uninorte

PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Edkallenn

Sistemas de Informação - Uninorte

Programação Orientada a Objetos

- **Prof. Edkallenn Lima**
- edkallenn@yahoo.com.br (somente para dúvidas)
- **Blogs**:
  - http://professored.wordpress.com (Computador de Papel – O conteúdo da forma)
  - http://professored.tumblr.com/ (Pensamentos Incompletos)
  - http://umcientistaporquinzena.tumblr.com/ (Um cientista por quinzena)
  - http://eulinoslivros.tumblr.com/ (Eu li nos livros)
  - http://linabiblia.tumblr.com/ (Eu li na Bíblia)
- **YouTube**:
  - https://www.youtube.com/user/edkallenn
  - https://www.youtube.com/channel/UC-pD2gnahhxUDVuTAA0DHoQ
- **Redes Sociais**:
  - http://www.facebook.com/edkallenn
  - http://twitter.com/edkallenn ou @edkallenn
  - https://plus.google.com/u/0/113248995006035389558/posts
  - Instagram: http://instagram.com/edkallenn ou @edkallenn
  - Foursquare: https://pt.foursquare.com/edkallenn
  - Pinterest: https://br.pinterest.com/edkallenn/
- **Telefones**:
  - 68 98401-2103 (CLARO e Whatsapp) e 68 3212-1211.

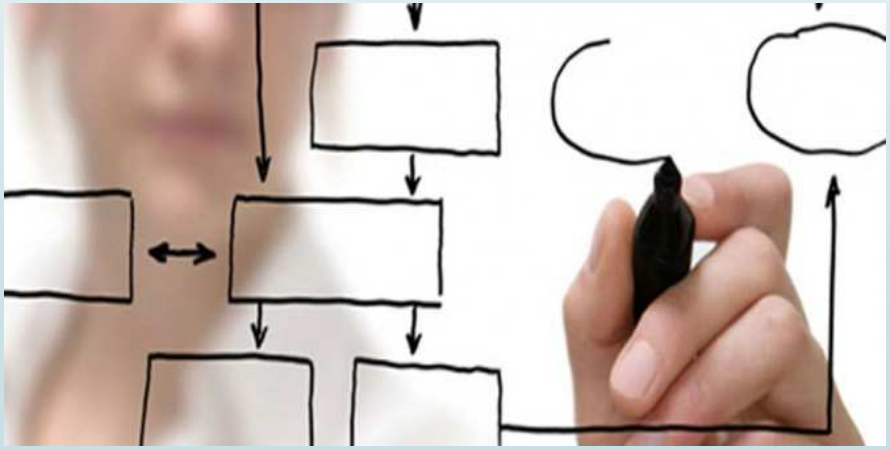
Os exercícios devem ser enviados SEMPRE para o e-mail:
edkevan@gmail.com
ou para o e-mail:
edkallenn.lima@uninorteac.com.br

Agenda

- Métodos
- Métodos especiais
- Construtores e Sobrecarga
- Escopo
- Modificadores de acesso

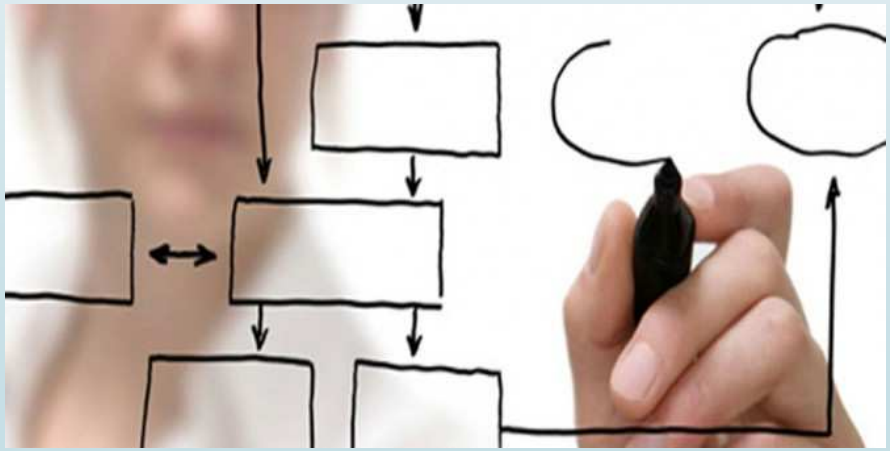
Métodos

- Tendo identificado a classe com seus atributos, a seguinte pergunta pode surgir: mas o que fazer com eles?
- Como utilizar a classe e manipular os atributos?
- É nessa hora que o método entra em cena.
- Este é responsável por identificar e executar as operações que a classe fornecerá.
- Essas operações, via de regra, têm como finalidade manipular os atributos



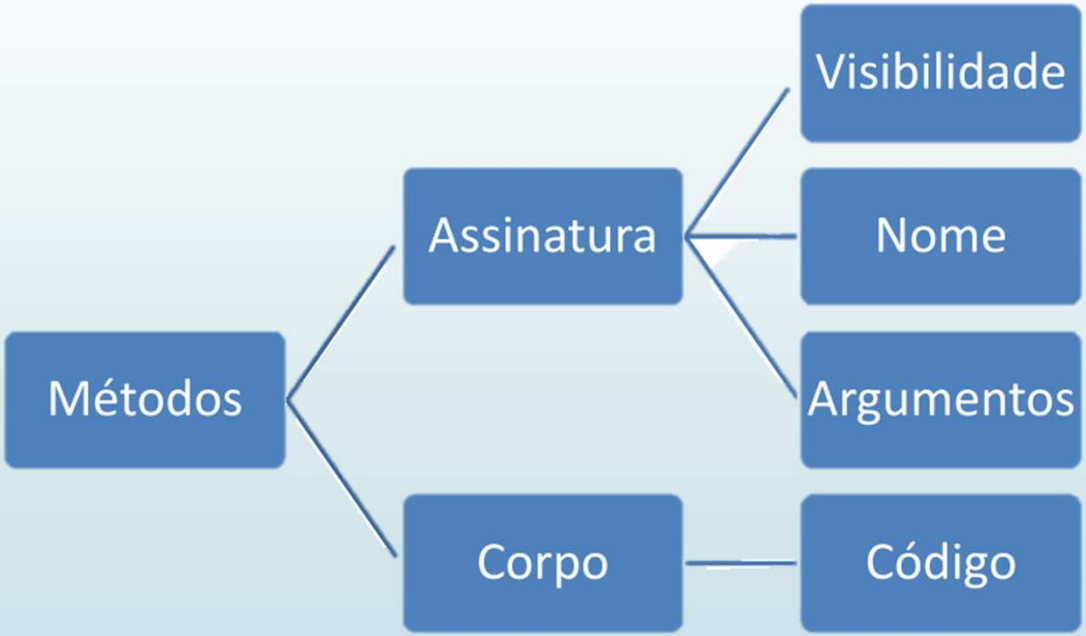
Método

- ***Método é uma porção de código (sub-rotina) que é disponibilizada pela classe. Este é executado quando é feita uma requisição a ele.***
- ***Um método serve para identificar quais serviços, ações, que a classe oferece. Eles são responsáveis por definir e realizar um determinado comportamento.***
- **Para facilitar o processo de identificação dos métodos de uma classe, podemos pensar em verbos.**
- **Isso ocorre devido à sua própria definição: *ações*. Ou seja, quando se pensa nas ações que uma classe venha a oferecer, ela identifica seus métodos.**



Assinatura do método

- **Assinatura nada mais é do que o nome do método e sua lista de parâmetros.**



Parâmetros e Retorno de métodos

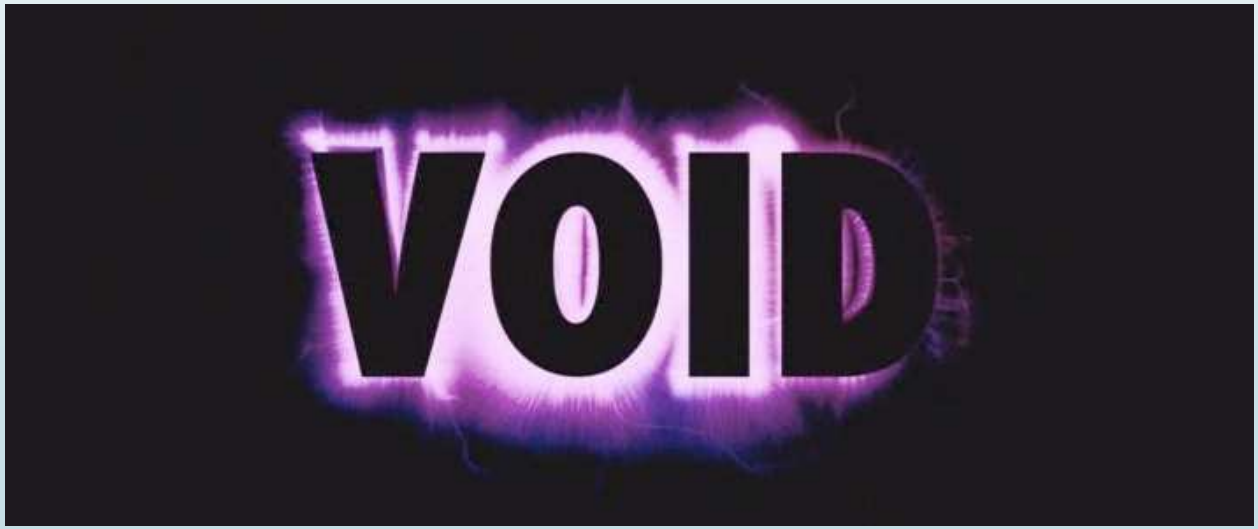
- A lista de parâmetros são informações auxiliares que podem ser passadas aos métodos para que estes executem suas ações. (entre parênteses!)
- Cada método terá sua lista específica, caso haja necessidade.
- Esta é bem livre e, em determinados momentos, podemos não ter parâmetros, como em outros podemos ter uma classe passada como parâmetro, ou também tipos primitivos e classes ao mesmo tempo

Parâmetros e Retorno de métodos

- Embora não faça parte de sua assinatura, os métodos devem possuir um retorno.
- Se uma **ação** é disparada, é de se esperar uma **reação**.
- O retorno de um método pode ser qualquer um dos tipos primitivos vistos na seção sobre atributos.
- Além destes, o método pode também retornar qualquer um dos conceitos (classes) que foram definidos para satisfazer as necessidades do sistema em desenvolvimento, ou também qualquer outra classe — não criada pelo programador — que pertença a linguagem de programação escolhida

void?

- **Quando necessitamos que um método não tenha retorno, não devemos simplesmente omitir o tipo de retorno.**
- **Na verdade, devemos usar a palavra reservada** void **, que significa que o método não retornará nada.**
- **Linguagens como** C **já disponibilizam esse recurso**
- **Métodos** void **são, na verdade, procedimentos**



Método especial: construtor

- **Independente da quantidade e finalidade dos métodos de uma classe, existem dois métodos especiais que toda classe possui: o construtor e o destrutor.**

Construtor

- O construtor é responsável por criar objetos.
- Ou seja, sempre que for necessário criar objetos de uma determinada classe, seu construtor deverá ser utilizado.
- É através do seu uso que será possível instanciar objetos e, a partir disto, manipular de forma efetiva seus atributos e métodos.
- Além disto, uma outra função do construtor é prover alguns valores iniciais que o objeto precisa ter inicialmente.
- Como Java e C# são as linguagens usadas para ilustrar os nossos conceitos, o processo de definição dos construtores nessas linguagens é o seguinte: criar um método com o mesmo nome da classe e sem retorno, podendo ou não ter parâmetros.

Construtor

- **Não precisamos definir retorno algum (o construtor pertence a esta determinada classe e a sua função é criar objetos a partir dela)**
- **Muitas linguagens —** Java **e** C# **são exemplos — possuem construtores implícitos.**
- **Ou seja, mesmo se os programadores não definirem um construtor para a classe, ele estará disponível.**
- **Por padrão, o construtor implícito tem como assinatura a já apresentada anteriormente: o mesmo nome da classe e sem parâmetros.**

Destrutor

- **O destrutor tem a função inversa: destruir o objeto criado a partir da classe.**
- **Ou seja, sempre que não precisarmos mais de objetos que foram criados a partir de uma determinada classe, devemos usar seu destrutor.**
- **É através do seu uso que poderemos eliminar os objetos criados.**
- **Ao contrário do construtor, os destrutores em** Java **e** C# **possuem sintaxes bem diferentes: em** Java **se chama** finalize **e se usa o** void **.**
- **Já em** C# **, tem o mesmo nome da classe, mas com um til (~) no início, e não usa o** void **.**
- **A ideia por detrás desse processo de eliminação dos objetos é liberar possíveis recursos que ele teve de se apoderar para realizar suas atividades, além de também simplesmente eliminá-lo.**

Deve-se usar destrutores?

- Caso haja necessidade, devemos definir os destrutores para nossas classes e futuros objetos, mas não devemos usá-los diretamente.
- Isto não é proibido, mas também não é uma boa prática.
- Na verdade, mesmo se os usarmos diretamente, ainda não teremos a certeza de que, no exato momento de seu uso, o objeto será eliminado.
- Isso ocorre devido a uma funcionalidade que as linguagens orientadas a objetos modernas proveem: o **Garbage Collector.**

Sobrecarga

- Muitas vezes, é preciso que um mesmo método possua entradas (parâmetros) diferentes.
- Isso ocorre porque ele pode precisar realizar operações diferentes em determinado contexto.
- Este processo é chamado de sobrecarga de método.
- Para realizá-la, devemos manter o nome do método intacto, mas alterar sua lista de parâmetros.
- Podem ser acrescentados ou retirados parâmetros para assim se prover um novo comportamento.

Sobrecarga

- Sempre que a lista de parâmetros muda, seja acrescentando ou eliminando parâmetros, mudando seus tipos e até mesmo sua sequência, estaremos criando sobrecargas de um método.
- Mas lembre-se de que o nome dele deve permanecer intacto

Estado de um objeto

- ***O estado de um objeto é o conjunto dos valores dos seus atributos de um determinado momento.***
- ***Estes valores podem mudar a qualquer momento e em qualquer quantidade, sob qualquer circunstância.***
- ***Com isso, o objeto pode ter quantos estados necessitar enquanto ele estiver em execução.***
- **o conceito de estado de um objeto é intrinsecamente e somente ligado aos seus atributos**

TIPOS DE ATRIBUTO E MÉTODO

- Os atributos de **instância**, embora definidos na classe, pertencem ao objeto.
- Ou seja, só poderão ser acessados/usados a partir de instâncias de uma classe, no caso um objeto.
- Com isso, cada um pode ter valores distintos para seus atributos e assim conseguir armazenar os valores que necessitam
- Ex: se em uma classe chamada Pessoa existir um atributo de instância chamado nome , poderemos criar dois objetos e cada um ter um valor em particular para este atributo.
- No caso, poderíamos ter um objeto com o valor "Isadora" para seu atributo nome , e um outro objeto com o valor "Lorena" para seu atributo nome

TIPOS DE ATRIBUTO E MÉTODO

- Por padrão, todo atributo é de instância e, para defini-los desta forma, basta criar os atributos como já vem sendo feito
- O atributo **estático** pertence à **classe**, e não ao **objeto**.
- Atributos deste tipo devem ser acessados/usados diretamente a partir da classe.
- Podem até ser acessados/usados via o objeto, mas isso não é uma boa prática e deve ser desencorajado
- Devido a esse comportamento de pertencer à classe e não ao objeto, ele se comporta de forma oposta ao de instância
- Valores armazenados neles são iguais em todos os objetos criados a partir da classe, pois eles pertencem a ela antes mesmo de existir um objeto
- Objetos distintos terão o mesmo valor para esse determinado atributo

TIPOS DE ATRIBUTO E MÉTODO

- Para definir atributos desta forma, é necessário informar que ele é estático a partir da palavra reservada static .
- Sobre o **método**: o de instância assim como o atributo são definidos na classe, mas são acessados/usados via o objeto.
- Ou seja, a execução deles só poderá ser requisitada através de um objeto.
- Ele só pode ser requisitado através de um objeto.
- Por padrão, todo método é de instância e, para definir isso, basta criá-los como já vinha sendo feito

TIPOS DE ATRIBUTO E MÉTODO

- Já o método **estático** pertence à **classe** e não ao objeto, ou seja, deve ser acessado diretamente através da classe.
- Mais uma vez, ele **executa uma ação** e ela será a mesma independente do objeto, pois antes mesmo de pertencer ao objeto, o método já pertencia à classe.
- Assim como no atributo, para definir o método como estático, a palavra reservada static deve ser usada
- Ex: main

Métodos estáticos (mais detalhes)

- De novo (e importante) métodos estáticos (ou métodos de classe) podem ser chamados sem a necessidade de criação de instâncias das classes às quais pertencem
- Métodos estáticos podem ser usados em classes que tenham o método main para servirem como sub-rotinas deste
- Também são adequados para implementar rotinas que sejam independentes de dados armazenados em classes, ou seja, métodos que só necessitem dos dados passados como argumentos para efetuar a tarefa requerida e que sejam executados como resultado independentemente de qual instância da classe a que pertencem seja usada para a sua chamada
- A aplicação mais frequente de métodos estáticos é a criação de bibliotecas de métodos, classes que contêm somente métodos estáticos, geralmente agrupados por função.
- Ex: métodos da classe Math

TRABALHO PARA O DIA 04/05/2018

- Faça um trabalho científico usando as regras da disciplina metodologia científica cobrindo os seguintes tópicos:
  - As palavras reservadas de Java e de C#
  - TODOS os tipos de dados de Java e de C# e um comparativo entre eles mostrando as semelhanças e diferenças entre as duas linguagens (com exemplos de códigos em ambas as linguagens)
  - Uma discussão sobre escopo de atributos e variáveis dentro de uma classe em Java e C#
  - Os **modificadores de acesso** (e uma discussão sobre o encapsulamento) cobrindo obrigatoriamente todos os modificadores em ambas as linguagens (Java e C# e as diferenças do uso entre as duas linguagens) → com exemplos de código em ambas as linguagens
  - Como as palavras-chave **new** e **null** funcionam (com exemplos de código).

- Enviar para o Google Class com o nome: **[POO-2018/1-TRAB01-N2]NomeSobrenomeAluno**
- **O trabalho é individual.**

Perguntas e Discussão