

Programação Orientada a Objetos : Classes

Prof. Alexandre Krohn



Roteiro

- Classes
 - Definição
 - Atributos
 - Getters / Setters
 - Métodos
 - Instanciação
- Exercícios



Classes

- Uma classe é o “molde” para criação de objetos.
- Lembre-se : Tudo é objeto, portanto necessita de uma classe para ser feito.

Estrutura da classe

```
public class NomeDaClasse {
```

Nome da classe

```
    private int atributo1;  
    private double atributo2;  
    private String atributo3;  
    private boolean atributo4;  
    private char atributo5;
```

Atributos

```
    public NomeDaClasse() {  
        // implementação do método  
    }
```

Construtor

```
    public void metodo1(String a, int b) {  
        // implementação do método  
    }
```

```
    public int metodo2() {  
        // implementação do método  
        return atributo1;  
        // exemplo de retorno  
    }
```

Métodos

```
    public String metodo3() {  
        // implementação do método  
        return atributo3;  
        // exemplo de retorno  
    }
```



Criação de uma classe

- Exemplo: Um encontro de médicos precisa registrar seus inscritos. Isso é feito com uma ficha de papel.
- Como deve ser a classe para substituir a ficha?

Exemplo : ficha de inscrição



Nome _____

Endereço _____

Cidade _____ Estado _____ Tel. () _____

Email _____

INVESTIMENTO

R\$20 (vinte reais)


Pagamento da inscrição poderá ser efetuado na secretaria do Gruparj Petrópolis ou através de depósito no Banco do Brasil - Ag. 080-9 C/C. 71.576-X .

Enviar comprovante de depósito junto com a inscrição por email para a secretaria do Gruparj Petrópolis ou apresentar no dia do evento.

Cria-se uma nova classe

- Lembre-se das regras de nomenclatura usando CamelCase:

```
public class FichaInscricao {  
  
}
```



Atributos da classe

- Analisando a ficha, podemos constatar a existência das seguintes informações:
 - Nome
 - Endereço
 - Cidade
 - Estado
 - Telefone
 - E-mail



FichaInscricao : Atributos

- Todos os atributos de nossa classe até aqui podem ser considerados textuais, portanto serão definidos como sendo do tipo ***String***
- Além disso, para garantir o encapsulamento, todos eles são declarados como privados (***private***)

FichaInscricao : Atributos

```
public class FichaInscricao {  
  
    private String nome;  
    private String endereco;  
    private String cidade;  
    private String estado;  
    private String telefone;  
    private String eMail;  
  
}
```



Atributos da classe

- Falta algum atributo??
 - Nome
 - Endereço
 - Cidade
 - Estado
 - Telefone
 - E-mail

Atributos da classe


- Quanto custa a inscrição???
- Nome
- Endereço
- Cidade
- Estado
- Telefone
- E-mail
- Investimento

Todas as fichas
tem um valor de
investimento

Atributos da classe

- O valor de investimento é o mesmo para todas as inscrições:

- Nome
- Endereço
- Cidade
- Estado
- Telefone
- E-mail
- Investimento



Esse valor será
uma constante, pois sempre
será de R\$20,00

FichaInscricao : Constantes

```
public class FichaInscricao {  
  
    private static final double INVESTIMENTO = 20;  
  
    private String nome;  
    private String endereco;  
    private String cidade;  
    private String estado;  
    private String telefone;  
    private String eMail;  
  
}
```



FichaInscricao : Constantes

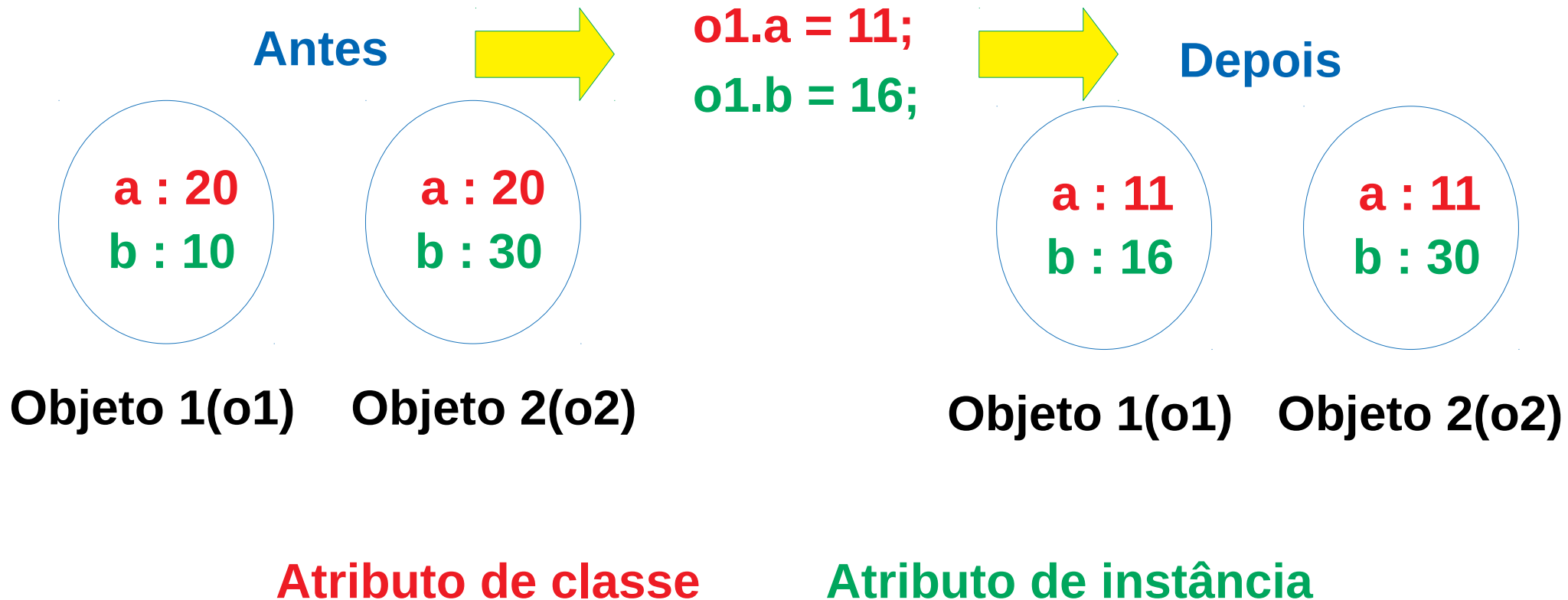
- O valor do investimento é o mesmo para todas as fichas, por isso usaremos uma constante. Além de ser uma constante, ela é também um atributo de instância



Atributos de classe **X** Atributos de instância

- Atributos de **instância** assumem **valores diferentes** para cada objeto
- Atributos de **classe** possuem o **mesmo valor** para todos os objetos de uma determinada classe, podendo ser variáveis ou constantes

Atributos de classe X Atributos de instância



Constantes

```
private static final double INVESTIMENTO = 20;
```

- **private** define que o atributo é privado, e só pode ser manipulado por métodos da própria classe
- **static** define que o atributo é de classe
- **final** informa que depois de receber um valor, este não poderá mais ser alterado
- **double** é o tipo de dados que usamos sempre que precisamos representar valores financeiros (\$\$\$)
- ***INVESTIMENTO*** : Os nomes de constantes sempre são representados em letras maiúsculas
- ***INVESTIMENTO*** = 20; Inicialização obrigatória de constante.

FichaInscricao : Outros atributos

- Vamos supor que a inscrição possa ser paga depois.
- Para isso, precisamos criar um atributo para saber se a inscrição foi paga ou não
- Fazemos isso com um atributo lógico (**boolean**)

FichaInscricao : Outros atributos

```
public class FichaInscricao {  
    private static final double INVESTIMENTO = 20;  
  
    private String nome;  
    private String endereco;  
    private String cidade;  
    private String estado;  
    private String telefone;  
    private String eMail;  
  
    private boolean pago;  
  
}
```



Getters / Setters

- Todos os atributos devem ser acessados através de métodos somente.
- Isso faz parte do princípio do encapsulamento.
- Para acesso simples aos atributos, cria-se métodos:
 - **Getters** : para **ler** dados
 - **Setters** : para **alterar** dados

Método getter

```
private String nome;
```

```
public String getNome() {  
    return nome;  
}
```

Sempre que queremos consultar o valor de um atributo de um objeto, fazemos isso através de seu método get

Método getter

```
private String nome;
```

```
public  
ret  
}
```

NÃO crie getters para
Atributos que você não quer
Que sejam acessados!

Método setter


```
private String nome;  
  
public void setNome(String nome) {  
    this.nome = nome;  
}
```

O método setter recebe um valor por parâmetro e o copia para o atributo da classe.

Método setter

```
private String nome;
```

```
public void setNome(String nome) {  
    this.nome = nome;  
}
```



Para isso, usa-se
a palavra **this**

this

- A palavra-chave *this* pode ser lida como “desta classe”
- Serve para desambiguar uma atribuição em um setter
- Sem ela, teríamos algo como:
nome = nome;

this

- A palavra-chave **this** pode ser lida como “desta classe”
- Serve para desambiguar uma atribuição em um setter
- Sem ela, teríamos algo como:

nome = nome;

Isso não faz nada!!!

Métodos

- Os métodos são os comportamentos do objeto. O que se pode fazer com ele.
- Nesse sentido, métodos possuem a seguinte estrutura:

```
<modificador> <tipo de retorno> <nome do método> ( <lista de parâmetros> )  
{  
    // corpo do método  
}
```




Métodos : Modificador

- Modificador informará como o método **será visto** (ou não) pelas outras classes : Pode ser **public**, **protected**, **private** ou **default**



Métodos : Tipo de Retorno

- O tipo de retorno indica o que o método retornará. Pode ser um tipo primitivo, um objeto de uma classe ou mesmo nada
- Métodos que não retornam nada retornam ***void***

Métodos : Nome do método

- O método precisa possuir um nome, sem espaços, **começando com letra minúscula** e usando notação CamelCase.

Métodos : Lista de parâmetros

- O método possui uma lista de parâmetros, que pode variar desde nenhum parâmetro até vários parâmetros, sempre indicados por seu tipo e nome e separados por vírgulas, quando houver mais do que um.
- Parâmetros são informados entre parênteses.

Métodos : Corpo do Método

- É onde é descrita a lógica interna de funcionamento do método
- Todo o código do corpo do método fica entre os delimitadores { e }

Métodos : Exemplos

```
public int soma(int a, int b) {  
    int c = a + b;  
    return c;  
}
```

```
public void incrementa() {  
    contador++;  
}
```

```
public String maiuscula(String texto) {  
    return texto.toUpperCase();  
}
```



Construtores

- Construtores são métodos especiais, que não possuem tipo de retorno, e que tem o mesmo nome da classe
- Sua função é inicializar os atributos do objeto no momento de sua criação



Construtores

- Uma classe pode possuir mais do que um construtor.
- O que os difere é a sua lista de parâmetros



Múltiplos Construtores

```
public FichaInscricao(String nome, String endereco,  
    String cidade, String estado, String telefone, String eMail) {  
    this.nome = nome;  
    this.endereco = endereco;  
    this.cidade = cidade;  
    this.estado = estado;  
    this.telefone = telefone;  
    this.eMail = eMail;  
}  
  
public FichaInscricao(String nome, String telefone) {  
    this.nome = nome;  
    this.telefone = telefone;  
}
```



Construtor Padrão

- Sempre que não definimos um construtor para a classe, ela tem um construtor padrão que não faz nada, com o próprio nome da classe.
- Se criarmos construtores com parâmetros, precisamos recriar o construtor padrão

Construtor Padrão

```
public FichaInscricao() {  
}
```

Sem reprogramar o construtor padrão pode não ser possível criar objetos de uma classe, por não possuir todos os valores necessário para os parâmetros no momento da construção do objeto



Instanciação

- Na instânciação, ou criação de um objeto, precisaremos de uma variável do tipo do objeto, da palavra-chave **new** e de um construtor da classe.



Instanciação

- A palavra-chave **new** realiza a **alocação de memória** para o objeto
- Na sequência o **construtor inicializa os atributos** da classe com os parâmetros, ou com código contido no construtor

Instanciação : Exemplos

```
FichaInscricao f1 = new FichaInscricao();
```

```
FichaInscricao f2 = new FichaInscricao("Maria Angélica",  
    "Av. Tomazini, 456", "Bom Princípio", "RS",  
    "mangel@ucs.br", "55 45678905");
```

```
FichaInscricao f3 = new FichaInscricao("Airton Madureira", "55 990822347");
```

Instanciação : erro

- Apenas declarar a variável não cria um objeto

```
FichaInscricao f4;
```

Ao tentar executar um método do objeto f4, receberemos um erro do tipo

NullPointerException

Exception in thread "main" [java.lang.NullPointerException](#)

Chamada de métodos

- Para chamar um método de um objeto, usamos o nome da variável e depois um **ponto (.)** antes do nome do método.
- Exemplos:

```
f3.mostraInscricao();  
f1.setNome("João da Silva");  
String texto = t1.maiuscula("meu texto");
```




Demo

- Vamos baixar o projeto Primeiro, importá-lo no Eclipse e depois executá-lo.
- Analisem o código das classes Teste e FichaInscricao



Dúvidas?





Atividades

- Execute as atividades presentes no documento

02.Lista.de.Exercícios.POO.pdf



Referências

- **Rafael Santos, Introdução A Programação Orientada A Objetos: USANDO JAVA**

Próximos passos



- Estruturas de Controle