**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());

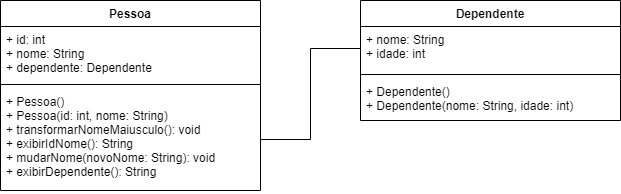
}

}

}

**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

package classe\_atributo;

public class Dependente {

*String* nome;

*int* idade;

Dependente(*String* *nome*, *int* *idade*){

*this*.nome = nome;

*this*.idade = idade;

}

}

Pessoa.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());

}

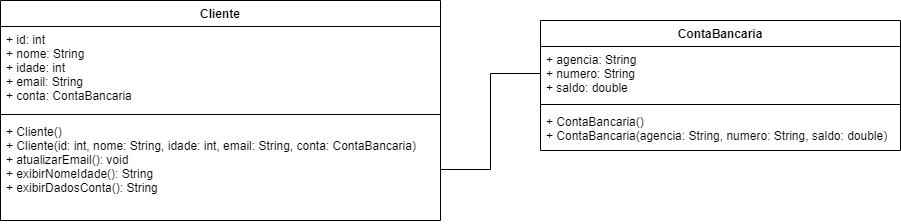
}

}

}

**8. Exercício**

53. Crie as classes conforme o Diagrama de Classe (UML) abaixo. 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.

****

\*correção: atualizarEmail(String novoEmail);