Sejam bem-vindos! PROGRAMAÇÃO ORIENTADA A OBJETOS



Diogenes Carvalho Matias

Formação:

- Graduação: Sistemas de Informação;
- Especialista em: Engenharia e Arquitetura de Software;
- MBA EXECUTIVO EM BUSINESS INTELLIGENCE (em andamento);
- Mestrado Acadêmico em Engenharia de Computação (UPE em andamento);

Maiores informações :



<u>Linkedin</u>





Construtores

Os construtores são os responsáveis por criar o objeto em memória, ou seja, instanciar a classe que foi definida. Eles são obrigatórios e são declarados.



```
public class Carro{
/* CONSTRUTOR DA CLASSE Carro */
public Carro(){
        //Faça o que desejar
        //na construção do objeto
```



Construtores

modificadores de acesso (public nesse caso) + nome da classe (Carro nesse caso) + parâmetros (nenhum definido neste caso).

OBS: O construtor pode ter níveis como: public, private ou protected.



```
public class Carro{
                                         Construtores
/* CONSTRUTOR DA CLASSE Carro */
public Carro(){
       //Construção do objeto
public class Aplicacao {
public static void main(String[] args) {
//Chamamos o construtor sem nenhum parâmetro
Carro celta suv = new Carro();
```



```
class Conta {
    String titular;
    int numero;
    double saldo;
    //Construtor
    Conta() {
        System.out.println("Construindo uma conta.");
```



Construtores

Por padrão, o Java já cria esse construtor sem parâmetros para todas as classes, então você não precisa fazer isso se utilizará apenas construtores sem parâmetros. Por outro lado, poderá criar mais de um construtor para uma mesma classe. Onde, podemos criar um construtor sem parâmetros, com dois parâmetros e outro com três parâmetros.



```
class Conta {
    String titular;
    int numero;
    double saldo;

    //Construtor
    Conta(String titular) {
        this.titular = titular;
    }
}
```



```
public class Carro{
private String cor;
private double preco;
private String modelo;
/* CONSTRUTOR PADRÃO */
public Carro(){
/* CONSTRUTOR COM 2 PARÂMETROS */
public Carro(String modelo, double preco){
       //Se for escolhido o construtor sem a COR do veículo
       // definimos a cor padrão como sendo Azul
this.cor = "Azul";
this.modelo = modelo;
this.preco = preco;
/* CONSTRUTOR COM 3 PARÂMETROS */
public Carro(String cor, String modelo, double preco){
this.cor = cor;
this.modelo = modelo;
this.preco = preco;
```



```
public class AplicacaoPrincipal {

public static void main(String[] args) {
  String nome = "Diogenes Carvalho Matias";
  Conta c = new Conta(nome);
  System.out.println(c.titular);
}
```



```
public class Carro{
private String cor;
private double preco;
private String modelo;
/* CONSTRUTOR PADRÃO */
public Carro(){
/* CONSTRUTOR COM 2 PARÂMETROS */
public Carro(String modelo, double preco){
        //Se for escolhido o construtor sem a COR do veículo
       // definimos a cor padrão como sendo Azul
this.cor = "Azul";
this.modelo = modelo;
this.preco = preco;
/* CONSTRUTOR COM 3 PARÂMETROS */
public Carro(String cor, String modelo, double preco){
this.cor = cor;
this.modelo = modelo;
this.preco = preco;
```

Construtores

Agora três construtores padrões, ou seja, podemos criar um novo Carro sem definir sua Cor, ou podemos criar um novo carro definindo todos os seus atributos.

Construtores

OBS: Podemos criar vários carros com diferentes construtores.

```
public class AplicacaoPrincipal {
public static void main(String[] args) {
//Construtor sem parâmetros
Carro prototipoDeCarro = new Carro();
//Construtor com 2 parâmetros
Carro celtaPreto = new Carro("New Celta","54000");
//Construtor com 3 parâmetros
Carro fuscaAmarelo = new Carro("Fusca", "88000");
```



Destrutores

Não existe o conceito de destrutores em Java, mas você nem precisa se preocupar com isto, pois não tem como literalmente destruir um objeto, assim como você faz em C/C++. Isso porque não é garantido que o **Garbage Collection** irá destruir este objeto, já que ele o faz na hora que achar conveniente e o programador não tem nenhum controle sobre isso.

A forma mais adequada de "tentar" destruir um objeto em Java é atribuir valores nulos a ele. Assim, quando o **Garbage Collection** for realizar o seu trabalho, verá que seu objeto não está sendo mais utilizado e o destruirá.



Exercício para ser feliz XD

1-Crie sua classe com seus construtores e construtores variados no mínimo 3 conforme os modelos UML abaixo:

Editoras

- + codEditora : int
- + razaoSocial : string
- + contao : string
- + telefone : string

Produtos

- + codigo : int
- + descProduto : string
- + preco:float
- + quantidade : int
- + ListaraDados() : void

Revistas

- + codigo : int
- + titulo : string
- + tipo : string
- + SetEdicao(numEdicao : int, dataEdicao : date, numArtigos : int) : void
- + GetEdicao() : void



Exercício para ser feliz XD

2-Crie sua classe com seus construtores e construtores variados no mínimo 3 conforme os modelos UML abaixo:

Pessoa

-personID : unsigned long

-surname : CString

-givenName : CString

-middleInitial : char

-streetAddress : CString

-postCode : CString

-countryname : CString

eMailAddress : CString

+Person()

+PersonInfo()

Empregado

-jobType : CString

-officeNr : unsigned long

-department : CString

-division : CString-iobTitle : CString

-reportsto : unsigned long

-headsDept : CString

-headsDivision : CString

-mobileNr : CString

-birthDate : unsigned long

+Employee()

+GetCurrentAge()

+EmployeeInfo()

Candidato

-worksforCompany : CString

experience : CString

-reference1 : CString

reference2 : CString

reference3 : CString

+Applicant()

+ApplicantInfo()

+MakeApplication()

Livros

+ codigo : int

+ descLivro : string

+ ISBN : string



Exercício para casa em casa XD

1-O cliente usa uma comanda eletrônica durante suas compras na Padaria. A cada produto consumido, o atendente registra em sua comanda(que possui uma numeração) o produto e a quantidade. Ao passar no caixa na saída da padaria, a caixa lê os gastos da comanda, finalizando a compra. Na leitura da comanda, verifica-se o valor unitário de cada produto a fim de calcular o valor total da compra.

OBS: Crie uma classe para podemos ver os construtores e os objetos sendo executados.

