partir da classe Televisor, criada anteriormente, a sintaxe será: Televisor televisor1.

O nome Televisor se refere à classe a partir da qual o objeto “televisor1” está sendo declarado.

Utilização de objetos da classe

Vale a pena ressaltar que a declaração não cria o objeto em si, trata-se apenas de uma declaração dizendo que `televisor1` é um objeto do tipo `Televisor`.



```
1 public static void main(String[] args) throws Exception {  
2     Televisor televisor;  
3 }
```

Utilização de objetos da classe

Conforme já citado, uma classe permite criar objetos que podem ser utilizados em outras classes ou aplicações. Para utilizar um objeto, existem três partes a serem consideradas:

A **instanciação do objeto**: corresponde à criação do objeto pela alocação de memória para armazenar informações sobre ele, semelhante ao que ocorre na declaração de uma variável qualquer, isto é, são reservados endereços de memória para armazenar os dados correspondentes. Para realizar a instanciação de um objeto qualquer, é usado o operador new. Seguindo o exemplo, a sintaxe será nome-do-objeto = new (“inicialização-do-objeto”).



```
1
2 public static void main(String[] args) throws Exception {
3     Televisor televisor = new Televisor();
4 }
```

Utilização de objetos da classe

Conforme já citado, uma classe permite criar objetos que podem ser utilizados em outras classes ou aplicações. Para utilizar um objeto, existem três partes a serem consideradas:

A **inicialização do objeto**: corresponde ao processo de definir valores iniciais às variáveis do objeto. Conforme apresentado no item anterior, a inicialização é precedida pelo operador `new`. Para inicializar um objeto, é usado o método construtor.

Por enquanto o método construtor será usado em sua forma default (com o mesmo nome da classe sem nenhum parâmetro associado) e as variáveis terão seus valores iniciais conforme definidos na classe. Quando as variáveis não possuem um valor inicial definido em sua classe, ao criar um objeto todas as variáveis recebem um valor padrão: “0” (zero) para variáveis numéricas, “false” para variáveis booleanas e “null” para objetos.

Utilização de objetos da classe

Dessa forma, a partir da definição de uma classe podem ser criados inúmeros objetos que passam a conter o estado e o comportamento declarados na classe. Essa é a base de funcionamento de toda a linguagem Java. Os procedimentos de declaração, instanciação e inicialização de objetos podem ser realizados em uma ou duas linhas de código. Segue a sintaxe das duas formas possíveis:

Em uma única linha:

```
nome-da-classe nome-do-objeto = new nome-do-construtor();
```

Em duas linhas:

```
nome-da-classe nome-do-objeto;  
nome-do-objeto = new nome-do-construtor();
```

Utilização de objetos da classe

Ao instanciar um objeto do tipo Televisor, ele passa a possuir todos os atributos e métodos definidos em sua classe. No caso, um objeto “televisor” conterá os dois atributos e os três métodos definidos na classe Televisor.



```
1  public static void main(String[] args) throws Exception {  
2      Televisor televisor = new Televisor();  
3      televisor.volume = 150;  
4      televisor.canal = 10;  
5  
6      televisor.aumentarVolume();  
7      televisor.trocarCanal(20);  
8      televisor.mostrar();;  
9  }
```

Escopo de classe e escopo de instância

Escopo de classe e escopo de instância existem para atributos e métodos. Vamos iniciar o estudo considerando apenas o uso em atributos. Conforme demonstrado anteriormente, cada objeto (ou instância) mantém o conteúdo das variáveis de forma exclusiva, isto é, cada objeto mantém um conteúdo diferente para um mesmo atributo da classe. Entretanto, em muitos casos não é necessário manter um conteúdo exclusivo para cada objeto, isto é, para todos os objetos da classe a variável pode conter o mesmo conteúdo.

Essa característica faz com que o conteúdo da variável seja controlado pela própria classe e não pelos objetos individualmente. Para que essa funcionalidade seja possível, basta declarar a variável como sendo static, ou seja, o conteúdo da variável será estático, controlado apenas pela classe.



```
1 public static void main(String[] args) throws Exception {  
2     System.out.println(MyMath.PI);  
3 }
```

Mensagens

Agora vamos definir outro conceito em OO: mensagem entre objetos. Já aprendemos que um objeto pode possuir diversos métodos definidos em sua classe. Em uma aplicação real é muito comum que existam diversos tipos de objetos e que um objeto necessite realizar uma tarefa que já está definida em outro objeto, ou seja, numa aplicação pode haver comunicação e interação entre objetos por meio de mensagens.

Em outras palavras, um objeto x pode necessitar de um procedimento (método) já definido em um objeto y. Para realizar esse processo, o objeto x solicita ao objeto y que execute o método, ou seja, uma mensagem nada mais é do que o fato de um objeto chamar um método de um outro objeto (ou ainda um método estático de uma classe).

Mensagens

Agora vamos definir outro conceito em OO: mensagem entre objetos. Já aprendemos que um objeto pode possuir diversos métodos definidos em sua classe. Em uma aplicação real é muito comum que existam diversos tipos de objetos e que um objeto necessite realizar uma tarefa que já está definida em outro objeto, ou seja, numa aplicação pode haver comunicação e interação entre objetos por meio de mensagens.

Em outras palavras, um objeto x pode necessitar de um procedimento (método) já definido em um objeto y. Para realizar esse processo, o objeto x solicita ao objeto y que execute o método, ou seja, uma mensagem nada mais é do que o fato de um objeto chamar um método de um outro objeto (ou ainda um método estático de uma classe).

Mensagens

Digamos que um objeto “gerente” precise enviar um e-mail. Ele não sabe como fazer isso, porém o objeto “email” sabe. Então, o objeto “gerente” solicita ao “email” que faça isso por ele. Em orientação a objetos, quando um objeto x solicita a um objeto y que execute um de seus métodos, diz-se que o objeto x enviou uma mensagem ao objeto y. Uma mensagem pode conter parâmetros que são valores enviados de um objeto a outro quando um método é invocado.

transforme ■ se

O conhecimento é o poder
de transformar o seu futuro.