INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
BAHIA

OO - Orientação a Objetos

Motivação

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- **Orientação a objeto**s é uma maneira de programar que ajuda na **organização** e resolve muitos **problemas enfrentados** pela **programação procedural**;

- Imaginem em um **grande sistema** em programação **procedural** quando nos encontramos na necessidade de **ler o código** que foi escrito por **outro desenvolvedor** e descobrir como ele **funciona internamente**;

Vantagens de OO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Em um sistema grande, simplesmente não **temos tempo de ler todo o código existente**.

- Orientação ajuda a **organizar e escrever menos**, além de concentrar as **responsabilidades** nos pontos certos, **flexibilizando** sua aplicação, **encapsulando** a **lógica de negócios**.

Criando um Tipo

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Considere um **programa** para um **banco**, é bem fácil perceber que uma **entidade** extremamente **importante** para o nosso sistema é a **conta**.

- O que toda conta **tem** e é **importante** para nós?

  - número da conta
  - nome do dono da conta
  - saldo
  - limite

Criando um Tipo

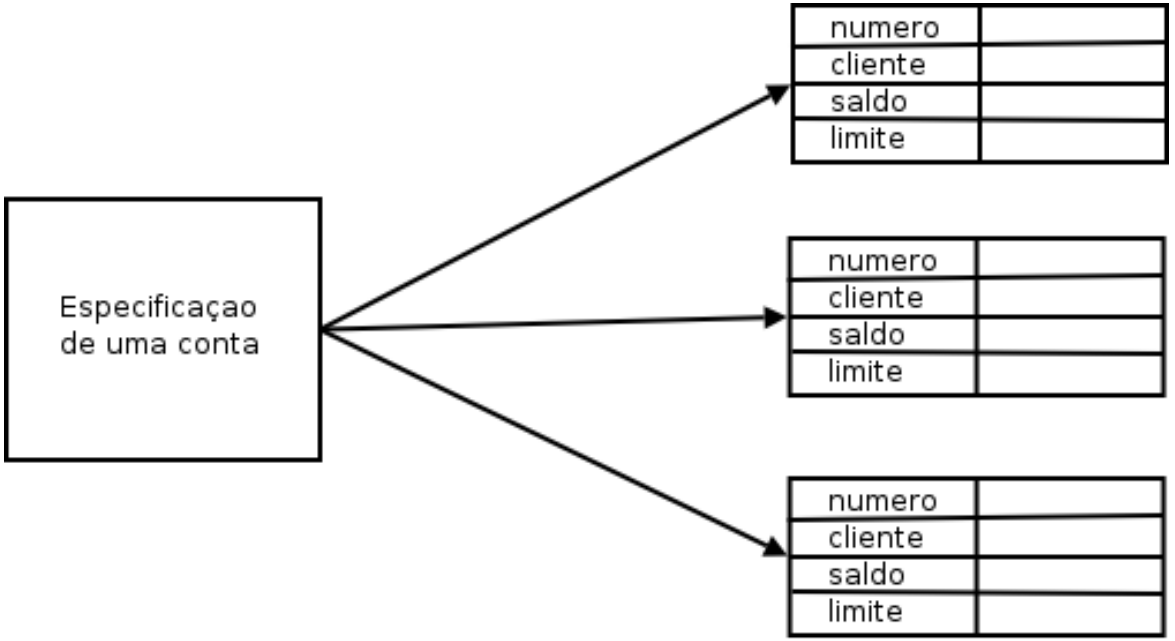
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- O que toda conta **faz** e é **importante** para nós? Isto é, o que **gostaríamos** de "**pedir** à **conta**"?

  - saca uma quantidade x
  - deposita uma quantidade x
  - imprime o nome do dono da conta
  - devolve o saldo atual
  - transfere uma quantidade x para uma outra conta y
  - devolve o tipo de conta

Criando um Tipo

- Com isso, temos o **projeto** de uma **conta bancária**;
- Podemos pegar esse projeto e **acessar seu saldo**?
- Antes, precisamos **construir** uma **conta**, para poder acessar o que ela tem, e pedir a ela que **faça algo**.



Criando um Tipo

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Ao **projeto** da conta, isto é, a **definição da conta**, damos o nome de **classe;**
- Ao que podemos **construir** a partir desse **projeto**, as contas de verdade, damos o nome de **objetos.**
- Toda **classe** têm uma série de **atributos** e **comportamentos** em comum, mas não são iguais, podem variar nos valores desses **atributos** e como realizam esses **comportamentos.**

Exemplo

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Exemplo classico: uma **receita de bolo**. Você **come** uma **receita** de **bolo**?

- Precisamos **instaciá-la,** criar um **objeto** bolo a partir dessa especificação (**a classe**) para utilizá-la.

- Podemos criar centenas de bolos a partir dessa classe (a receita, no caso), eles **podem ser bem semelhantes**, alguns até **idênticos**, mas são **objetos diferentes**.

Parece mas não

- A maior dificuldade inicial do paradigma da orientação a objetos é justo saber **distinguir** o que é **classe** e o que é **objeto**.

- É comum o **iniciante** utilizar, obviamente de forma **errada**, essas duas palavras como **sinônimos**.

Uma classe Java

- Vamos começar apenas com o que uma **Conta tem**, e não com o **que ela faz**;

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

    // ..

}
```

- **String** é uma **classe em Java**. Ela guarda uma **cadeia de caracteres**, uma frase completa.
- Por enquanto, declaramos o que **toda conta** deve ter. Estes são os **atributos** que toda conta, quando **criada**, vai **ter**.

Uma classe Java

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

    // ..

}
```

- Repare que essas variáveis foram declaradas **fora de um bloco**, diferente do que fazíamos quando tinha aquele **main**.

- Quando uma variável é declarada **diretamente** dentro do **escopo da classe**, é chamada de **variável de objeto**, ou **atributo**.

Criando e Usando Objetos

- Temos uma **classe em Java** que especifica o que todo **objeto dessa classe deve ter**;

- Mas precisamos de uma classe **Programa.java** e a partir dele é que iremos **utilizar** a **classe Conta**.

- Para **criar (construir, instanciar)** uma Conta, basta usar a **palavra chave new**;

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

    Conta minhaConta;
    minhaConta = new Conta();
    }
}
```

Criando e Usando Objetos

- Através da variável **minhaConta**, podemos acessar o **objeto** recém criado para alterar seu **dono**, seu **saldo** etc:

```
1  class Programa {
2      public static void main(String[] args) {
3          Conta minhaConta;
4          minhaConta = new Conta();
5
6          minhaConta.dono = "Duke";
7          minhaConta.saldo = 1000.0;
8
9          System.out.println("Saldo atual: " + minhaConta.saldo);
10     }
11 }
```

- É importante fixar que o ***ponto*** foi utilizado para acessar algo em **minhaConta**.

Métodos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA

- Dentro da classe, também **declararemos** o que cada **conta faz** e como **isto é feito** - os **comportamentos** que cada classe tem;

- Por exemplo, de que maneira que uma **Conta saca dinheiro**? Especificaremos isso **dentro** da **própria classe Conta**, e não em um local desatrelado das informações da própria Conta

- É por isso que essas "**funções**" são chamadas de **métodos.** Pois é a maneira de fazer uma **operação com um objeto**.

Métodos

- Queremos criar um método que **saca** uma determinada **quantidade** e não devolve **nenhuma informação** para quem acionar esse método;

```
1 class Conta {
2     double salario;
3     // ... outros atributos ...
4
5     void saca(double quantidade) {
6         double novoSaldo = this.saldo - quantidade;
7         this.saldo = novoSaldo;
8     }
9 }
```

- Quando alguém pedir para sacar, ele também vai dizer **quanto quer sacar**.

- Por isso precisamos declarar o método com **algo dentro dos parênteses** - o que vai aí dentro é chamado de **argumento** do método (ou **parâmetro**)

Métodos

```
1 class Conta {
2     double salario;
3     // ... outros atributos ...
4
5     void saca(double quantidade) {
6         double novoSaldo = this.saldo - quantidade;
7         this.saldo = novoSaldo;
8     }
9 }
```

- Essa variável é uma **variável comum**, chamada também de **temporária ou local**, pois, ao **final** da **execução** desse **método**, ela **deixa de existir**.

- Dentro do método, estamos declarando uma **nova variável**. Essa variável, assim como o **argumento**, vai **morrer no fim do método**, pois este é seu **escopo;**

Métodos

- Método para **depositar** uma **quantia**;

```
1 class {
2     // ... outros atributos e métodos ...
3
4     void deposita(double quantidade) {
5         this.saldo += quantidade;
6     }
7 }
```

- Observe que não usamos uma **variável auxiliar** e, além disso, usamos a **abreviação +=** para deixar o método **bem simples**.

- O **+=** soma quantidade ao valor **antigo do saldo** e guarda no **próprio saldo**, o **valor resultante**.

Invocação de Metodos

- Para mandar uma **mensagem** ao **objeto** e pedir que ele **execute** um **método**, também usamos o ponto. O termo usado para isso é **invocação de método.**

```
1  class TestaAlgunsMetodos {
2      public static void main(String[] args) {
3          // criando a conta
4          Conta minhaConta;
5          minhaConta = new Conta();
6
7          // alterando os valores de minhaConta
8          minhaConta.dono = "Duke";
9          minhaConta.saldo = 1000;
10
11         // saca 200 reais
12         minhaConta.saca(200);
13
14         // deposita 500 reais
15         minhaConta.deposita(500);
16         System.out.println(minhaConta.saldo);
17     }
18 }
```