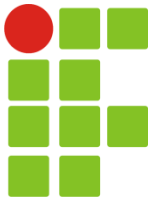


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS - ORIENTAÇÃO A OBJETOS

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

Introdução

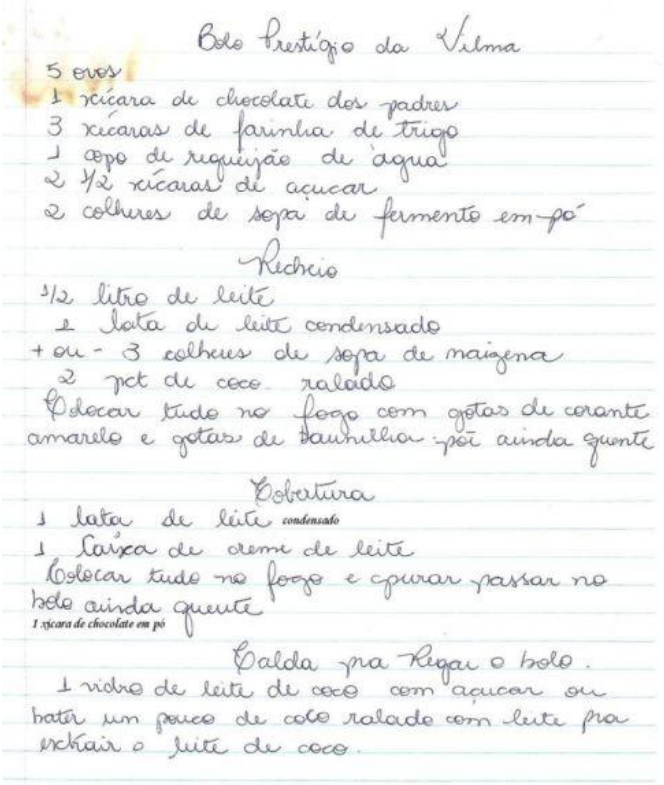
- Programação Orientada a Objetos (POO) é uma forma de programar que ajuda na organização do código.
- Vamos pensar nos objetos do mundo real e utilizá-los nos programas.
- Existem várias linguagens orientadas a objetos:
  - Java
  - C#
  - C++
  - Python

Introdução

- Com o uso da orientação a objetos iremos:
  - Concentrar as responsabilidades;
  - Reutilizar código;
  - Flexibilizar o programa;
  - Escrever menos;

Classes

- É a principal estrutura da POO.
- Reúne todas as características e os serviços disponíveis por seus objetos.
- Servem para especificar (modelo).
- Funciona como uma receita de bolo.



Bolo Prestígio da Vilma

5 ovos
1 xícara de chocolate dos padres
3 xícaras de farinha de trigo
1 copo de requeijão de água
2 1/2 xícaras de açucar
2 colheres de sopa de fermento em pó

Recheio

1/2 litro de leite
1 lata de leite condensado
+ ou - 3 colheres de sopa de maizena
2 pct de coco ralado
Colocar tudo no fogo com gotas de corante amarelo e gotas de baunilha - pôr ainda quente

Cobertura

1 lata de leite condensado
1 caixa de creme de leite
Colocar tudo no fogo e apurar passar no bolo ainda quente
1 xicara de chocolate em pó

Calda pra Regar o bolo.

1 vidro de leite de coco com açucar ou bater um pouco de coco ralado com leite pra extrair o leite de coco.

Objetos

- Nós comemos a receita do bolo ??
- Precisamos cozinhar um bolo (**objeto**), a partir de uma receita (**classe**), para então comê-lo.
- Podemos criar vários bolos a partir dessa classe. Porém, são objetos diferentes.

Classe x Objeto

X

Atributos

- São as características da classe.
- Por exemplo, uma Pessoa possui:
  - Nome
  - Idade
  - Nacionalidade
  - Profissão

```
public class Pessoa {

    String nome;
    int idade;
    String nacionalidade;
    String profissao;


}
```

Atributos

- Nos **objetos** esses atributos possuem valores.
  - Nome: Nickerson
  - Idade: 33
  - Nacionalidade: Brasileiro
  - Profissão: Professor

Métodos

- As classes também podem realizar determinadas operações.
- Cada classe possui seus próprios comportamentos.
- Os métodos são responsáveis por realizar uma operação com um determinado objeto.
- Pode ou não ter um retorno.
- Ex: Classe Conta (Número, Saldo, Limite)
  - sacar
  - verLimite
  - depositar

Métodos


```
class Conta {
  int numero;
  String dono;
  double saldo;
  double limite;

  void sacar(double quantidade) {
    double novoSaldo = this.saldo - quantidade;
    this.saldo = novoSaldo;
  }

}
```

Parâmetros

- Valores passados para o método.
- São valores necessários para que o método seja realizado.
  - Ex: Para sacar uma quantia de uma conta precisamos saber do valor.
  - Ex2: Para somar dois números é necessário saber quais são os números.
- Os parâmetros são informados na declaração dos métodos.

```
void sacar(double quantidade) {
  double novoSaldo = this.saldo - quantidade;
  this.saldo = novoSaldo;
}
```

Como usar isso tudo ??

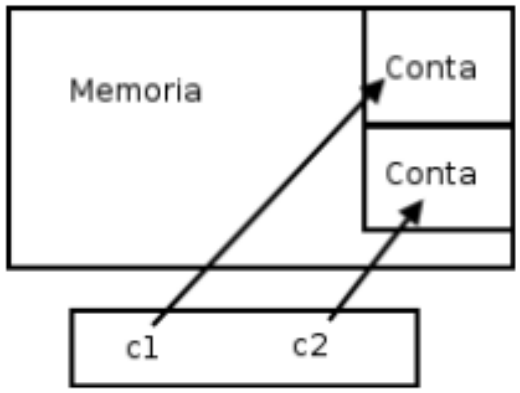

```
class Conta {
  int numero;
  String dono;
  double saldo;
  double limite;

  void sacar(double quantidade) {
    double novoSaldo = this.saldo - quantidade;
    this.saldo = novoSaldo;
  }
}
```

```
public static void main(String[] args) {

    Conta co = new Conta();
    co.numero = 1234;
    co.dono = "Nickerson";
    co.saldo = 100.00;
    co.limite = 1000.00;
    co.sacar(50.00);
}
```

Como usar isso tudo ??

- Ao declarar uma variável para armazenar um objeto, ela não guarda o objeto em si, mas uma referência para poder acessá-lo.
- Diferente dos tipos primitivos, por isso utilizamos o **new** na criação de objetos.

```
Conta c1;
c1 = new Conta();

Conta c2;
c2 = new Conta();
```



Construtores

- Ao chamar a palavra **new**, estamos criando um novo objeto.
- E para criarmos um novo objeto é necessário chamar um construtor.
- O construtor da classe possui o mesmo nome da classe.
- O bloco do construtor é executado toda vez que um novo objeto for criado.
- A classe pode ter mais de um construtor, porém, com a lista de parâmetros diferente.

Construtores


```
class Conta {
  int numero;
  Cliente titular;
  double saldo;
  double limite;

  // construtor
  Conta() {
    System.out.println("Construindo uma conta.");
  }

  //
}
```

Então, quando fizermos:

```
Conta c = new Conta();
```

```
Saída - (run)
run:
Construindo uma conta.
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Pacotes

- Utilizamos pacotes para organizar o projeto.
- Ao criarmos um pacote, um diretório também é criado.
  - Ex: package br.edu.ifrn.teste **(pacote)**
    br/edu/ifrn/teste **(diretório)**
- Classes relacionadas devem ser agrupadas em pacotes.

Visibilidade

- Podemos controlar o acesso aos atributos, métodos e construtores das classes.
- Existem 4 tipos de modificadores de acesso:
  - public
  - private
  - protected
  - default (padrão)

Exercícios

- Crie uma classe Java para funcionários. Ele deve ter o nome do funcionário, o departamento onde trabalha, seu salário (double), a data de entrada no banco (String) e seu RG (String).
- Você deve criar alguns métodos de acordo com sua necessidade. Além deles, crie um método recebeAumento que aumenta o salario do funcionário de acordo com o parâmetro passado como argumento. Crie também um método calculaGanhoAnual, que não recebe parâmetro algum, devolvendo o valor do salário multiplicado por 12..
- Teste-a, usando uma outra classe que tenha o main. Você deve criar a classe do funcionário com o nome Funcionario, mas pode nomear como quiser a classe de testes, contudo, ela deve possuir o método main.

Referências

- **Apostila Caelum:** https://www.caelum.com.br/apostila-java-orientacao-objetos/orientacao-a-objetos-basica
- H.M. Deitel, P.J. Deitel, **Java Como programar.**