Análise Orientada a objetos

Aula 4

Prof. Me. Juliana Costa-Silva

Na aula de hoje...

1. Entrada de Dados
2. Introdução a POO
3. Objetos
4. Classes
5. Métodos
6. Atividade
7. Leitura recomendada

Entrada de dados

A leitura do console (teclado) obtém dados a partir do objeto *System.in*.

- Objetos System.in leem somente bytes;
- Por isso é necessário transforma-lo em um objeto *InputStreamReader*.
- O *InputStreamReader* lê somente caracteres, então criamos um *BufferedReader*.
- O *BufferedReader* é um objeto que acopla vários caracteres lidos (uma frase por exemplo).

Utilizando *BufferedReader*

Criando um objeto *BufferedReader*

```
1   InputStreamReader ent =
2           new InputStreamReader(System.in);
3   BufferedReader ler = new BufferedReader(ent);
```

Declaração da String que receberá a leitura:

```
1   String s;
2   System.out.println("Digite um texto:");
```

Utilizando *BufferedReader*

Executando a leitura:

```
1   try {
2     s = ler.readLine();
3     System.out.println("Voce digitou: "+ s);
4   } catch (IOException ex) {
5     // Se acontecer um erro de leitura, imprime
6     System.out.println("Erro de leitura: "+ ex);
7   }
```

Para saber sobre as diferenças da classe Scanner leia:
[https://www.devmedia.com.br/entrada-de-dados-classe-scanner/21366]

Paradigmas de Programação

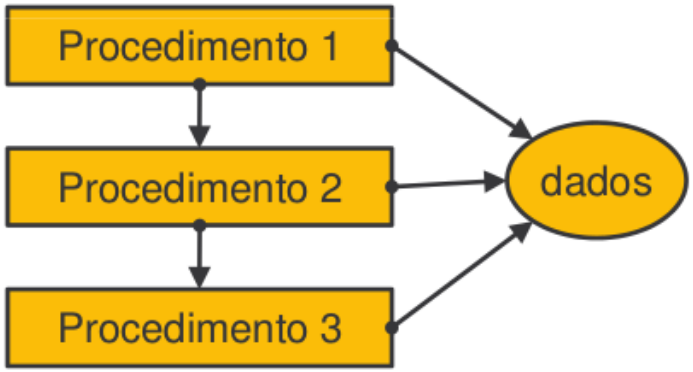
- **Definição:** Conjunto de regras e/ou hipóteses que governam a definição de um modelo.
- **Aplicação:** Auxilia na conduta do processo de busca da solução (modelo conceitual) de um problema.

Paradigmas de Programação:
Estruturado e Orientado a Objetos.

Paradigmas de Programação

Estruturado

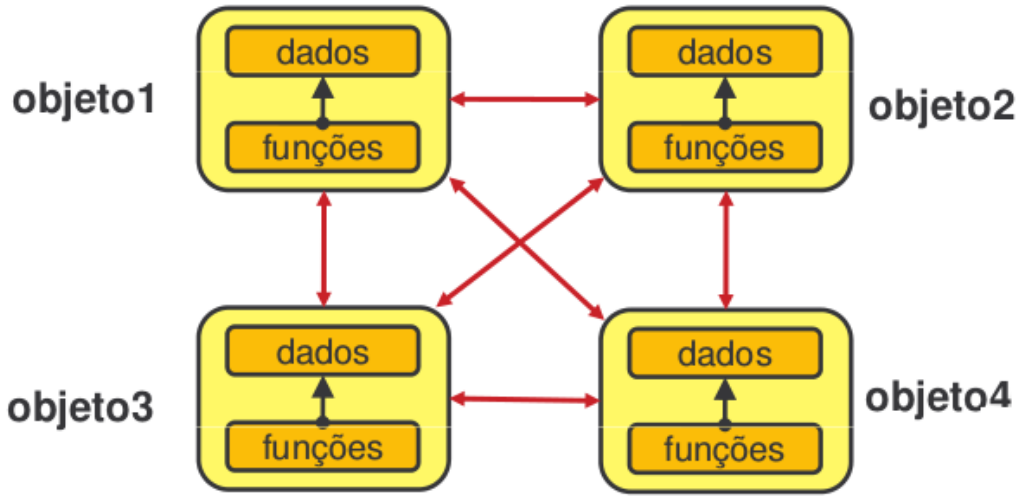
- **Estruturado:** Enfase em processos. Trabalha com a identificação de processos, que são aplicados sequencialmente sobre dados para realizar a computação desejada (foco nas ações, procedimentos e funções).
- Muitas variáveis/ muitas funções/ manutenção difícil



Paradigmas de Programação

Orientado a objetos

- **Orientado a objetos:** Enfase em dados. Trabalha com entidades comportamentais (estado e ações) independentes.
- Objeto é a unidade.
- Une dados e funções.



Orientação a objetos

Introdução

Orientação a objetos é uma maneira de programar. Ajuda a organizar o código e resolve muitos problemas.

POO - Definição:

- "... um termo geral que inclui qualquer estilo de desenvolvimento que seja baseado no conceito de **objetos** - uma entidade que exibe características e comportamento [Sintes, 2002]"
- "... uma maneira natural de pensar e escrever programas de computador [Deitel, 2010]".

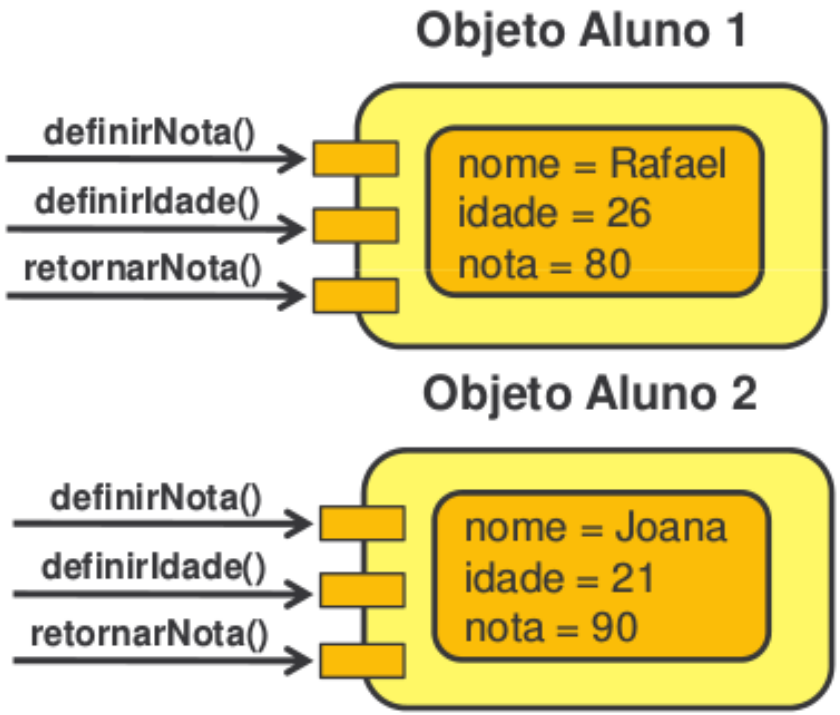
Orientação a Objetos

Por que mudar o Paradigma?

- ”**O ser humano conhece o mundo e gerencia sua complexidade através de objetos**. É como desenvolvemos nossa cognição“.
- **Teste:** Explique como funciona a administração de uma folha de pagamentos.
- E o programa em C para gerenciar isso? Como funcionaria?
- No mundo real, funcionário é funcionário.
- No aplicativo (C) um funcionário é RecFunc, que realiza tarefas implementadas nas funções *CalcSal, IRRFSal*, que estão codificadas nos programas Módulo1 e MóduloControle...

Orientação a objetos

- Na **Orientação a objetos**, não há separação dos dados e funções.
- Objeto é a unidade que une dados e funções.

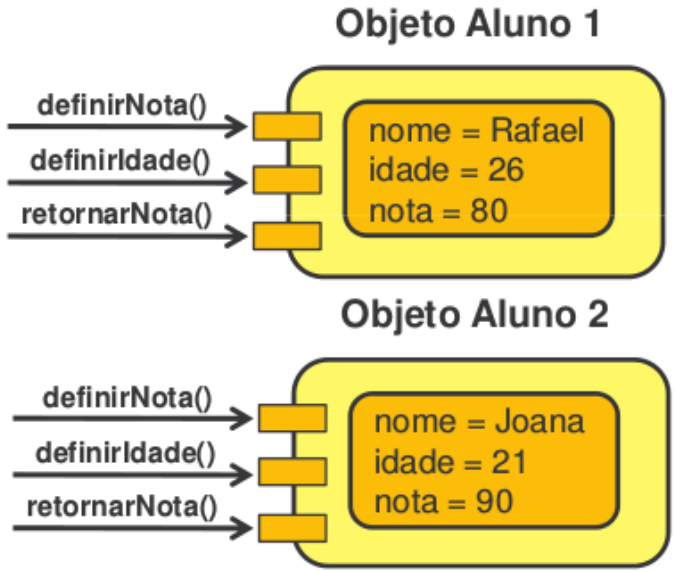


Exemplo

Estruturado

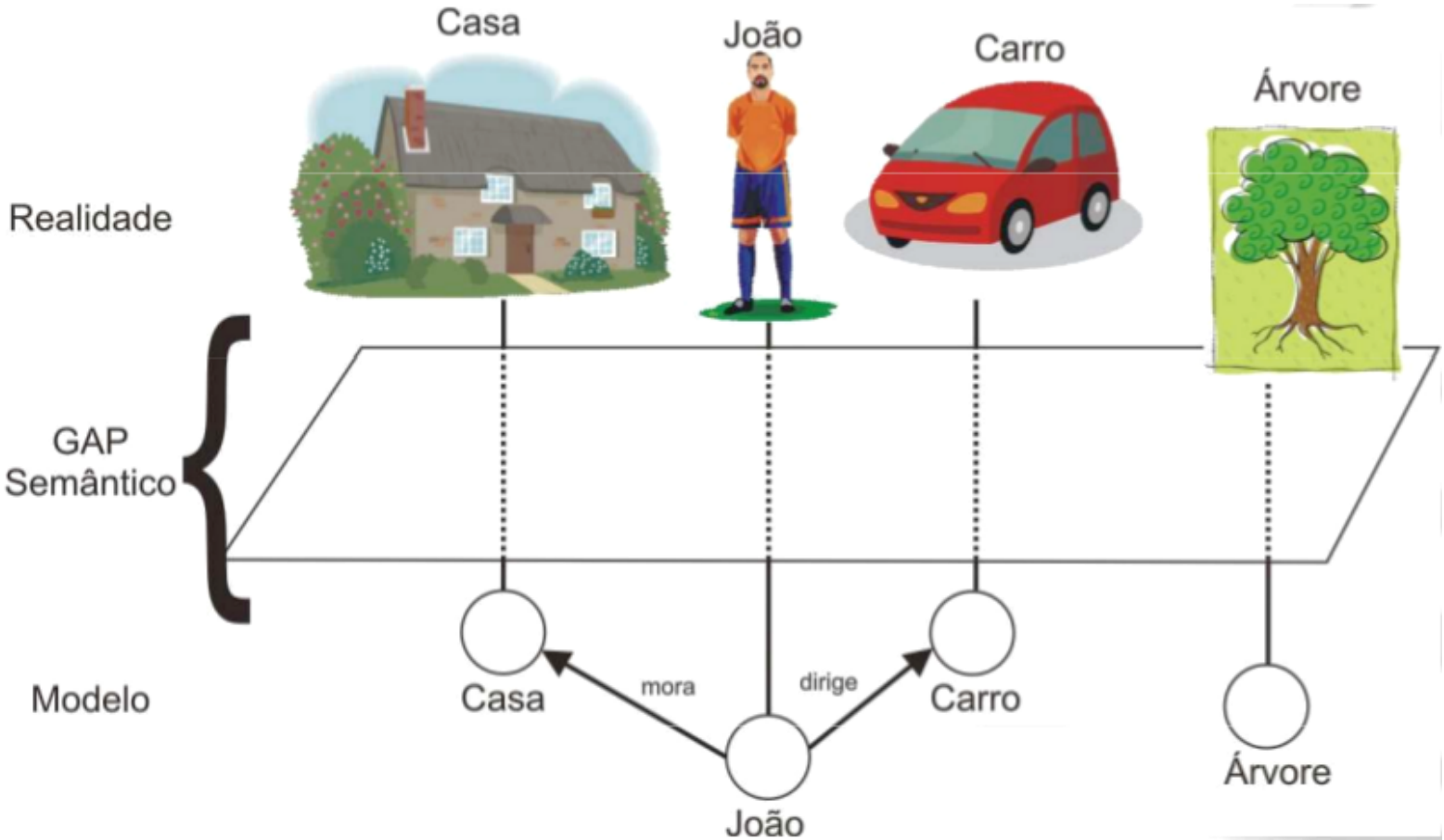
```
procedure definirNota(cod: integer; n: integer){
...
}

procedure definirIdade(cod: integer; i: integer){
...
}

function retornarNota(cod: integer):real{
...
}
```

Orientado a Objetos



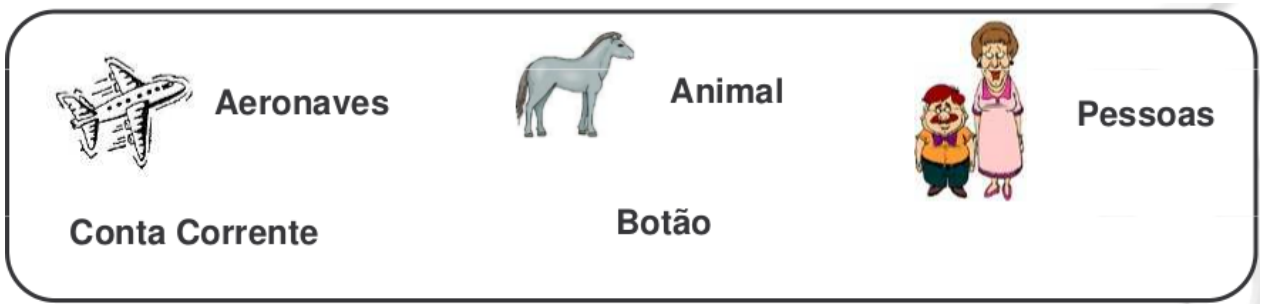
Gap semântico

- Diferença entre escopo dos problemas e das soluções



Objetos

- Um objeto representa qualquer coisa do mundo real



- Como no mundo real, os objetos possuem **estado** e **comportamento**.

Objeto Pessoa

Estado	Comportamento
Altura: 1,70m **Cor dos Olhos**: verde **Cor do cabelo**: loiro	anda() corre(velocidade) pula()

Objeto

- Como no mundo real, os objetos possuem **estado** e **comportamento**.
  - **Estado:** são informações sobre o objeto, como sua cor, tamanho, saldo, etc
  - **Comportamento:** são ações que o objeto pode realizar, como, depositar em uma conta, mudar a cor.

Quem define o que?

- Os **atributos** definem o **estado** do objeto: *tamanho, cor altura, etc*.
- Os **métodos** definem o **comportamento** do objeto: *andar, correr, depositar, etc*.

Objetos

Atributos

Atributos são similares a variáveis:

Modelo: LCD
Cor: preto
Polegadas: 17

Modelo: CRT
Cor: preto
Polegadas: 15

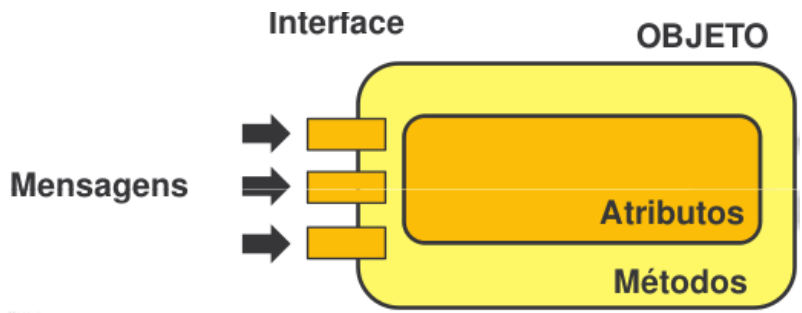
Métodos

Métodos são similares a sub rotinas, procedimentos ou funções:

ligar()
projetarImagem()
aumentarBrilho()
focarImagem()
desmagnetizarTela()

Objetos - Visibilidade

- Como regra geral, os **atributos** do objeto deverão ser ”**privados**“ e somente os **métodos** do objeto poderão ter acesso a eles. Por que?
- Imagine que você está comprando um produto pela internet. Se você tivesse a oportunidade de alterar o preço do produto, você não se sentiria tentado a faze-lo?
- O atributo preço, do objeto produto deve ser modificado somente por objetos (ou pessoas) autorizados.

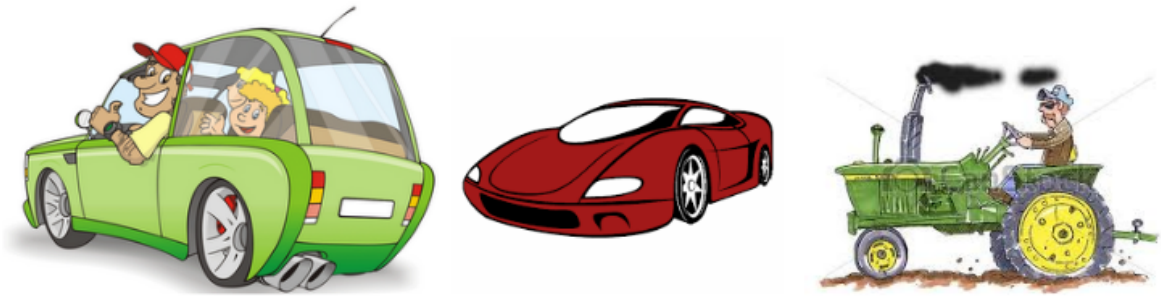


De onde vem um objeto?

Classe

- Modelo ou forma pela qual um objeto é criado
- Abstração de um conjunto de objetos que possuem os mesmos tipos de características (atributos) e comportamentos (métodos).
- Objetos se comportam de acordo com o comportamento da classe que os moldou.

Classe: Veículo



Carro de passeio | Carro Esportivo | Carro Rural

Atributos: cor, passageiros, portas, etc.

Métodos: ligar, desligar, acelerar, parar, etc.

Conta.java

Criando uma classe

```
1  public class Conta {
2      //Atributos
3      int numero;
4      String nome;
5      double saldo;
6      double limite;
7
8    void sacar(double quantidade){
```

TestaConta.java

Declarar e instanciar objeto de uma classe

```
1  public class TestaConta {
2    public static void main(String[] args) {
3      Conta minhaConta;
4      minhaConta = new Conta();
```

* Não esqueça de fechar as chaves!!

TestaConta.java

Editar objeto de uma classe

```
1       minhaConta.numero = 1;
2       minhaConta.nome = "Juliana";
3       minhaConta.saldo = 100.00;
```

TestaConta.java

Utilizar objeto de uma classe

```
1       System.out.println("O numero da minha conta: " +
            minhaConta.numero);
```

Conta.java

Método para sacar determinada quantidade de uma conta.

```
1   void sacar(double quantidade){
2     double novoSaldo = this.saldo - quantidade;
3     this.saldo = novoSaldo;
4   }
```

A expressão **void** antes do nome do método, indica que o método ”sacar“ não retorna nenhuma informação.
A variável novoSaldo ”morre“ no fim do método, assim como o argumento quantidade. Este é o escopo dessas variáveis.

Métodos com retorno

Método que realiza saque de uma conta, somente se houver saldo.

```
1  boolean sacarVerifica(double quantidade){
2    if (quantidade > this.saldo){
3      return false;
4    }else{
5      double novoSaldo = this.saldo - quantidade;
6      this.saldo = novoSaldo;
7      return true;
8    }
9  }
```

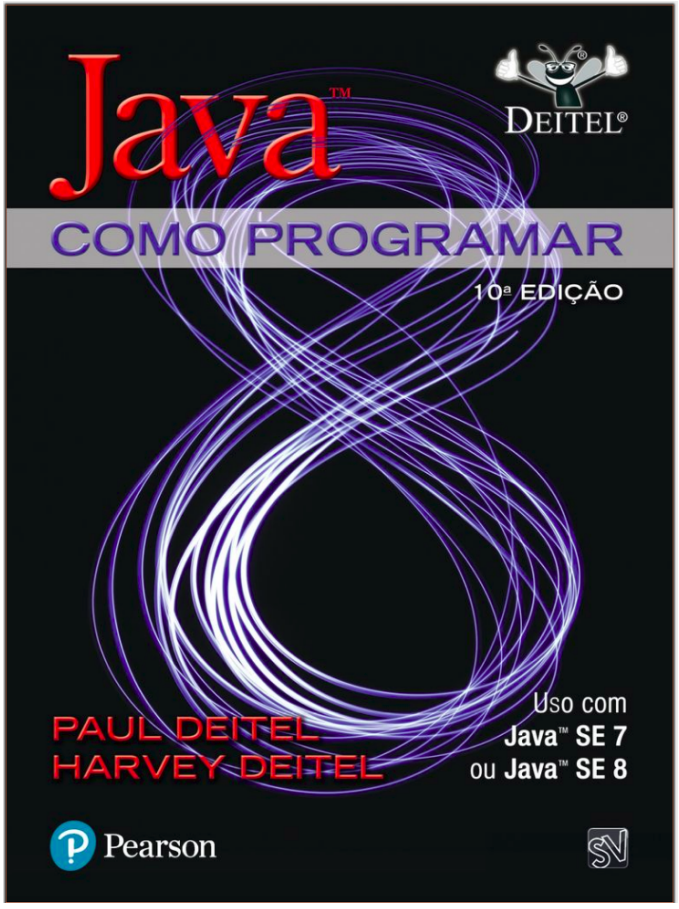
A expressão **boolean** antes do nome do método, indica que o método ”sacarVerifica“ retorna uma informação do tipo boolean.

Atividade de Aula

1. Implemente o método depositar, na classe conta.
2. Lembre-se que ele deve atualizar o saldo do cliente;
3. Teste o método criado.
4. Envie o código desenvolvido em aula, como atividade de aula.

Leitura complementar

Para mais informações sobre JAVA, leia:



Capítulo 3 a 7:
[Deitel, 2016]

Referências

Deitel, Paul J.; Deitel, H. M. (2010).
Java: Como programar. 8ª Edição.
Pearson.

Deitel, Paul J.; Deitel, H. M. (2016).
Java 8: Como programar. 10ª Edição.
Pearson.

Sintes, A. (2002).
Aprenda Programação Orientada a objetos em 21 dias.
Pearson Education do Brasil.