# Atributos e métodos de classe

## Apresentação

Atributos e métodos são elementos importantes no desenvolvimento de aplicações. Sabemos que os atributos têm a capacidade de guardar valores, que são os dados, que podem ser inseridos por um usuário ou inicializados dentro da classe. Métodos são ações ou sub-rotinas que operam sobre os dados da classe, muitas vezes servindo para acessar esses dados.

Em orientação a objetos, existem dois tipos de atributos, de instância e de classe ou estático, e a mesma regra segue para métodos. Com isso, temos métodos de instância e de classe ou estático.

Nesta Unidade de Aprendizagem, você vai estudar a diferença entre os tipos de atributos para poder fazer bom uso desses elementos em programação orientada a objetos.

Bons estudos.

**Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:**

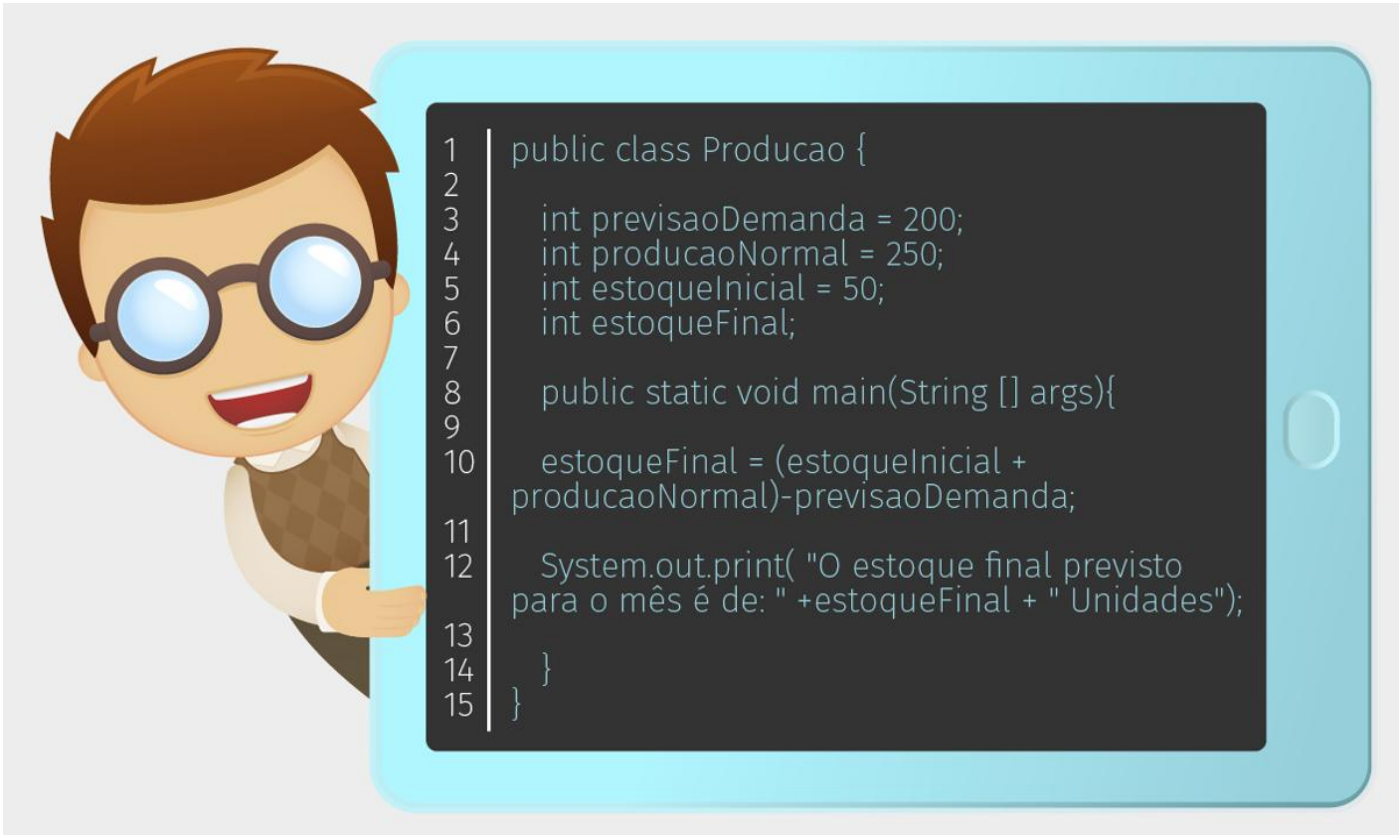
- Identificar atributos de instância e de classe ou estáticos.
- Definir métodos de instância e de classes ou estáticos.
- Construir classes com métodos e atributos de classes ou estáticos.

# Desafio

Trabalhar com métodos e atributos estáticos pode nos trazer muitas vantagens, como economia de código. Entretanto, é importante termos um bom conhecimento sobre o assunto para aplicarmos em nossos projetos. A partir disso, analise a situação a seguir.

Você trabalha como analista/programador em uma fábrica de *software* e foi encarregado de atender um cliente de uma fábrica. O cliente necessita saber se a quantidade que produz mensalmente, somada ao estoque inicial, pode atender à demanda do mês. Entretanto, a aplicação não está rodando. Com análises preliminares, você descobriu que
o problema está na classe “Producao”.

Verifique o código da classe e corrija o problema. O código da classe “Producao” atual está assim:



# Infográfico

Em linguagens orientadas a objetos como o Java, é necessário lidar com diferentes cenários, nos quais os objetos representam “coisas” do mundo real. Os atributos descrevem um objeto e, por isso, são denominados "propriedades". Os objetos também precisam executar ações, e, por isso, os métodos são aplicados. Além disso, as classes definem as características dos objetos, que permitem instanciar objetos, inicializar atributos e invocar os métodos.

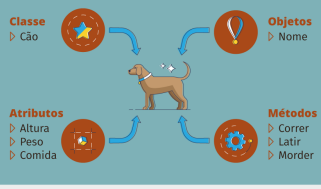
No Infográfico a seguir, veja as características de atributos e métodos estáticos ou de classe.

# CARACTERÍSTICAS DE ATRIBUTOS E MÉTODOS DE CLASSE

Os objetos são os principais componentes da programação orientada a objetos, porém não trabalham sozinhos e precisam ser combinados com atributos e métodos de classes para garantir a funcionalidade de um programa. Confira a ilustração a seguir.

Observe que o objeto "cão", dentro da classe principal "cachorros", pode ter diferentes raças, que são as classes filhas.

Os atributos são as características dos objetos, e os métodos representam as ações: "latir", "correr", "morder".




**Conheça as características de cada um deles a seguir.**

## ATRIBUTOS DE CLASSE

Também denominados "atributos estáticos", compartilham o estado com todos os objetos da mesma classe, ao passo que os atributos de instância são dinâmicos.

**Exemplo**

Suponha que, para o mesmo sistema, você precise criar um módulo para cadastro dos clientes.

Cada cliente apresenta características próprias de identificação, como: **nome, CPF e data de nascimento. Estes são os atributos dos clientes.**

Veja como fica:

```
package javamundo;

public class Cliente {

    // Atributos da classe ser humano
    public String nome;
    public int idade;
    public int RG;
    public int CPF;
```

## ATRIBUTOS DE INSTÂNCIA

Determinam o estado de uma classe e são independentes para cada objeto.

**Exemplo**

Imagine que você crie um módulo para o sistema bancário denominado "cadastro", composto pela classe **conta**. Os atributos de instância seriam:

- **número da conta;**
- **titular;**
- **saldo.**

A classe e os atributos de instância seriam criados da seguinte forma:

```
package br.com.fw3;
public class Conta {
}

private int numeroConta;
    private String titular;
    //representa o saldo atual dessa conta
    private double saldo; }
```

## MÉTODOS ESTÁTICOS

Um método estático (*static*), ou de classe, pode ser chamado mesmo que qualquer objeto da classe tiver sido instanciado.

Executa uma função sem dependência de conteúdo, **realizando a chamada de qualquer método**.

Confira um exemplo de soma de valores:

```
public class Teste_Metodo_Static {

  public static void main(String[] args) {
    double num1 = 8.5;
    double pi = Math.PI;

    System.out.println("Valor num1 = "+num1);
    System.out.println("Valor PI = "+pi);


    System.out.println("Soma dos valores = "+(num1+pi));

  }
```

Como pode ser visto, atributos e métodos são importantes para definir o comportamento e as características de objetos.

Na linguagem Java, a utilização de atributos e métodos é importante para a declaração e a invocação de objetos.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

# Conteúdo do Livro

Membros estáticos ou de classe podem nos trazer uma redução significativa de código e espaço em memória, sem contar com as vantagens funcionais. Portanto, o domínio destes conceitos torna-se vital para o desenvolvimento de *softwares*.

Acompanhe o trecho do livro **Conceitos de computação em Java**, base teórica desta Unidade de Aprendizagem, e confira mais detalhes sobre o atributos, métodos e classes.

Boa leitura.

CAY HORSTMANN

# CONCEITOS DE COMPUTAÇÃO COM

Compatível com
Java 5 & 6

# Java

5ª Edição

bookman®

## Sobre o autor

CAY S. HORSTMANN é professor de ciência da computação no departamento de ciência da computação da Universidade Estadual de San Jose. É experiente programador, foi vice-presidente e gerente de tecnologia da Preview Systems, Inc. Presta consultoria em C++, Java, Windows e programação Internet para importantes corporações, universidades e organizações. Horstmann é autor de muitos livros profissionais e acadêmicos bem-sucedidos, incluindo *Padrões e Projetos Orientados a Objetos*, *Big Java*, *Conceitos de Computação com o Essencial de C++* (publicados pela Bookman Editora sob esses títulos), *Big C++* e *Core Java*, com Gary Cornell.


H819c Horstmann, Cay.
Conceitos de computação em Java [recurso eletrônico] / Cay Horstmann ; tradução Edson Furmankiewicz. – 5. ed. – Dados eletrônicos – Porto Alegre : Bookman, 2009.

Editado também como livro eletrônico em 2009.
Conteúdo: Capítulos 16, 17, 18 e apêndices de D a M disponíveis em: www.bookman.com.br.
ISBN 978-85-7780-407-8

1. Computação – Linguagem de programação. I. Título.

CDU 004.438JAVA

Catalogação na publicação: Renata de Souza Borges CRB-10/Prov-021/08

## 8.6 Métodos estáticos

Um método estático não é invocado em um objeto.

Às vezes você precisa de um método que não seja invocado a partir de um objeto. Chamamos esse método de método estático ou de *método de classe*. Por outro lado, os métodos que você escreveu até agora são freqüentemente chamados de *métodos de instância* porque operam em uma instância particular de um objeto.

Um exemplo típico de um método estático é o método `sqrt` da classe `Math`. Quando você chama `Math.sqrt(x)`, você não fornece nenhum parâmetro implícito. (Lembre-se de que `Math` é o nome de uma classe, não de um objeto.)

Por que você iria querer escrever um método que não opera em um objeto? A razão mais comum é encapsular algum cálculo que envolve apenas números. Como números não são objetos, você não pode invocar métodos neles. Por exemplo, a chamada `x.sqrt()` nunca será válida em Java.

Eis um exemplo típico de um método estático que realiza alguns cálculos algébricos simples: calcular a porcentagem `p` da quantia `a`. Como os parâmetros são números, o método não opera em absolutamente nenhum objeto, portanto nós o tornamos um método estático:

```
/**
   Calcula uma porcentagem de uma quantia.
   @param p porcentagem a aplicar
   @param a quantia à qual a porcentagem é aplicada
   @return p porcentagem de a
*/
public static double percentOf(double p, double a)
{
   return (p / 100) * a;
}
```

Você precisa encontrar um local para esse método. Vamos pensar em uma nova classe (semelhante à classe `Math` da biblioteca Java padrão). Como o método `percentOf` tem a ver com cálculos financeiros, projetaremos uma classe `Financial` para armazená-lo. Eis a classe:

```
public class Financial
{
   public static double percentOf(double p, double a)
   {
      return (p / 100) * a;
   }
   // Outros métodos financeiros podem ser adicionados aqui.
}
```

Ao chamar um método estático, você fornece o nome da classe que contém o método para que o compilador possa localizá-lo. Por exemplo,

```
double tax = Financial.percentOf(taxRate, total);
```

Observe que você não fornece um objeto do tipo `Financial` ao chamar o método.

Agora podemos dizer por que o método `main` é estático. Quando o programa inicia, não há nenhum objeto. Portanto, o *primeiro* método no programa deve ser um método estático.

Talvez você esteja se perguntando por que esses métodos são chamados estáticos. O significado normal da palavra *estático* (“permanecer fixo em um lugar”) não parece estar relacionado com aquilo que os métodos estáticos fazem. Na realidade, essa palavra foi adotada por acidente. Java usa a palavra-chave `static` porque C++ a usa no mesmo contexto. C++ usa `static` para indicar métodos de classe porque os criadores de C++ não queriam criar uma outra palavra-chave. Alguém observou que havia uma palavra-chave

raramente utilizada, `static`, que indica algumas variáveis que permanecem em uma localização fixa para múltiplas chamadas de método. (Java não tem esse recurso, nem precisa dele.) Acabou-se descobrindo que a palavra-chave poderia ser reutilizada para indicar métodos de classe sem confundir o compilador. O fato de que ela pode confundir as pessoas aparentemente não foi uma grande preocupação. Você simplesmente tem de conviver com o fato de que "método estático" significa "método de classe": um método que não opera em um objeto e que só tem parâmetros explícitos.

### AUTOVERIFICAÇÃO DA APRENDIZAGEM

12. Suponha que Java não tivesse métodos estáticos. Todos os métodos da classe `Math` seriam então métodos de instância. Como você calcularia a raiz quadrada de $x$?
13. Harry entrega seu dever de casa, um programa que executa o jogo-da-velha. A solução dele consiste em uma única classe com muitos métodos estáticos. Por que isso não é uma solução orientada a objetos?

## 8.7 Campos estáticos

Às vezes, você precisa armazenar valores fora de um objeto específico. Utilize campos estáticos para esse propósito. Eis um exemplo típico. Utilizaremos uma versão da nossa classe `BankAccount` em que cada objeto conta bancária tem um saldo e um número de conta:

```
public class BankAccount
{
   . . .
   private double balance;
   private int accountNumber;
}
```

Queremos atribuir números de conta seqüencialmente. Isto é, queremos que o construtor de conta bancária crie a primeira conta com o número 1001, a próxima com o número 1002 e assim por diante. Portanto, devemos armazenar o último número de conta atribuído em algum lugar.

Não faz sentido, porém, transformar esse valor em um campo de instância:

```
public class BankAccount
{
   . . .
   private double balance;
   private int accountNumber;
   private int lastAssignedNumber = 1000; // NÃO – não funcionará
}
```

Nesse caso, cada *instância* da classe `BankAccount` teria um valor próprio de `lastAssignedNumber`.

Um campo estático pertence à classe, não a um objeto da classe.

Em vez disso, precisamos ter um único valor de `lastAssignedNumber` que seja o mesmo para toda a *classe*. Esse campo é chamado campo estático, porque você o declara utilizando a palavra-chave `static`.

```
public class BankAccount
{
   . . .
   private double balance;
   private int accountNumber;
   private static int lastAssignedNumber = 1000;
}
```

Cada objeto `BankAccount` tem campos de instância `balance` e `accountNumber` próprios, mas há apenas uma única cópia da variável `lastAssignedNumber` (veja Figura 4). Esse campo é armazenado em um local separado, fora de qualquer objeto `BankAccount`.

Um campo estático às vezes é chamado *campo de classe* porque há um único campo para toda a classe.

Cada método de uma classe pode acessar seus campos estáticos. Eis o construtor da classe `BankAccount`, que incrementa o último número atribuído e então o usa para inicializar o número de conta do objeto a ser construído:

```
public class BankAccount
{
   public BankAccount()
   {
      // Gera o próximo número de conta a ser atribuído
      lastAssignedNumber++; // Atualiza o campo estático

      // Atribui o campo ao número de conta dessa conta bancária
      accountNumber = lastAssignedNumber; // Configura o campo de instância
   }
   . . .
}
```

Como você inicializa um campo estático? Você não pode configurá-lo no construtor da classe:

```
public BankAccount()
{
   lastAssignedNumber = 1000; // NÃO – seria redefinido para 1000 a cada novo objeto
   . . .
}
```

Assim, a inicialização ocorreria toda vez que uma nova instância fosse construída.

Há três maneiras de inicializar um campo estático:

1. Não fazer nada. O campo estático é então inicializado com `0` (para números), `false` (para valores `boolean`) ou `null` (para objetos).
2. Utilizar um inicializador explícito, como:

```
public class BankAccount
{
   . . .
   private static int lastAssignedNumber = 1000;
}
```

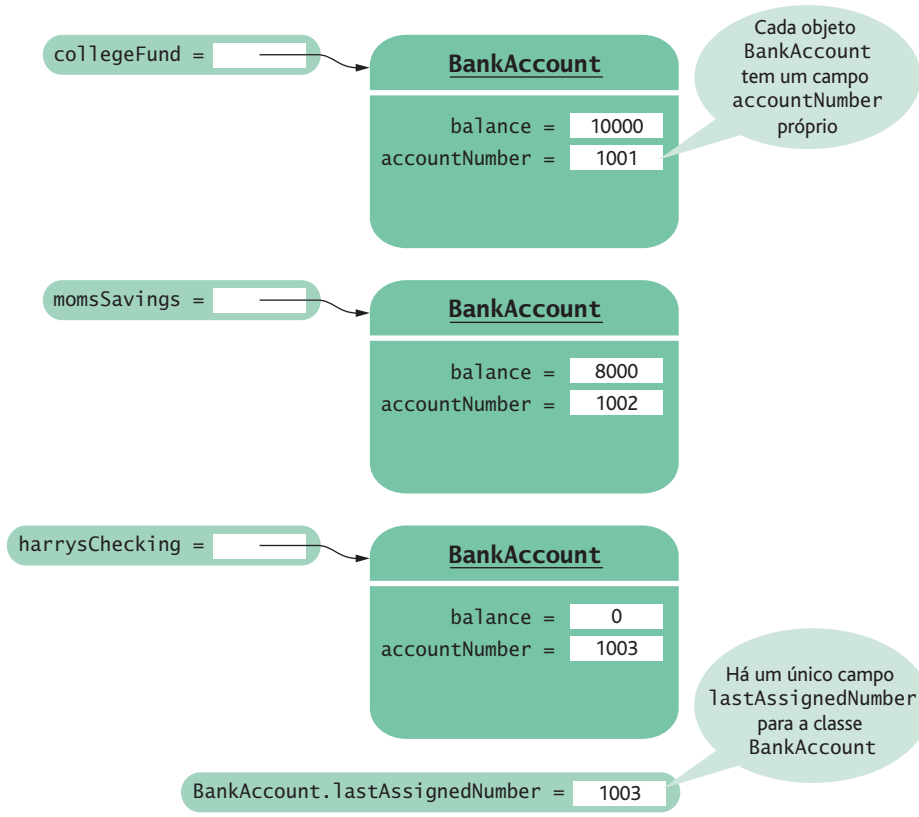



**Figura 4** Um campo estático e campos de instância.

A inicialização é executada depois que a classe é carregada.

3. Utilizar um bloco de inicialização estático (ver Tópico Avançado 8.3).

Como ocorre com campos de instância, campos estáticos sempre devem ser declarados como `private` para assegurar que os métodos das outras classes não alterem seus valores. A exceção a essa regra são as *constantes* estáticas, que podem ser privadas ou públicas. Por exemplo, a classe `BankAccount` poderia definir o valor de uma constante pública, como

```
public class BankAccount
{
   . . .
   public static final double OVERDRAFT_FEE = 5;
}
```

Métodos de qualquer classe referenciam essa constante como `BankAccount.OVERDRAFT_FEE`.

Faz sentido declarar constantes como `static` – você não iria querer que cada objeto da classe `BankAccount` tivesse seu próprio conjunto de variáveis com os valores dessas constantes. É suficiente ter um conjunto delas para a classe.

Por que as variáveis de classe são chamadas `static`? Como ocorre com os métodos estáticos, a própria palavra-chave `static` é simplesmente uma remanescente sem sentido

de C++. Mas campos estáticos e métodos estáticos têm muito em comum: eles são aplicados a toda a *classe*, não a instâncias específicas da classe.

Em geral, é recomendável minimizar o uso dos campos e métodos estáticos. Se encontrar utilizando vários métodos estáticos é uma indicação de que talvez você não tenha encontrado as classes corretas para resolver seu problema de uma maneira orientada a objetos.

### Autoverificação da aprendizagem

14. Cite dois campos estáticos da classe `System`.
15. Harry informa que encontrou uma excelente maneira de evitar esses objetos incômodos: colocar todo o código em uma única classe e declarar todos os métodos e campos como `static`. Então `main` pode chamar os outros métodos estáticos e todos eles podem acessar os campos estáticos. O plano do Harry funcionará? Ele é uma boa idéia?

### Tópico avançado 8.3

**Formas alternativas de inicialização de campos**

O Tópico Avançado 8.3 abrange dois mecanismos menos comuns para inicialização de campo: especificar os valores iniciais para os campos e usar blocos de inicialização.

# Dica do Professor

Os atributos de classe definem as propriedades de um objeto. Na linguagem Java, utilizada na programação orientada a objetos, também são conhecidos como variáveis ou campos. As propriedades são utilizadas em um código-fonte para definir o estado de um objeto, permitindo também que os valores possam ser alterados quando necessário.

Na Dica do Professor, acompanhe como funciona a aplicação de atributos de classe.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

# Exercícios

1) **Na programação orientada a objetos, uma variável é um recurso capaz de armazenar um ou mais valores na memória do programa. O valor pode ser acessado em qualquer parte do programa, sempre que necessário, desde que seja declarado corretamente.**

**Sobre as varáveis, marque a alternativa correta.**

A) Uma variável estática representa informações em nível de classe.

B) Uma variável estática não muda seu valor.

C) As variáveis de classe são usadas quando apenas um objeto da classe precisa utilizar uma cópia dessa variável.

D) Todo objeto tem sua própria cópia de todas as variáveis estáticas da classe.

E) Se vários objetos precisam acessar determinada variável em comum, à medida que se transforma essa variável em estática, há desperdício de memória.

2) **As palavras-chaves, ou palavras reservadas, são utilizadas na estrutura de uma linguagem de programação para chamar métodos, funções e recursos nativos, garantindo que o termo não possa ser utilizado pelo desenvolvedor para outras funcionalidades. Da mesma forma, a declaração de uma variável de classe utiliza uma palavra-chave para ser invocada.**

**Selecione a alternativa que representa a palavra-chave correta.**

A) *public*.

B) *class*.

C) *void*.

D) *private*.

E) *static*.

3) **Um método em Java é equivalente a uma função, sub-rotina ou procedimento em outras linguagens de programação. O Java não trata dos métodos como globais. Cada método deve**

**ser definido dentro de uma classe conforme suas características próprias.**

**Em relação aos métodos estáticos ou de classes, marque a afirmativa correta.**

A) Métodos são variáveis que têm a capacidade de receber uma quantidade maior de informação.

B) Métodos estáticos não são membros de classes.

C) Métodos estáticos servem apenas para operações de inserção de dados.

D) Por serem estáticos, os métodos não podem ser chamados em outras classes.

E) Métodos estáticos são declarados colocando-se a palavra-chave *static* antes do tipo de retorno.

4) **Os métodos de classes são definidos como funções independentemente de qualquer variável de instância atribuída aos objetos. Os métodos são invocados sem depender de nenhum objeto e suas características.**

**Analise as afirmativas a seguir e marque a alternativa correta:**

**I. Para um método de classe acessar membros de classe não estáticos, deve-se colocar a palavra-chave *static* no membro a ser acessado.**

**II. Um método estático não pode acessar membros de classe não estáticos.**

**III. Para acessar membros de classe não estáticos, é preciso colocar a palavra-chave *protected* no membro a ser acessado e declarar o método usando o *get* na frente do seu nome.**

**Está correto o que se afirma em:**

A) I, apenas.

B) III, apenas.

C) II, apenas.

D) I e II, apenas.

E) I e III, apenas.

5)

**A programação orientada a objetos é um modelo de programação em que o desenvolvedor pode trabalhar com diferentes linguagens de programação. Em Java, é possível trabalhar com várias classes, que têm as características de definir objetos na vida real. Cada classe determina o comportamento de um objeto. Tanto as classes como os métodos são importantes e marcados por diferentes estados definidos por propriedades.**

**Nesse contexto, marque a alternativa correta.**

A) As variáveis e os métodos de classe estáticos existem apenas quando um objeto dessa classe tiver sido instanciado.

B) A referência *this* pode ser usada em métodos estáticos.

C) Se um método estático tentar acessar outro método não estático da classe usando somente o nome do método, ocorrerá um erro de compilação.

D) Para fazer a chamada de um método estático, é necessário apenas colocar o nome do método seguido de parênteses.

E) Métodos e variáveis estáticas são associados a um objeto.

# Na prática

No mundo da programação orientada a objetos, as classes são os objetos abstraídos do mundo real, e é por meio das classes que todo programa é desenvolvido. As classes basicamente são estruturadas em atributos e métodos. Os atributos são associados às características (estado) dos objetos, e os métodos, às ações (comportamento).

Veja, Na Prática, como os atributos armazenam as informações e como os métodos as manipulam.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

# Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

## Adicionar um método à classe criada

Assista ao vídeo a seguir, que demonstra como inserir um método a uma classe existente em Java.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

## Princípios de programação orientada a objetos em Java: conceitos de POO para iniciantes

Leia o conteúdo, que relembra os conceitos de classes e apresenta um cenário prático com aplicação de diferentes classes em Java.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

## Como criar uma classe com atributos

Veja o vídeo a seguir, que apresenta o processo de criação de classes e vinculação de atributos.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.