

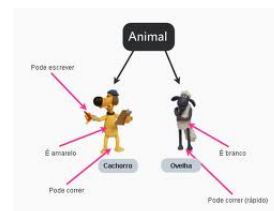
# Programação Orientada a Objetos

## Classes e Objetos

Robério Gomes Patricio  
roberiogomes@gmail.com

## Agenda

- Histórico
- Objetos
- Tipos Abstratos de Dados
- Classes
- Criação de Objetos
- Variáveis de Instância
- Métodos
- Passagem de Mensagens
- Sobrecarga de Métodos
- Construtores



2

## Orientação a objetos

- Surgiu no início dos anos 80.
- Grande evolução desde os Métodos Estruturados.
- Orientação a Objetos é uma abordagem (ou paradigma) para modelagem de sistemas.
- O paradigma da Orientação a Objetos compreende várias disciplinas:
  - Análise Orientada a Objetos
  - Projeto Orientado a Objetos
  - Programação Orientada a Objetos
  - ...
- Orientação a Objetos é um paradigma maduro e continua sendo amplamente utilizado.

3

## Orientação a objetos

- O princípio da orientação a objetos é mapear um modelo da realidade, visto como interação entre objetos, num modelo computacional de dados e programas, aproximando as soluções computacionais dos problemas do mundo real.
- A Orientação a Objetos contrasta da Programação Estruturada convencional na qual a estrutura e os dados são fracamente acoplados.



4

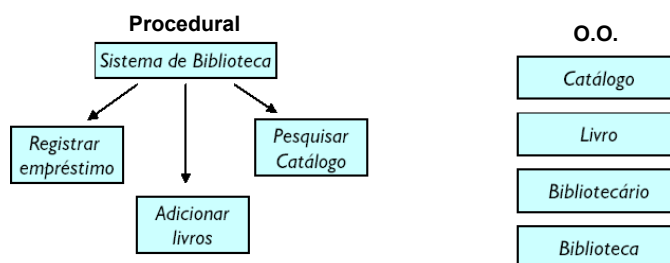
## Orientação a Objetos

- O paradigma de orientação a objetos consiste basicamente em organizar o software como uma coleção de objetos que incorporam a estrutura de dados e o comportamento, e trocam mensagens entre si.
- Foco do sistema é voltado aos dados (objetos), não às funções, já que o comportamento muda mais que a estrutura de dados.
- Uma das grandes vantagens da Orientação a Objetos é que as abstrações são mais próximas do mundo real.

5

## Orientação a objetos

- Comparando a forma de pensar Orientada a Objetos (focada nos dados) com a Estruturada/Procedural (focada nas funções).



6

## Orientação a Objetos

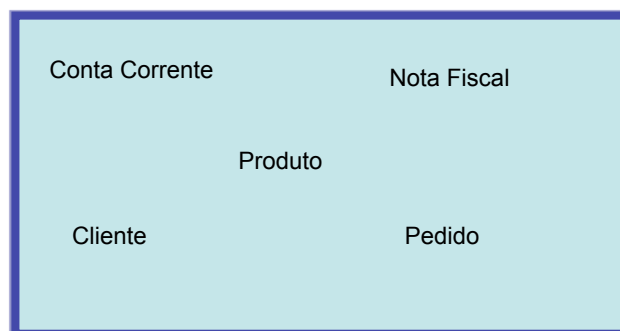
### Objetos

- Um objeto é qualquer coisa, real ou abstrata, na qual nós armazenamos dados e as operações que manipulam os dados [ Martin ].
- Um objeto ou uma instância é uma abstração de alguma coisa no domínio do problema, refletindo as capacidades do sistema de manter informações sobre ele, interagir com ele, ou ambos, um encapsulamento de valores de atributos e seus serviços exclusivo [ Yourdon ].

7

## Orientação a Objetos

### Objetos



8

## Orientação a Objetos

### Objetos

- Um objeto é uma entidade independente, composta por:
  - Estado Interno: uma memória interna em que valores podem ser armazenados e modificados ao longo da vida do objeto (conjunto de atributos ou variáveis de instância).
  - Comportamento: conjunto de ações pré-definidas (denominadas métodos) através das quais o objeto responderá a demanda de processamento por parte de outros objetos;

9

## Orientação a Objetos

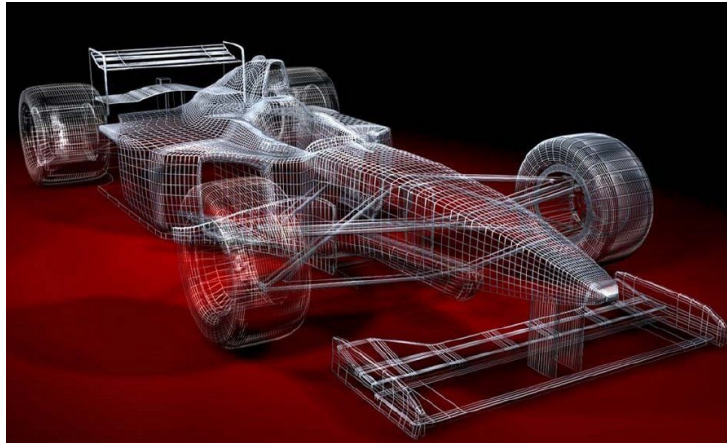
**Objetos** precisam de um molde, uma fôrma  
um modelo...

... é aí que entram as **Classes**!!!



10

# Classes



11

## Orientação a objetos

### Definindo Classes

- Suponha que precisamos trabalhar com um tipo de dados diferente dos tipos de dados já existentes, data, por exemplo.
- Precisamos, então, defini-lo.
- Antes de trabalharmos com variáveis do tipo Data, precisamos primeiro definir esse novo tipo de dados, conhecido como Tipo Agredado de Dados.

12

## Orientação a objetos

### Definindo Classes

- Vamos criar o novo Tipo Agregado de Dados com os seguintes campos: dia, mes e ano.

```
class Data
{
    int dia;
    int mes;
    int ano;
}
```

13

## Orientação a objetos

### Definindo Classes

- A classe Data que acabamos de criar é, inicialmente, o que chamamos de Tipo Agregado de Dados, ou Tipo Estruturado de Dados, ou ainda Registro.
- O Tipo Agregado de Dados é um tipo de dados definido pelo programador.
- A maioria das linguagens de programação suporta a definição de Tipos Agregados de Dados.

14

## Orientação a objetos

### Criando Objetos

- Em Java, para criar uma variável chamada `hoje` do novo tipo recém criado `Data`, faça:

```
Data hoje;
```

- Depois, para criar o objeto propriamente dito, faça:

```
hoje = new Data();
```

- Ou faça tudo junto:

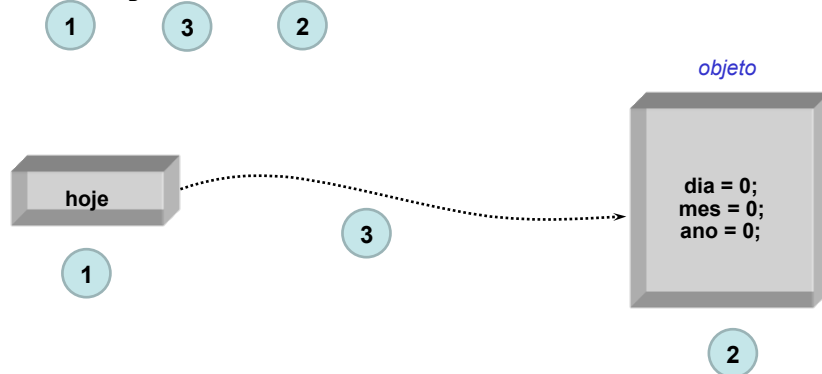
```
Data hoje = new Data();
```

15

## Orientação a objetos

### Criando Objetos

```
Data hoje = new Data();
```



16



## Orientação a objetos

### Acessando Propriedades dos Objetos

- As propriedades de um objeto são chamadas de **variáveis de instância**, pois cada instância tem seus próprios valores para suas variáveis de instância.
- Como acessar as variáveis de instância do objeto?
- Resposta: Através do operador “.”, como mostrado abaixo:

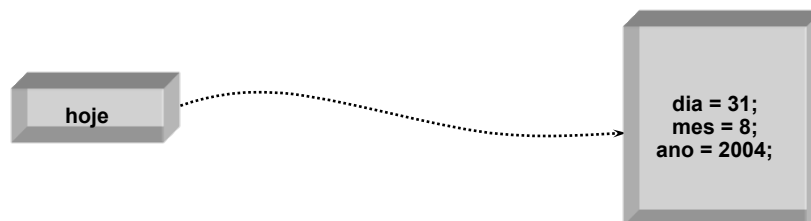
```
hoje.dia = 31;  
hoje.mes = 8;  
hoje.ano = 2004;
```

17

## Orientação a objetos

### O código completo...

```
Data hoje;  
hoje = new Data();  
hoje.dia = 31;  
hoje.mes = 8;  
hoje.ano = 2004;
```

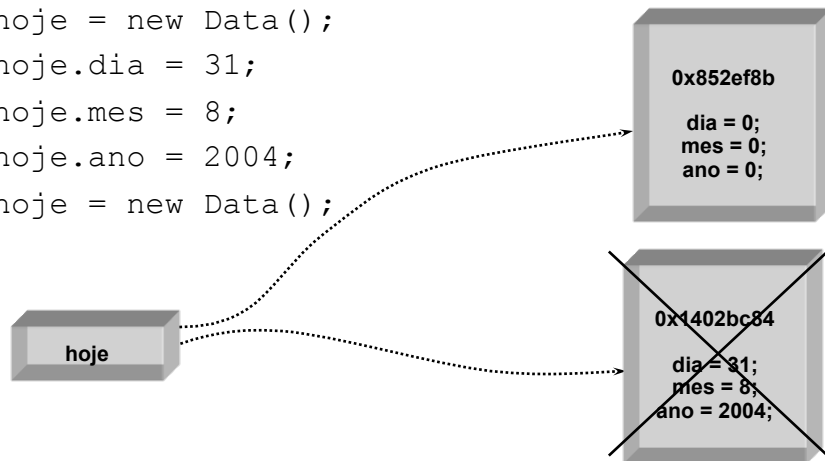


18

## Orientação a objetos

Apontando para um novo objeto...

```
Data hoje;  
hoje = new Data();  
hoje.dia = 31;  
hoje.mes = 8;  
hoje.ano = 2004;  
hoje = new Data();
```



19

## Orientação a objetos

- Classes podem ter propriedades (como vimos nos slides anteriores – dia, mes e ano).
- E também podem ter operações, as quais chamamos de **métodos**:

```
class Data  
{  
    int dia;        int mes;        int ano;  
  
    void exibir() {  
        System.out.print( dia + "/" + mes + "/" + ano );  
    }  
  
    int diasDoAno() {  
        int dias = 0;  
        . . .  
        return dias;  
    }  
}
```

20

## Orientação a objetos

### Tipos Abstratos de Dados

- A capacidade de uma classe ter definições de variáveis juntamente com definições de operações dá a ela o status de ***Tipo Abstrato de Dados***.
- Assim sendo, uma Classe, no seu sentido mais amplo, representa um Tipo Abstrato de Dados.
- Diversas linguagens, incluindo Java, suportam o conceito de Tipo Abstrato de Dados.
- Em linguagens OO, a Classe representa esse conceito.

21

## Orientação a objetos

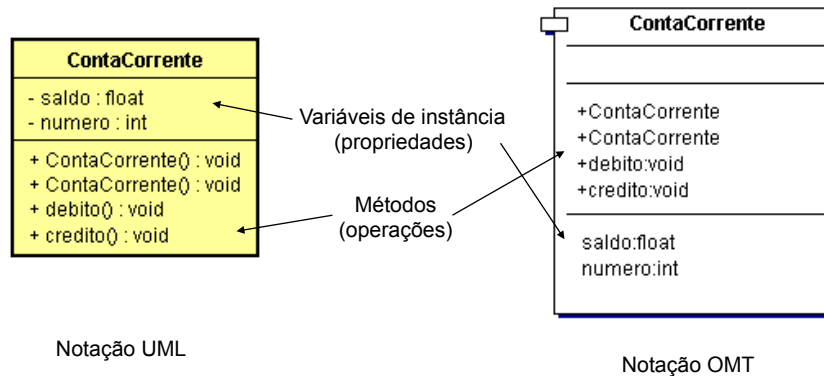
### Classes

- Classe é o agrupamento de objetos com a mesma estrutura de dados (atributos) e comportamento (operações) [ Rumbaugh ].
- Classe é uma coleção de um ou mais objetos com um conjunto uniforme de atributos e serviços, incluindo uma descrição de como criar novos objetos na classe [ Yourdon ].
- Descreve um conjunto infinito de objetos.
- A classe é o esqueleto (molde) para criação de objetos.
- Cada objeto é dito ser uma instância de uma classe.

22

## Orientação a objetos

### Representação de uma Classe



23

## Orientação a objetos

### Comunicação entre Objetos

- Para se comunicar com objetos, utilizamos o mecanismo de **passagem de mensagens**.
- Para executarmos um método de um objeto, enviamos uma mensagem a ele dizendo que operação ele deve realizar.
- O envio ou passagem de mensagens é feito através do operador "." seguido do nome do método.

```
...  
Data hoje = new Data();  
hoje.ano = 2003;  
hoje.incrementarAno( 2 );
```

24

## Orientação a objetos

### Classes X Objetos

- Cada instância da classe tem seus próprios valores para cada atributo, mas compartilham as operações com outras instâncias da classe.
- Em programação orientada a objetos, os termos principais têm a seguinte correspondência aproximada com termos tradicionais de programação:
  - classe → tipo
  - objeto → variável de um tipo
  - método → procedimento ou função
  - mensagem → chamada de procedimento ou função

25

## Orientação a objetos

### Sobrecarga de Métodos

- Métodos podem ter o mesmo nome e diferentes argumentos.
- Tal mecanismo é o que chamamos de **sobrecarga de métodos** ou **overloading**.

```
class Data
{
    int dia;
    int mes;
    int ano;
    void exibir() { ... }
    int diasDoAno() { ... }
    void incrementarAno() {
        ano++;
    }
    void incrementarAno( int anos ) {
        ano += anos;
    }
}
```

26

## Orientação a objetos

### Sobrecarga de Métodos

- Assinatura dos métodos:
  - nome + tipos e ordem dos parâmetros.
- Regra básica para sobrecarga:
  - métodos sobrecarregados não podem ter a mesma assinatura.
- Os parâmetros têm que diferir em tipo, ordem ou número.

27

## Orientação a objetos

### Sobrecarga de Métodos – Exemplos

```
int metodo( int a, double b );
void metodo( int a, long b );
    // OK - tipos dos parâmetros diferentes
void metodo( int a, long b, float c );
    // OK - número de parâmetros diferente
void metodo( double a, int b );
    // OK - ordem de parâmetros diferente
long metodo( int x, double b );
    // NOK - igual a 1a. Definição
```

28

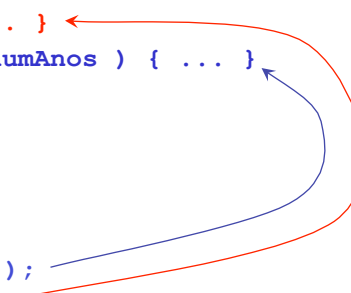
## Orientação a objetos

### Funcionamento do Mecanismo de Sobrecarga

- O método a ser executado é decidido com base na assinatura da chamada.

```
class Data
{
    ...
    void incrementarAno() { ... }
    void incrementarAno( int numAnos ) { ... }
}

// em um programa...
Data hoje = new Data();
hoje.ano = 2003;
hoje.incrementarAno( 2 );
hoje.incrementarAno();
...
```



29

## Orientação a objetos

### Construtores

- Construtores são como “métodos especiais” chamados no momento da criação de um objeto.
- Servem para inicializar objetos de forma organizada, ou seja, servem para “setar” o estado inicial de um objeto quando ele é criado.
- Pode haver mais de um construtor por classe (overloading ou sobrecarga de construtores).

30

# Orientação a objetos

## Construtores

```
class Data
{
    int dia;
    int mes;
    int ano;
    Data()
    { ano = 2004; }
    ...
}
```

Construtores são métodos especiais chamados quando o objeto é instanciado

## Utilizando...

```
// construtor default - sem parâmetros
Data hoje = new Data();
```

31

# Orientação a objetos

## Construtores Sobrecarregados

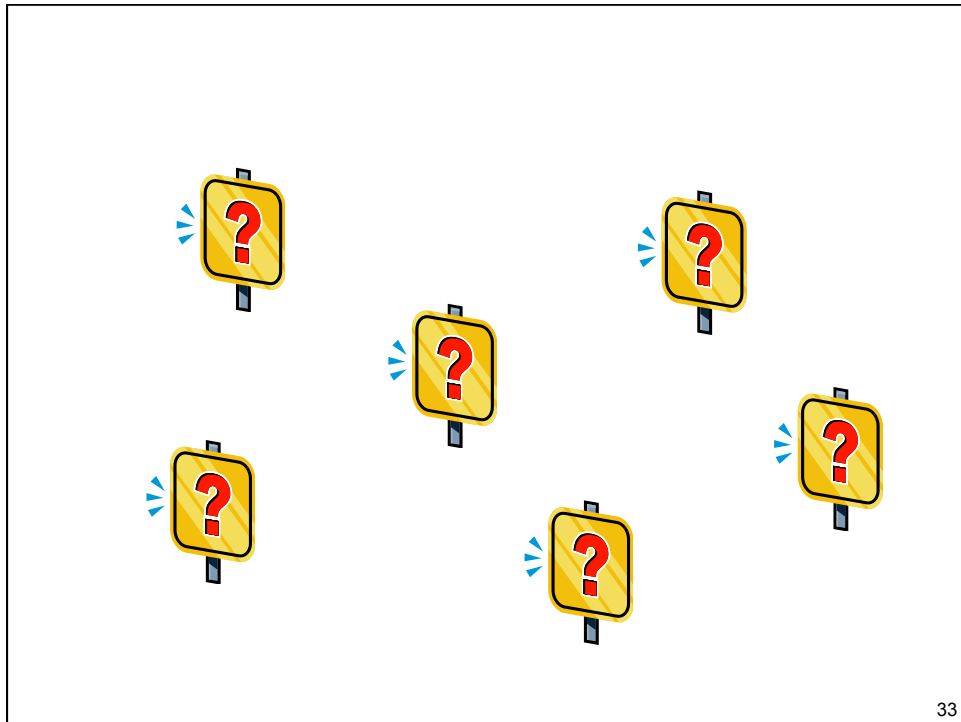
```
class Data
{
    ...
    Data()
    { ano = 2004; }
    Data( int numAnos )
    { ano = numAnos; }
    ...
}
```

## Utilizando...

```
Data hoje = new Data();
Data depois = new Data( 2005 );
...
```

32





33

## Revisão

1. Em que consiste o princípio da Orientação a Objetos?
2. O foco da Orientação a Objetos é nos dados ou nas funções? Justifique.
3. O que é um objeto? Dê exemplos.
4. Qual a composição de um objeto? Explique.
5. O que é uma variável de instância?
6. O que representam os métodos de instância?
18. Defina classe.
19. Como os objetos se comunicam? Explique.
20. O que é *overloading*?
21. O que diferencia, dentro de uma classe, métodos com mesmo nome?
22. O que são construtores?
23. Qual a finalidade dos construtores?
24. Construtores podem ser sobrecarregados? Justifique.

34

## Exercícios

1. Implementar uma classe Estudante com as seguintes propriedades:

- matricula (int)
- nome (String)
- sexo (char)
- um array chamado 'notas' com 4 posições (double []), sendo uma para cada nota parcial.

e as seguintes operações:

- void exibir() – mostra os dados do Estudante;
- void atribuirNota(int numProva, double nota) – coloca a nota no array na posição numProva-1;
- double lerNota(int numProva) – procura no array e retorna a nota existente na posição numProva-1.

35

## Exercícios – Dica de Resolução

```
public class Estudante {  
    int matricula;  
    double notas[] = new double[4];  
    //coloque aqui as outras variáveis de instância...  
  
    void exibir() {  
        System.out.println("matricula = " + matricula);  
        // mostre aqui as outras variáveis  
    }  
    // coloque aqui os demais métodos...  
}
```

36

## Exercícios

2. Implementar um programa chamado `TesteEstudante` que realiza as seguintes operações:
  - criar um estudante;
  - atribuir os valores 2004001, “Maria Silva” e ‘F’ para as variáveis de instância matrícula, nome e sexo, respectivamente;
  - preencher o array de notas com os valores 7.0, 8.2, 6.0, 5.5;
  - exibir na tela os valores de todas as propriedades do estudante;
  - exibir na tela os valores de todas as notas do estudante.
  - realizar outras operações para praticar.

37

## Exercícios – Dica de Resolução

```
public class TesteEstudante {  
    public static void main (String args[]) {  
        Estudante e = new Estudante();  
        e.matricula = 111;  
        // atribua aqui os valores às demais variáveis  
        e.atribuirNota( 1, 7.0 );  
        // atribua aqui as demais notas  
        e.exibir();  
        double nota1 = e.lerNota( 1 );  
        System.out.println ( “nota1 = ” + nota1 );  
    }  
}
```

38

## Atividade

3. Aperfeiçoar a classe Estudante adicionando as seguintes funcionalidades:
  - criar um método sobrecarregado ao método *atribuirNota* já existente, para atribuir a nota 0.0 (zero) ao índice passado como parâmetro – *atribuirNota(int indice)*;
  - criar um método sobrecarregado ao método *exibir* passando o título dos dados a serem exibidos – *exibir(String titulo)*;
  - criar três construtores sobrecarregados:
    - um default (sem parâmetros), que inicializa as variáveis de instância para os valores default (padrões);
    - outro que recebe somente nome e sexo do Estudante;
    - e um terceiro que recebe nome, sexo e matricula do Estudante.

39

## Atividade

4. Complementar o programa TesteEstudante com as seguintes operações:
  - criar três estudantes, cada um utilizando um construtor diferente:
    - sem parâmetros
    - passando nome e sexo
    - passando nome, sexo e matricula
  - executar os métodos sobrecarregados para os estudantes acima:
    - *atribuirNota(1, 7.0)*;
    - *atribuirNota(2, 5.0)*;
    - *atribuirNota(3)*;
    - *atribuirNota(4, 8.5)*;
    - *exibir()*;
    - *exibir("dados do estudante")*;

40

# Programação Orientada a Objetos

*Obrigado!!!*

Robério Gomes Patricio  
roberiogomes@gmail.com