Aula 06 - Construtores e Métodos

1. Construtor NÃO Padrão

Construtor não padrão é um método sem tipo, com o mesmo nome da classe e com assinatura, ou seja, parâmetros. O construtor não padrão, tem como principal objetivo ser criado para popular o novo objeto no momento de sua instância (criação).

```
public class Pessoa {
    public int id;
    public String nome;

    Pessoa(int id, String nome){
        this.id = id;
        this.nome = nome;
    }
}
```

2. Construtor Padrão

Construtor padrão é um método sem tipo, com o mesmo nome da classe e sem assinatura. O construtor padrão já existe de forma implícita, ou seja, ele não tem necessidade de estar no seu código, pois por padrão ele só serve para instanciar um objeto.

```
public class Pessoa {
    public int id;
    public String nome;

    Pessoa(){

    }
}
```

3. Programa utilizando Construtores

```
package classe_atributo;

public class Projeto {

    public static void main(String[] args) {
        // Cria um array contendo 10 posições para armazenar objetos
        Pessoa[] listaPessoas = new Pessoa[10];
```

```
        // Instancia (cria) o objeto com o construtor padrão
        Pessoa a = new Pessoa();

        // Popula o objeto (preenche os atributos do objetos)
        a.id = 1;
        a.nome = "João";

        // Adiciona o objeto no array "listaPessoas"
        listaPessoas[0] = a;

// ====================================

        // Instancia (cria) o objeto com o construtor não padrão já
populando ele
        Pessoa b = new Pessoa(2, "Maria");

        // Adiciona o objeto no array "listaPessoas"
        listaPessoas[1] = b;

// ====================================

        // Instancia (cria) o objeto com o construtor não padrão já
populando ele
        Pessoa c = new Pessoa(3, "José");

        // Adiciona o objeto no array "listaPessoas"
        listaPessoas[2] = c;

// ====================================

        System.out.printf("Pessoa %d criada com sucesso (%s)!",
listaPessoas[0].id, listaPessoas[0].nome);
        System.out.printf("\nPessoa %d criada com sucesso (%s)!",
listaPessoas[1].id, listaPessoas[1].nome);
        System.out.printf("\nPessoa %d criada com sucesso (%s)!",
listaPessoas[2].id, listaPessoas[2].nome);
    }

}
```

4. Métodos Void

Os métodos que não retornam valor tem em sua declaração a palavra reservada void e em seu corpo não tem a presença da palavra return. Esse tipo de método realiza uma ação e não retorna nenhum valor para quem o chamou.

```
public class Pessoa {
    public int id;
    public String nome;

    Pessoa(){

    }

    Pessoa(int id, String nome){
        this.id = id;
        this.nome = nome;
    }

    public void transformarNomeMaiusculo() {
        this.nome = this.nome.toUpperCase();
    }

    public void mudarNome(String novoNome) {
        this.nome = novoNome;
    }
}
```

5. Métodos NÃO Void

Os métodos que retornam valor tem em sua declaração o tipo de dados que o método irá retornar e em seu corpo deve ter a presença da palavra reservada return. Esse tipo de método realiza uma ação e retorna um valor conforme tipo de dado em sua declaração para quem o chamou.

```
public class Pessoa {
    public int id;
    public String nome;

    Pessoa(){

    }

    Pessoa(int id, String nome){
        this.id = id;
```

```
        this.nome = nome;
    }

    public void transformarNomeMaiusculo() {
        this.nome = this.nome.toUpperCase();
    }

    public String exibirIdNome() {
        return (this.id + " - " + this.nome);
    }

    public void mudarNome(String novoNome) {
        this.nome = novoNome;
    }
}
```

6. Programa utilizando Métodos

```
package classe_atributo;

import java.util.Scanner;

public class Projeto02 {

    public static void main(String[] args) {

        Scanner ler = new Scanner(System.in);

        // Cria um array contendo 10 posições para armazenar objetos
        Pessoa[] listaPessoas = new Pessoa[10];

        int id;
        String nome;

        for (int i=0; i<=2; i++) {

            // Popula o objeto (preenche os atributos do objetos)
            System.out.printf("Digite o id da pessoa: ");
            id = ler.nextInt();

            System.out.printf("Digite o nome da pessoa: ");
            nome = ler.next();
```

```
            // Instancia (cria) o objeto
            Pessoa p = new Pessoa(id, nome);

            // Adiciona o objeto no array "listaPessoas"
            listaPessoas[i] = p;
        }

        // Mudando o nome de algumas pessoas da lista, utilizando o
método "mudarNome"
        listaPessoas[0].mudarNome("Joseffe");
        listaPessoas[2].mudarNome("Jefferson");

        // Atualizando o nome para letras maiúsculas de uma pessoa da
lista
        listaPessoas[1].transformarNomeMaiusculo();

        for (int i=0; i<=2; i++) {
            System.out.printf("\nPessoa %d criada com sucesso (%s)!",
listaPessoas[i].id, listaPessoas[i].nome);

            // Exibindo os dados utilizando o método "exibirIdNome"
            System.out.printf("\n%s", listaPessoas[i].exibirIdNome());
        }
    }
}
```

6.1. Outro exemplo utilizando métodos

Pessoa.java

```
package classe_objeto;

public class Pessoa {
    public int id;
    public String nome;
    public double saldo;

    Pessoa(){

    }
```

```
    Pessoa(int id, String nome, double saldo){
        this.id = id;
        this.nome = nome;
        this.saldo = saldo;
    }

    public String exibirIdNome() {
        return (this.id + " - " + this.nome);
    }

    public void depositar(double valorDeposito) {
        this.saldo = this.saldo + valorDeposito;
    }

    public String exibirNomeSaldo(int opcao) {
        String texto;

        if (opcao == 1)
            texto = this.nome + " - R$ " + this.saldo;
        else
            texto = this.nome + " - US$ " + (this.saldo / 4.70);

        return texto;
    }
}
```

Programa.java

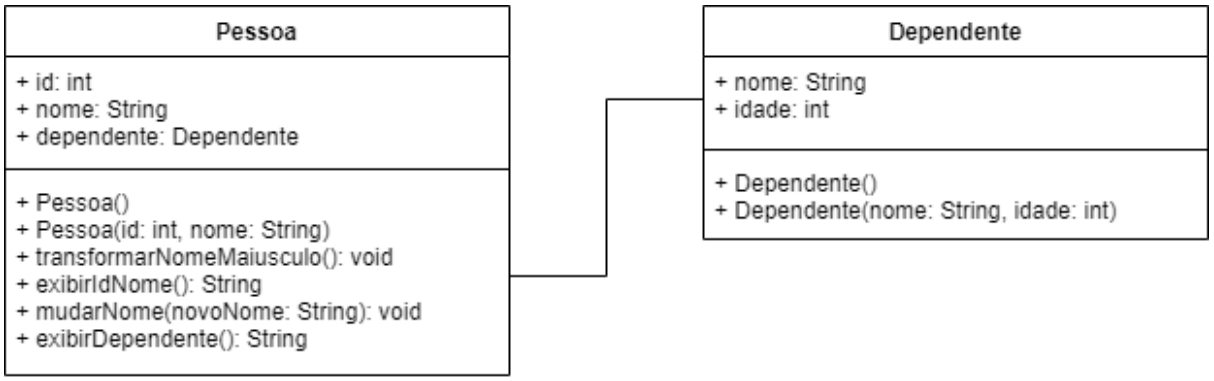
```
package classe_objeto;

import java.util.Scanner;

public class Programa {

    public static void main(String[] args) {

        Scanner ler = new Scanner(System.in);


        Pessoa a = new Pessoa();
        a.id = 1;
        a.nome = "Joseffe";
        a.saldo = 10;
```

```
        Pessoa b = new Pessoa(2, "André", 100);

        Pessoa c = new Pessoa();
        c.id = 3;
        c.nome = "Paulo";
        c.saldo = 5;

        int realDolar;

        System.out.printf("Deseja exibir o saldo em:");
        System.out.printf("\n1 - Real");
        System.out.printf("\n2 - Dólar");
        System.out.printf("\nDigite sua opção: ");
        realDolar = ler.nextInt();

        while(realDolar > 2) {
            System.out.printf("Erro! Opção inválida, escolha apenas as
opções abaixo:");
            System.out.printf("\n1 - Real");
            System.out.printf("\n2 - Dólar");
            System.out.printf("\nDigite sua opção: ");
            realDolar = ler.nextInt();
        }

        if (realDolar == 1) {
            System.out.printf(a.nome + "- R$ " + a.saldo + "\n");
        }
        else {
            System.out.printf(a.nome + "- US$ " + a.saldo / 4.70 +
"\n");
        }

        System.out.printf(a.exibirNomeSaldo(realDolar) + "\n");
        System.out.printf(b.exibirNomeSaldo(realDolar) + "\n");
        System.out.printf(c.exibirNomeSaldo(realDolar) + "\n");
        a.depositar(100);
        b.depositar(100);
        c.depositar(100);
        System.out.printf(a.exibirNomeSaldo(realDolar) + "\n");
        System.out.printf(b.exibirNomeSaldo(realDolar) + "\n");
        System.out.printf(c.exibirNomeSaldo(realDolar) + "\n");

        ler.close();
```

```
    }
}
```

7. Trabalhando com mais de uma classe

É perfeitamente possível e natural termos um software com mais de uma classe e tendo ligação entre elas, veja:



Ou seja, vamos ao código:

Dependente.java

```java
package classe_atributo;

public class Dependente {
    String nome;
    int idade;

    Dependente(){
    }

    Dependente(String nome, int idade){
        this.nome = nome;
        this.idade = idade;
    }
}
```

Pessoa.java

```java
package classe_atributo;

public class Pessoa {
    public int id;
    public String nome;
```

```
    public Dependente dependente;

    Pessoa(){

    }

    Pessoa(int id, String nome, Dependente dep){
        this.id = id;
        this.nome = nome;
        this.dependente = dep;
    }

    public void transformarNomeMaiusculo() {
        this.nome = this.nome.toUpperCase();
    }

    public String exibirIdNome() {
        return (this.id + " - " + this.nome);
    }

    public void mudarNome(String novoNome) {
        this.nome = novoNome;
    }

    public String exibirDependente() {
        return ("Nome: " + this.dependente.nome + " - " + "Idade: " +
this.dependente.idade);
    }
}
```

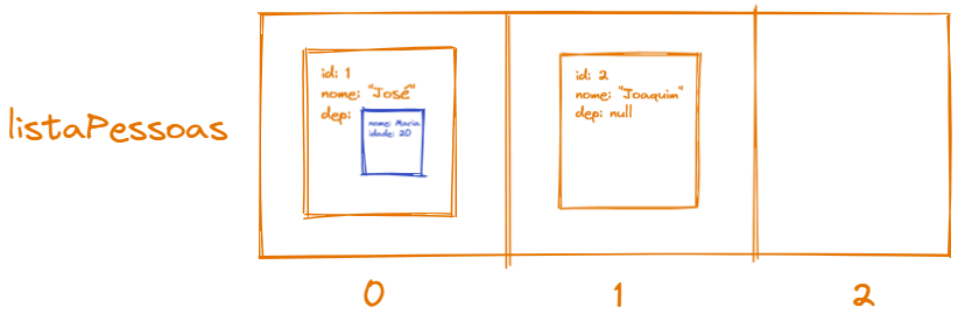
Programa

```
package classe_atributo;

import java.util.Scanner;

public class Projeto03 {

    public static void main(String[] args) {

        Scanner ler = new Scanner(System.in);

        // Cria um array contendo 10 posições para armazenar objetos
```

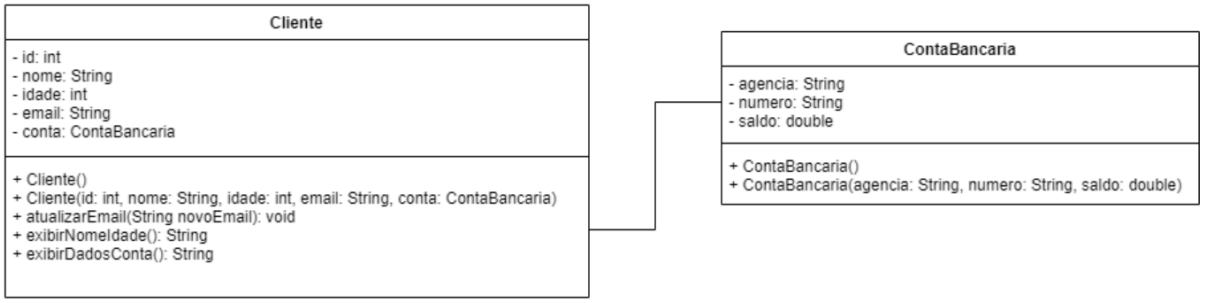
```
        Pessoa[] listaPessoas = new Pessoa[3];

        int id;
        String nome;
        char possuiDependente;
        int idadeDep;
        String nomeDep;

        for (int i=0; i<=2; i++) {

            Pessoa p;

            // Popula o objeto (preenche os atributos do objetos)
            System.out.printf("Digite o id da pessoa: ");
            id = ler.nextInt();

            System.out.printf("Digite o nome da pessoa: ");
            nome = ler.next();

            System.out.printf("Possui dependente? (S/N)");
            possuiDependente = ler.next().charAt(0);

            if (possuiDependente == 'S') {
                System.out.printf("Digite o nome do dependente: ");
                nomeDep = ler.next();

                System.out.printf("Digite a idade do dependente: ");
                idadeDep = ler.nextInt();

                // Instancia o dependente
                Dependente d = new Dependente(nomeDep, idadeDep);

                // Instancia (cria) o objeto pessoa com o dependente
                p = new Pessoa(id, nome, d);
            }
            else {
                // Instancia (cria) o objeto pessoa sem o dependente
                p = new Pessoa(id, nome, null);
            }

            // Adiciona o objeto no array "listaPessoas"
            listaPessoas[i] = p;
        }
```

```
        for (int i=0; i<=2; i++) {
            System.out.printf("\nPessoa %d criada com sucesso (%s)!",
listaPessoas[i].id, listaPessoas[i].nome);

            if (listaPessoas[i].dependente != null) {
                System.out.printf("\n Dependente: %s",
listaPessoas[i].exibirDependente());
            }
        }
    }
}
```

O programa, por trás dos panos, funcionará dessa maneira:



29. Crie as classes conforme o Diagrama de Classe (UML) abaixo. Crie Getters e Setters para todos os atributos das classes. Crie um programa que utilize essas classes para cadastrar 5 clientes em uma lista de clientes e pergunte para cada cliente se ele tem ou não conta bancária. Caso o cliente tenha, permita ele cadastrar os dados da conta bancária. Ao final exibir todos os clientes e suas respectivas contas bancárias, se houver.



ContaBancaria.java

```
package Ex29;
```

```java
public class ContaBancaria {
    public String agencia;
    public String numero;
    public double saldo;

    public ContaBancaria() {

    }
    public ContaBancaria(String agencia, String numero, double saldo) {
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = saldo;
    }
}
```

Cliente.java

```java
package Ex29;

public class Cliente {
    public int id;
    public String nome;
    public int idade;
    public String email;
    public ContaBancaria conta;

    Cliente(){

    }

    Cliente(int id, String nome, int idade, String email, ContaBancaria
conta){
        this.id = id;
        this.nome = nome;
        this.idade = idade;
        this.email = email;
        this.conta = conta;
    }

    public void atualizarEmail(String novoEmail) {
        this.email = novoEmail;
    }

    public String exibirNomeIdade() {
```

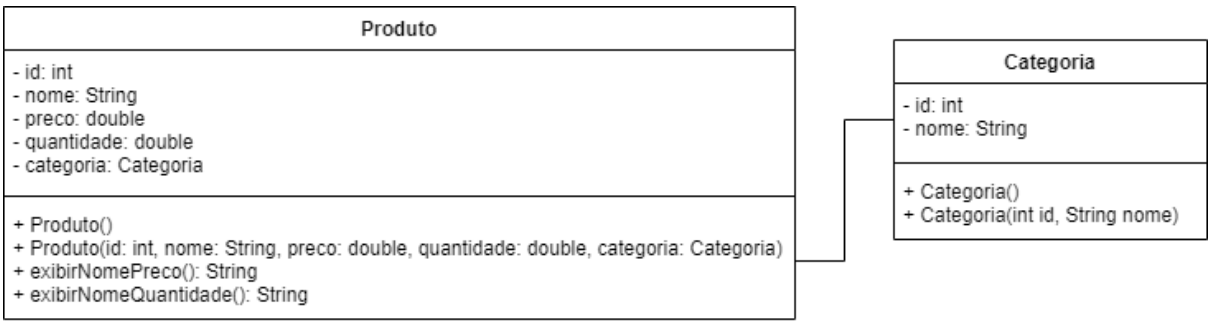
```
        return ("\nnome: " + this.nome + "\nidade: " + this.idade +
"\nemail: " + this.email);
    }

    public String exibirDadosConta() {
        if(this.conta != null){
            return "\nConta: " + this.conta.numero + "\nAgência: " +
this.conta.agencia + "\nSaldo: " + this.conta.saldo + "\n";
        }
        else {
            return("\nCliente sem conta cadastrada");
        }
    }
}
```

Programa.java

```
package Ex29;

import java.util.Scanner;

public class Programa {

    public static void main(String[] args) {
        int iter = 2;

        Scanner read = new Scanner(System.in);
        Cliente[] listaClientes = new Cliente[iter];
        Cliente c;
        String nome;
        int idade;
        String email;

        String hasAccount;

        ContaBancaria conta;
        String agencia;
        String numero;
        double saldo;

        for(int i=0; i<iter; i++) {
            System.out.printf("nome do %do cliente: ", i+1);
            nome = read.next();
            System.out.printf("idade do %do cliente: ", i+1);
```

```
            idade = read.nextInt();
            System.out.printf("email do %do cliente: ", i+1);
            email = read.next();

            System.out.printf("O cliente tem conta bancária? [S/N]");
            hasAccount = read.next();

            if(hasAccount.equals("S")) {
                System.out.printf("Informe os dados da conta:\n");
                System.out.printf("agencia: ");
                agencia = read.next();
                System.out.printf("núemro da conta: ");
                numero = read.next();
                System.out.printf("saldo: ");
                saldo = read.nextDouble();
                conta = new ContaBancaria(agencia, numero, saldo);
                c = new Cliente(i, nome, idade, email, conta);
            } else {
                c = new Cliente(i, nome, idade, email, null);
            }
            listaClientes[i] = c;
        }

        for(int i=0; i<iter; i++) {
            System.out.printf(listaClientes[i].exibirNomeIdade());
            System.out.printf(listaClientes[i].exibirDadosConta());
        }

        read.close();
    }
}
```

30. Crie as classes conforme o Diagrama de Classe (UML) abaixo. Crie Getters e Setters para todos os atributos das classes. Crie um programa que utilize essas classes para cadastrar 5 produtos em uma lista de produtos e pergunte para cada produto se ele tem ou não uma categoria. Caso o produto tenha, permita ele cadastrar os dados da categoria. Ao final, exibir todos os produtos e suas respectivas categorias, se houver.


Produto
- id: int
- nome: String
- preco: double
- quantidade: double
- categoria: Categoria
+ Produto()
+ Produto(id: int, nome: String, preco: double, quantidade: double, categoria: Categoria)
+ exibirNomePreco(): String
+ exibirNomeQuantidade(): String
Categoria
- id: int
- nome: String
+ Categoria()
+ Categoria(int id, String nome)